WEEK-1 & 2 Signals & Systems (P)

INTRODUCTION TO MATLAB

https://www.mathworks.com/products/matlab.html

- Matrix CALCULATOR
- **Matrix** is the basic data element

WHY MATLAB? (advantages)

MATLAB is high-level languages and mathematical programming environments for:

- Interactive Mathematical Calculations
- Visualization: availability of graphical tools (2D and 3D color graphics)
- Programming, algorithm development & Easy to use: gives integrated editor/debugger to write, modify, debug the program,
- Numerical computation: linear algebra, optimization, control, statistics, signal and image processing, etc.
- Auto dimensioning - Eliminates data type declarations
- Various In- Built Functions
- **m-file --- two types: m- function file** and **m script files**
  1. **m- Function file**: In- Built Functions
     - User can use these functions to perform a task
     - It is ASCII files containing the algorithms
     - Algorithms on specific topics such as Control Systems, Signal Processing, Neural Network, Fuzzy Logic, Communication, etc.
     - **User can create their own function files using appropriate commands**
  2. **m - script files**
     - Editing / coding text pad: Here, the user can code /edit various lines here, using various MATLAB commands / bult-in m-function files
     - It is convenient to modify and save a **script m-file** each time if a change of parameters is desired.
- Function m-files provide a sensible alternative. Unlike script m-files, function m-files can accept input arguments as well as return outputs.

✓ Basics of MATLAB workspace

How to run MATLAB in Lab

- Open to the Windows operating environment.
- Create a new folder in the desktop and rename it with your roll number and open that folder.
- Double-click the 'MATLAB' icon; this will route the terminal to the MATLAB Command Window directly, to a MATLAB prompt.
- There are generally 3 sub-windows. They are Command Window, Current Folder or (Files), Variables (Workspace).

- Command Window is the platform to work with MATLAB. It is necessary to route this Command Window to the above-created directory (folder) by using the change directory "cd" command. Use *pwd* (*p*resent *w*orking *d*irectory) OR *ls* (*l*ist files) commands if necessary.
- One may see >> (prompt) in the Command Window. One may type MATLAB commands here one by one. OR
- Select the "HOME" menu on the top left corner and under this click "New Script". An editor window will appear named "untitledx", where x refers to numbers 1, 2, 3… and the Command Window will be pushed to bottom.
- One may type MATLAB commands in this blank editor window "untitledx" by selecting it.
- It is compulsory to save it by selecting "Save" OR "Save As" under top middle menu "EDITOR" by giving a convenient name. This will be the script "m-file". While saving, make sure that one should save the "m-script file" in your created "folder" on the desktop for later easy access.
  NOTE: **Do not give the name with only numbers or only letters. Preferred one is convenient letters followed by numbers.** This is to avoid the any **clash** between **built-in function "m-file"** with the **created "m-script file"**.
- This is a window to edit/write MATLAB code/codes. Then it can be compiled and used for running as the program code.
- By using the appropriate commands, one can invoke various activities/built-in functions /create variables, etc.

✓ Creating m-script files or editor window -- second method

- Inside the opened folder (your roll No.), create a new file by right clicking and selecting new document and empty document. This will create a new file.
- Rename the file with some filename with an extension **.m** E.g. :- **filename1.m** {**Do not give name with only numbers or only letters. Preferred one is convenient letters followed by numbers.** This is to avoid the any **clash** between **built-in function "function m-file"** with the **created "m-script file**.} Example: MATLAB has the function file named "poly.m". If **poly.m** is used to save as one of the m-script files, then whenever any built-in commands are used which may internally uses this function file "poly.m", the **MATLAB compiler have a confusion in execution. It leads to an error output, which is very difficult to debug.**
- Type the required sequence of MATLAB commands in this "m-script file" and save it.
- To run/execute this m-script file, click on the MATLAB Command Window and type only the filename {**without .m extension**} OR use "**Run**" button in top menu in "Editor".
- When "m-script file" runs, MATLAB execute the commands in the order they are written just as if they were typed in the MATLAB Command Window.
- To exit the "m-script file", click on top right corner "close" (×) button in the Editor window.

- You can also use **Ctrl C** to come out in the half the way of the program execution, or infinite loops, or if any.

NOTE : Do not give the "m-script file" name same as built-in "function m-file" name.

