Introduction to MATLAB

Outline

- Brief history
- Creating vectors and matrices
- Array Indexing
- Generating sequence of numbers
- Plots (2D and 3D)
- Loops and conditional statements
- Reading external files (.csv, .txt, .wav, etc.)
- Writing custom functions
- Mathematics
- Some alternative languages
- References

(**Note**: This tutorial is just to introduce you to MATLAB. We have not tried to be exhaustive in our presentation. We have aimed at breadth of presentation rather than depth. So there are many things that we will not tell you in this short period of time. We hope, interested readers can explore things themselves after this very short introduction.)

Brief history:

The word "MATLAB" is derived from words "**MAT**rix" and "**LAB**oratory". Everything in MATLAB is a matrix. It was originally designed by Cleve Moler to demonstrate numerical computations in his class. The first codes of MATLAB were written in 'Fortran'. Its popularity grew gradually and it has become an industry now. MATLAB has evolved over the years. Many new functionalities have been added - and are being continually added- to it in terms of toolboxes. Each toolbox performs a specific set of tasks. Availability of toolboxes for a diverse set of tasks makes MATLAB really powerful at handling a wide range of engineering problems. It should be remembered that MATLAB is a commercial entity. Many institutions across the globe have bought subscriptions of it and provide their students with a licensed copy of MATLAB. You can also get your licensed copy form CIC, IIT Kharagpur.

- Desktop basics

MATLAB gets updated twice a year. The latest version is R2019a. Don't worry if you don't have this version installed. For the most part, any other version would also work. To check the version and see a list of available packages run the following command by removing the '%' sign. (% sign is used in MATLAB for commenting. Anything that comes after a % sign is ignored.)

```
% ver
```

To get help on any topic, either Google it or use the help command of MATLAB. For example, to get help on 'plot' command type "help plot".

Creating vectors and matrices:

```
C = [1 2 3 4; 5 6 7 8]
```

```
C = 2×4
```

```
     1     2     3     4
     5     6     7     8
```

```
a = [3; 4; 5]
```

```
a = 3×1
     3
     4
     5
```

```
% Some common functions to create matrices
% eye, zeros, ones, diag, rand, randn etc.
% size, length, reshape
```

- Broadcasting

```
[1,2,3] + 1
```

```
ans = 1×3
     2     3     4
```

```
% [1, 2, 3] + [4, 5]
% [1,2,3]*4
```

Array indexing:

```
B = rand(5);
B(2:4, 1:3)
```

```
ans = 3×3
    0.9816    0.1909    0.2262
    0.1564    0.4283    0.3846
    0.8555    0.4820    0.5830
```

- Array multiplication
- Workspace variables

```
% whos
% clear all
% clc
```

Sequence of numbers:

A sequence of equally spaced numbers can be created in MATLAB in several ways.

```
% linspace(initial_point, final_point, number_of_points)
seq = linspace(0,1,10)
```

```
seq = 1×10
         0    0.1111    0.2222    0.3333    0.4444    0.5556    0.6667    0.7778 ⋯
```

```
seq2 = 0:1/10:1;
```

- Some common functions

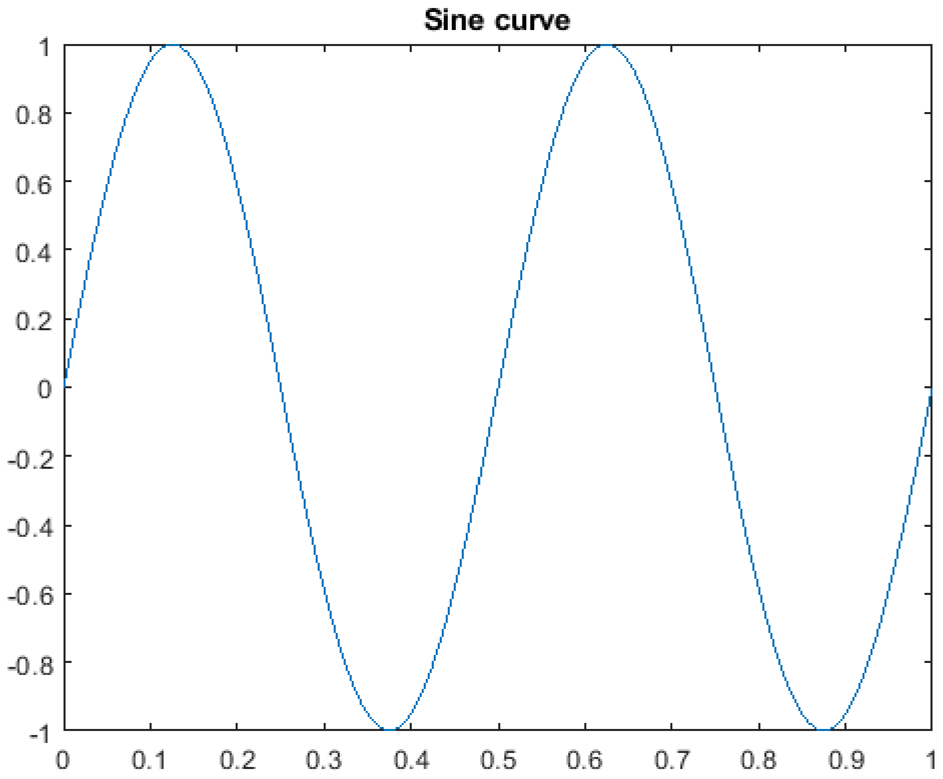
```
% max, min, mean, std, etc.
[val, loc] = max([2 3 28 3 8 5 7])
```

```
val = 28
loc = 3
```

```
% max([2, 3, 4], [1, 7, 2])
```