Examples: To create a matrix and find the inverse and determinant. Following is the code. Create an m-script file and save it as "ex1.m". Then type the following,

```
clear, clc,
% To compute inverse of a matrix and Determinant
A  = [1 2 9; 4 4 6;7 8 7];
disp('The inverse and determinant of matrix A is:')
A_inverse = inv(A)
A_determinant = det(A)
```

Note: 2nd line is the "comment line", and A, A_inverse, and A_determinant are the variables.

✓ Various commands helpful to work with MATLAB in are:
who whos clc clear load save path pause help lookfor edit etc.

❖ ***HELP OR DOC***

If we know the name of a MATLAB function and would like to know how to use it, we write the following command in the command window: > help > help sin e.t.c.
If we do not know exactly the keyword of any built-in function file, and to find it one may use "lookfor". Eg: inverse, we write:

>> **lookfor inverse** similarly, >> **lookfor sine**

It convenient to use 'help' command, for quick reference.
>*help '--list'* will give the list of topics related to help.

- **Familiarize the following useful commands in the MATLAB:**

>who	lists variables in workspace
>whos	lists variables and their sizes
>clc	clear command window
>clear	clear workspace
>save session1 a b	to save variables a & b of the workspace in session1
>load session1	to load variables to workspace

>disp ('I have successfully completed MATLAB basics')
> R = input ('How many apples') ----> gives the user the prompt in the text string and then waits for input from the keyboard.

✓ ***Saving the contents of the variable to "dat" file:***

There are two approaches:

First approach: Let **a** and **b** are variables
> **save myfile.dat a b** ---- > saves a, b in myfile.dat --- > This will save the contents of the variables "a" and "b" in binary form, which manually cannot be modified.

One can see the file contents by opening it in notepad/text environment but cannot comprehend anything from it.

Second approach:

> **save myfile.dat a b -ascii** ---- > saves a, b in myfile.dat --- > This will save the contents of the variables in ascii form, which can be manually modified .

One can see the file contents by opening it in notepad/text environment and add/delete/append the values.

>load **myfile.dat a b** ------ > is used for loading the contents to workspace. Here, the contents of **a** and **b** are loaded into a new variable, **myfile**.

❖ ***MATLAB* as calculator**

>pi^pi-10;
>cos(pi/4);
>ans*ans; the result will be $\frac{\sqrt{2}}{2}\times\frac{\sqrt{2}}{2}$ = ½ because the first output is eliminated, only last output is kept in the form of "ans".

➢ ELEMENTARY MATRIX OPERATIONS (Quick Reference)

✓ Entering Arrays/Matrices with real elements (use of : operator)

M = [first:increment:last] -----> Array with elements
M = linspace(first,last,n) -----> Array with n elements linearly spaced
M = logspace(first,last,n) -----> Array with n elements logarithmically spaced

To Enter an Array

>M = 0:1:5 = [0, 1, 2, 3, 4, 5]
>M = linspace(1,5,3) = [1,3,5]
>M = logspace(1,5,3) = [10,1000,100000]
>M = [(2+4/5), sqrt(2.8), 4.2,7]

To Enter a matrix with real elements

>M = [1 2 3; 4 5 6]
>M = [1,2,3; sqrt(2.8),5.2,5e-2]

✓ Vectorization

There are two approaches :

First approach:

```
> sine = [ ];                    %initializes all values of vector s to zero
> for t = 0:5                    %note that the index must be integer
> sine(t+1) = sin(0.2*pi*t);     %since we want values of s every 0.2 s we must
                                 multiply t by 0.2.
>end
```

Here, variables "t" and "sine" are arrays and containing respective values.
sin(.) is the built-in function m-file.

Note also that for t = 0 the output value computed is stored in variable "sine" in index "1". Hence "t+1" is used is index and index of the array always starts counting from 1 in MATLAB not from "0".

Second approach:

```
> t = 0:0.2:1
> sine = sin(0.2*pi*t)
```

The result of these two commands generate sine values in variable "sine" (sine function) at time instants (value of n) 0, 0.2, 0.4, 0.6, 0.4, 1.

This approach is preferable since MATLAB executes faster with the vectorization

Entering matrices with complex elements

a = 2 + 5i ------> complex number

A = [5+3j 7+8j; 9+2j 6+5j] -----> 2×2 complex matrix

A = [1 2;3 4] + i*[5 6;7 8]

```
>>z=3+j*4;%note the multiplication sign;
>>zs=z*z;%or z^2 will give you the same results;
>>rz=real(z);iz=imag(z):%will give rz=3, and iz=4;
>>az=angle(z); abz=abs(z);%will give az=0.9273 rad, and abz=5;
>>x=exp(-z)+4;%x=3.9675+j0.0377;
```

- **Matrix generation functions**

>>[ ]	Empty matrix	>>zeros(3)	3×3 matrix of zeros
>>ones(2)	2×2 matrix of ones	>>eye(3)	identity matrix of size 3
>>rand(3)	3×3 matrix of random numbers		

✡ *Produce a periodic function*

```
>>x=[1 2 3 4];
>>xm=x'*ones(1,5);%xm is 4×5 matrix and each of its column is x';
>>xp=xm(:)';%xp is a row vector, xp=[x x x x x];
```

✡ *Extracting and inserting numbers in arrays*

```
>>x=2:1:6;
>>y=[x zeros(1,3)];%y is an array of the numbers {2, 3, 4, 5,
                    %6, 0, 0, 0};
>>z=y(1,3:7); %1 stands for row 1 which y is and 3:7 instructs
                    %to keep columns
                    %3 through 7 the result is the array {4, 5, 6,
                    %0, 0};
lx=length(x);%lx is the number equal to the number of columns
                    %of the row
                    %vector x, that is lx=5;
x(1,2:4)=4.5*(1:3);%this assignment substitutes the elements
                    %of x at column
                    %positions 2,3 and 4 with the numbers 4.5*
                    %[1 2 3]=4.5, 9,
                    %and 13.5, note the columns of 2:4 and
                    %1:3 are the same;
x(1,2:2:length(x))=pi;%substitutes the columns 2 and 4 of x with
                    %the value of pi, hence the array is
                    %{2, 3.1416, 4, 3.1416, 6};
```

- **Practice the following examples:**

Given Matrix A = [1 2 9; 4 4 6;7 8 7] B = [3 6 9;2 4 6;1 2 3],

Accessing sub-matrices >>A(1:2,2:3); >>A(1,:); >>B(:,2)

Concatenating two matrices >>D=[A B]; >>E=[A;B]

Adding a row >>A(4,:)=[1 1 0]

>>B(:,2)=[ ] Deleting a column
>>A(3) accesses the third element of A.
>>A([1 2 3]) is the first three elements of A.
>>A([sqrt(2), sqrt(3), 4*atan(1)]). ???
If A has N components, >>A(N:-1:1) reverses them.
>>A([1,5],:) = A([5,1],:) interchanges rows 1 and 5 of A.

✓ MATRIX Manipulations / Functions /operations

- Given Matrix A (square)

A' -----> Transpose of a matrix
inv(A) -----> inverse of a matrix
det(A) -----> determinant of a matrix
A*A -----> matrix multiplication
diag(A) -----> diagonal of a matrix
triu(A) , tril(A) , lu(A), eig(A), eigs(A), etc.

Given an array A=[1 5 3 10 2 0 4 6], sort in ascending order. >>sort(A)

✓ OPERATORS

✓ Arithmetic operators

+ Plus.	- Minus.
* Matrix multiplication.	.* Array multiplication
^ Matrix power	.^ Array power
\ Backslash or left division	/ Slash or right division
./ Array division	

- **Array operations [Element by Element operation] [dot operators]**

Matrix operations preceded by a period "**.**" **(dot)** indicates array operation.
Element by element operations require vectors to have the same dimensions.
Eg. ------> **.*** **./** **.^**
A = [1 2 3]; B = [2 4 6];
C = A .* B = [1*2, 2*4, 3*6] ; C = A ./ B = [1/2, 2/4, 3/6] ;
C = A.^2 = [1, 4, 9];

- **Relational operators**

<	less than	<=	Less than or equal to
>	greater than	>=	Greater than or equal to
==	equal	~=	Not equal

- **Logical operators**

& ---> AND | ---> OR ~ ---> NOT xor ---> Exclusive OR

- **Simple math functions**

sin, cos, tan, asin, acos, atan, sinh, cosh log, log10, exp, sqrt …

- **Range of numbers used in MATLAB:** -10^{-308} TO 10^{308}

- **Permanent variables**

ans	Recent computed value	Default result variable
eps	2.22×10^{-16}	User definable
pi	π	Pre-calculated

INF	1/0		Default result variable/User definable
NAN	Not A Number	Eg: 0/0; inf/inf	User definable

- **Special characters**

( ) Parentheses are used to indicate precedence in arithmetic expressions and to arguments of functions in the usual way. Also, to accommodate arguments.