Plotting:

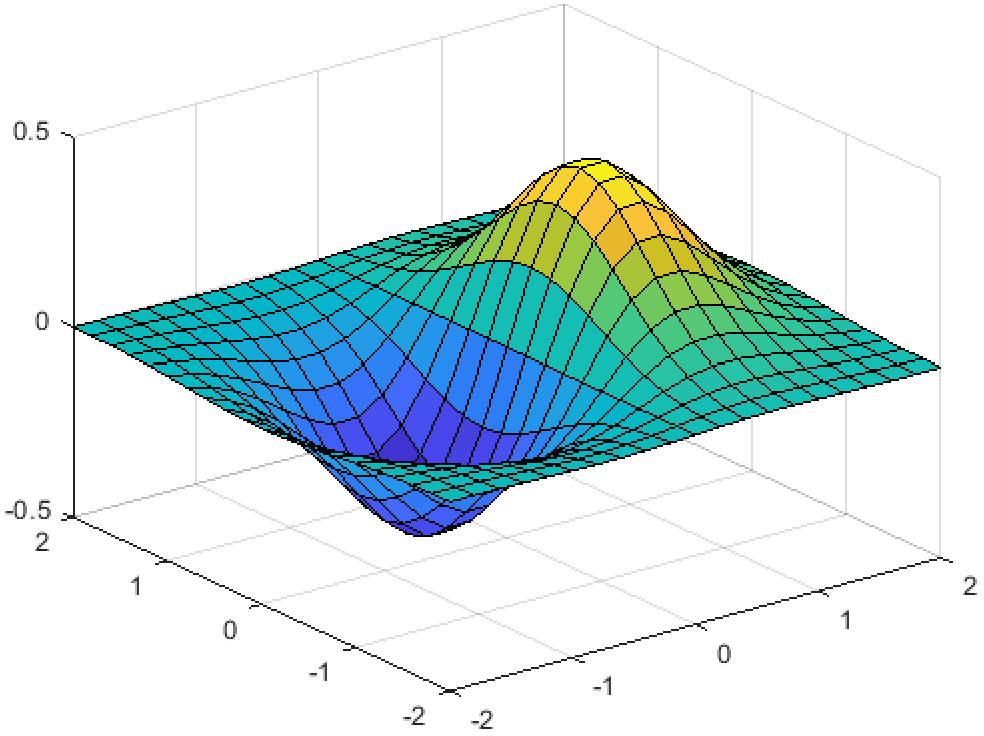
```
t = linspace(0, 1, 100);
y = sin(2*pi*2*t);
figure; plot(t,y); title("Sine curve")
```



```
% help plot       % type this command for further help
% hold on         % Used to show two or more plots in same figure.
% subplot         % Make a grid of plots.
```

For 3D plot we have to first generate a grid of points and then plot the function value over the grid of points. A grid of points can be created in MATLAB using 'meshgrid' command. Then 3D plots can be plotted using 'plot3', 'surf', 'mesh' command.

```
[X,Y] = meshgrid(-2:.2:2);
Z = X .* exp(-X.^2 - Y.^2);
figure; surf(X,Y,Z)
```



Interactively enter inputs:

```
% num = input("Enter a number: ") % Number provided by user is assigned to num
```

Loops and Conditional Statements:

Conditional statements first check the condition and then execute relevant portion of code. "if-else" and "switch-case" statements are conditional statements.

```
if (condition)
    statements
elseif (condition)
    statements
else
    statements
end
```