[ ] Brackets used to assign arrays/matrices

,	Element separator	;	Row, line delimiter
%	Comments	!	Exclamation point
'	Transpose, quote	=	Assignment
:	Colon ---> represents range	...	Continuation

- **Logical functions...**

exist	Check if variables or functions exist.
any	True if any element of a vector is true.
all	True if all elements of a vector are true.
find	Find indices of the non-zero elements.
isnan	True for Not-A-Number.
isinf	True for infinite elements.
finite	True for finite elements.
isempty	True for empty matrices.
isreal	True for matrix of only real elements.
issparse	True if matrix is sparse.
isstr	True for text string.
isglobal	True for global variables

- **display formats...**

>>format	Default. Same as SHORT	
>>format short	Scaled fixed-point format with 5 digits.	Eg. 1.3333
>>format long	Scaled fixed-point format with 15 digits.	Eg. 1.33333333333333

✓ Graphics

plot(x,y) Linear plotting -----> plots vector y versus vector x

plot(Y) Plots the columns of Y versus their index

plot(X1,Y1,S1,X2,Y2,S2,X3,Y3,S3,...) combines the plots

fplot('function', limits, linespecifiers) ----> plots a function with y = f(x)

Eg: x = *linspace* (a ,2*pi, 30); y = sin(x); *plot*(x,y)

- Include following modifications on the above plots

title text grid xlabel gtext legend ylabel ginput

- Customizing plot axes using axis command

axis([xmin xmax ymin ymax])

Set the maximum and minimum values of axes using specified values.

To auto scale: use min or max –inf or inf respectively.

- H = *subplot*(m,n,p), or *subplot*(m,n,p), breaks the figure window into an m×n figure windows of small axes, selects the p^{th} axes for the current plot.
- *Hold* on is used to have multiple figures in the same window
- Plotting with Logarithmic axis : *loglog* , *semilogx*(x,y), *semilogy*(x,y)
- Plotting with special graphics : *bar*, *barh*, *stairs*, *stem*, *hist*, *polar* etc.

✓ Creating m-function files – user defined functions :

- Function files are created and edited, like script file, in the Editor/Debugger Window. **Synthetically, a function m-file is identical to a script m-file except for the first line.** The general form of the first line (called "**function definition line**" is:
  function[output1, output2,…] = *filename1(input1, input2,…)*. OR
  *function[*output arguments*]* = *filename2(*input arguments*)*. OR
  function[a, b,…] = *multiply1(x, y,…)*.
- Save the function file with a name that is identical to the function name in the function definition line with extension **.m** E.g. :- *filename1.m, filename1.m, multiply1.m*

Function definition line	Comments
function[mpay,tpay] = loan(amount,rate,years)	Three input arguments, two output arguments.
function [A] = RectArea(a,b)	Two input arguments, one output argument.
function A = RectArea(a,b)	Same as above, one output argument can be typed without the brackets.
function [V, S] = SphereVolArea(r)	One input variable, two output variables.
function trajectory(v,h,g)	Three input arguments, no output arguments.

- Function body contains the computer program (code) that actually performs the computations
- User defined functions are in the same way as a built-in function. To use function file, the directory where it was saved must be in the current directory.
- In function body help comments can also be added for future references, in the **second line** of the function file.
- All the variables in the function file are local. To assign and use global variables, *global* MATLAB command can be used.

✓ **Flow control** (Loops) (Conditional Jumps)

CE -----> Conditional Expression

1. FOR loop
```
for k = F:I:L
  ….
end
```

2. IF statement
```
if  CE1
  ….
else if CE2
  ….
```

3. WHILE loop
```
while CE
  ….
end
```

4. Switch SE
```
case  ….
case
…..
end
```

```
    Else
    ….
   end
end
```

- Nested loops and Nested conditional statements are allowed.
- Indefinite loop may occur if there is a mistake in the usage of loops. In such cases Control+C OR Control +Break key may be used.
- *return* and *continue* commands:
  - when inside loop (*for* and *while*), *return* command terminates the execution of the whole loop.
  - when outside a loop (*for* and *while*), *return* command terminates the execution of the file.
  - *continue* command can be used inside a loop to stop the present pass and start the next pass in the looping process. It omits the remaining commands in the loop.

✓ Polynomials

- Polynomial is represented by a row vector of its coefficients in descending order
  $x^4 - 12x^3 + 25x + 116$ as p = [1 –12 0 25 116]
- **Roots of the polynomial** -----> r =*roots*(p)
- The polynomial can be got from the roots as p = *poly*(r)
- **Value of a Polynomial :** *polyval(*p,x) *x* : number, or a variable that has an assigned value can be array or matrix
- **Multiplication of polynomials:** c = *conv*(p,q)
- **Division of polynomials:** c = *deconv*(p,q)
- **Addition :** two polynomials can be added (or subtracted) by adding the vectors of the coefficients. If the polynomials are not of the same order, the shorter vector has to be modified by adding zeros(called padding)
- Partial fraction expansion : the function *residue* performs a partial fraction expansion F(s) = p / q
  [rr , pp , k] = residue (p, q)
  $F(s) = k + r_1 / (x - p_1) + r_2 / (x - p_2) + \ldots$
- **Derivatives of the polynomials:** pd = *polyder*(p)
- **Derivatives of a product of two polynomials:** pdp= *polyder*(p,q)
- **Derivatives of a quotient of two polynomials:** [k,n] = *polyder*(p,q)

MATLAB Basics Exercises

✓ *MATLAB as a calculator:*

1) Define the variables x and z as x = 9.6, and z = 8.1, then evaluate:

a) $xz^2 - \left(\frac{2z}{3x}\right)^{\frac{3}{5}}$ **b)** $\frac{433z}{2x^3} + \frac{e^{-xz}}{x+z}$ **c)** $\frac{\sqrt{14x^3}}{e^{3x}}$ **d)** $log|x^2 - x^3|$

2) Verify the trigonometric identities by substituting and calculating each side of the equation with x = π/5.

$cos^2\frac{x}{2} = \frac{tanx+sinx}{2tanx}$; $cos\frac{x}{2} = \sqrt{\frac{1+cosx}{2}}$; $tan2x = \frac{2tanx}{1-tan^2x}$

✓ <u>Entering array / Matrices:</u>

3) a) , b) and c)

Create a row vector in which the first element is 1, the last element is 33, with an increment of 2 between the elements (1, 3, 5, ..., 33).

Create a column vector in which the first element is 15, the elements decrease with increments of –5, and the last element is –25. (A column vector can be created by the transpose of a row vector).

Create a row vector with 15 equally spaced elements in which the first element is 7 and the last element is 40.

4)

Create the following matrix C: $C = \begin{bmatrix} 2 & 4 & 6 & 8 & 10 \\ 3 & 6 & 9 & 12 & 15 \\ 7 & 14 & 21 & 28 & 35 \end{bmatrix}$

Use the matrix C to:

a) Create a three-element column vector named `ua` that contains the elements of the third column of C.
b) Create a five-element column vector named `ub` that contains the elements of the second row of C.
c) Create a nine-element column vector named `uc` that contains the elements of the first, third and fifth columns of C.
d) Create a ten-element column vector named `ud` that contains the elements of the first and second rows of C.

5) Using the zeros and ones commands create a 3×5 matrix in which the first, second and fifth columns are 0's and the third and fourth columns are 1's.

✓ Matrix /subscription/concatenation:

6) Create a 5×7 matrix in which the first row are the numbers 1 2 3 4 5 6 7, the second row are the numbers 8 9 10 11 12 13 14, and so on. From this matrix create a new 3×4 matrix that is made from rows 2 through 4, and columns 3 through 6 of the first matrix.

7) Construct a matrix of A of size 39 × 75 whose elements vary from 1 to 10 randomly

a) **Write** a single line code/command in MATLAB to plot elements of 5th row vs. elements of 32nd row using matrix A.

b) **Write** a single line code/command in MATLAB to plot elements of 15th row vs. elements of 4th and 6th columns (suitably choosing the number of elements), using matrix A.

c) **Write** a single line code/command in MATLAB to construct matrix B whose size is 4 × 26 considering the elements of 1st row (75 elements) and the elements of 66th column (39 elements) of A matrix.

d) **Write** a single line code/ command in MATLAB to construct matrix B whose size is 25 × 29 considering the elements of all columns corresponding to 8th row to 14th row (suitable elements) and elements of all rows corresponding to 46th column to 51th column (suitable elements) of A matrix.

8) Construct a matrix of A of size 5 × 6 whose elements vary from 0 to 10.

Construct a matrix of B of size 2 × 45 whose elements vary from 0 to 10.

Write a single line code/ command in MATLAB to construct matrix C whose size is 8 × 15 considering all the elements of matrices A and B (i.e., using matrices A and B).

- ✓ **TRY TO CREATE m - SCRIPT FILES IN SOLVING THE BELOW EXERCISES**
- ✓ **Matrix Manipulations / Functions /operations**

9)

The depth of a well, d, in meters can be determined from the time it takes for a stone that is dropped into the well (zero initial velocity) to hit the bottom by: $d = \frac{1}{2}gt^2$, where t is the time in seconds and $g = 9.81$ m/s^2.

Determine d for $t = 1, 2, 3, 4, 5, 6, 7, 8, 9$, and 10 s.
(Create a vector t and determine d using element-by-element calculation.)

10) Define x and y as the vectors $x = 2, 4, 6, 8, 10$ and $y = 3, 6, 9, 12, 15$. Then use them in the following expression to calculate z using element-by-element calculations.

$$z = \frac{xy + \frac{y}{x}}{(x+y)^{(y-x)}} + 12^{x/y}$$

11)

Show that $\lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n = e$

Do this by first creating a vector n that has the elements: 1 10 100 500 1000 2000 4000 and 8000. Then, create a new vector y in which each element is determined from the elements of n by $\left(1+\frac{1}{n}\right)^n$.

Compare the elements of y with the value of e (type exp(1) to obtain the value of e).

- **Graphics (Plotting)**

12) Make two separate plots of the function f(x) = $0.6x^5-5x^3+9x+2$ in the same window; one plot for $-4\leq x\leq 4$ and one for $-2.7\leq x\leq 2.7$. Also try sketching the same using *fplot*.

13)

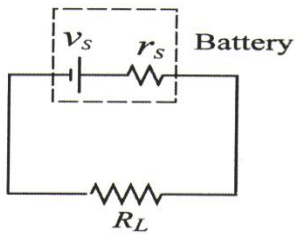
Plot the function $f(x) = \frac{1.5x}{x-4}$ for $-10 \le x \le 10$. Notice that the function has a vertical asymptote at $x = 4$. Plot the function by creating two vectors for the domain of x. The first vector (call it $x1$) with elements from -10 to 3.7, and the second vector (call it $x2$) with elements from 4.3 to 10. For each of the x vector create a y vector (call them $y1$ and $y2$) with the corresponding values of y according to the function. To plot the function make two curves in the same plot ($y1$ vs. $x1$, and $y2$ vs. $x2$).

14) Plot the function $f(x) = 3x\sin(x) - 2x$ and its derivative, both on the same plot, for $-2\pi \le x \le 2\pi$. Plot the function with a solid line, and the derivative with a dashed line. Add a legend and label the axes.

15) An electrical circuit that includes a voltage source v_S with an internal resistance r_S, and a load resistance R_L is shown in the figure. The power P dissipated in the load is given by:

$$P = \frac{v_S^2 R_L}{(R_L + r_S)^2}$$

Plot the power P as a function of R_L for $1 \le R_L \le 10\ \Omega$, given that $v_S = 12$ V, and $r_S = 2.5\ \Omega$.



16) Sketch a sine wave of frequency 1 kHz.

17) Find the error in the following MATLAB codes and mention it clearly,

a) exp(j*w)/(exp(j*w)-0.0591)).*(0.7071^5*exp(-j*w*5)-1)./(0.7071*exp(-j*w)-1))

b) x = 1:100; y = sin(x)*x OR y = sin(x)/x.