Every conditional statement or loop must end with "end" keyword.

```
% n = input("Enter a numer: ")
% if (n<10)
%     disp("Number is less than 10")
% elseif (n == 10)
%     disp("Number is equal to 10")
% else
%     disp("Number is greater than 10")
% end
```

Exercise:

- Implement a "switch-case" statement.

```
% for loop
for i = 1:5
    disp("MATLAB is great!")
end
```

```
MATLAB is great!
MATLAB is great!
MATLAB is great!
MATLAB is great!
MATLAB is great!
```

```
% While loop
sum = 0;
while sum <= 5
    sum = sum + 1;
end
sum
```

```
sum = 6
```

Data Import:

MATLAB can read external ".csv" files, ".txt" files, ".wav" files and many others. Explore following commands.

```
% csvread, csvwrite, audioread, imread, readtable etc.
```

Functions:

MATLAB has some built-in functions like- min, max, sort, fft, plot, etc. But sometimes we need other functions to perform specific tasks. For that purpose we can make our custom functions. These custom functions can be used directly or can be called inside another function.

```
% [outputs] = function_name[input_arguments]
%      function body
% end
```

We will write a simple function to find surface area and volume of rectangular parallelopiped, given its length, breadth, and height. Note that while saving a function, file name should be same as function name.

```
% function [surf_area, vol] = our_function(length, breadth, height)
% surf_area = 2*(length*breadth + breadth*height + height*length);
% vol = length*breadth*height;
% end
```

Now write a script and call your function from within that script.

```
% length = input('Enter length: ');
% breadth = input('Enter breadth: ');
% height = input('Enter height: ');
% [area, volume] = our_function(length, breadth, height)
```

Mathematics:

Many mathematical tasks such as linear algebra, optimization, interpolation, solution of differential equations, network analysis, signal processing, etc., can be done in MATLAB. We will only discuss two applications briefly.

- Linear Algebra (Solving linear system of equations)
- Solving ordinary differential equations

Linear Algebra:

Solving a set of linear equations is a breeze in MATLAB. In general, the system can be written as

$$5x_1 + 2x_2 + 3x_3 = 9$$

$$-3x_1 + 4x_2 + 7x_3 = -5$$

$$2x_1 + 3x_2 + 5x_3 = 3$$

This system can be written compactly as

$$\begin{bmatrix} 5 & 2 & 3 \\ -3 & 4 & 7 \\ 2 & 3 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 9 \\ -5 \\ 3 \end{bmatrix}$$

Using more compact notation, it can be written as

$$Ax = b$$

In MATLAB it can be solved as follows.

```
A = [5 2 3; -3 4 7; 2 3 5];
b = [9; -5; 3];
solution = A\b
```

```
solution = 3×1
    1.0000
   17.0000
  -10.0000
```

Caution: In the above, we have used "back slash" operator to solve the system. But "back slash" means different things in different contexts in MATLAB. So use it with care.

Another (**not recommended**) way of solving the system is

```
not_recommended_solution = inv(A)*b
```

```
not_recommended_solution = 3×1
     1.0000
    17.0000
   -10.0000
```

Some other things:

```
rank(A) % Finds rank
```

```
ans = 3
```

```
rref(A) % Finds reduced row echelon form. Read about it to find out more.
```

```
ans = 3×3
     1     0     0
     0     1     0
     0     0     1
```

```
rref([A,b]) % This also gives the solution
```

```
ans = 3×4
     1     0     0     1
     0     1     0    17
     0     0     1   -10
```

Finding eigenvalues and eigenvectors:

Finding eigenvalues and eigenvectors of a matrix is a recurrent problem in many fields of engineering. In MATLAB it can be obtained using the command "eig". Many different things happen under the hood when you call "eig". But we are not bothered about that right now. If you are interested in knowing how exactly eigenvalues are calculated, read "Numerical Linear Algebra" books.

```
[vec, vals] = eig(A)
```

```
vec = 3×3
   -0.0422    0.7834    0.6436
    0.8621   -0.5534    0.4524
   -0.5050    0.2828    0.6173
vals = 3×3
    0.0461         0         0
         0    4.6701         0
         0         0    9.2837
```

Some advanced stuff:

Read more about the following commands. Type "help command_name".

```
% svd, qr, lu, chol,etc.
```

Solving differential equations:

MATLAB solves ODE's of the form (we assume implicitly that independent variable is time)

$$\dot{x}(t) = f(t, x)$$

This is a first order differential equation. In practice, we mostly encounter second order differential equations in vibration. By a clever change of variable we can convert a second order differential equation into a system of first order differential equations. The required steps are as follows:

Let's suppose, we are interested in solving the second order equation

$$m\ddot{x} + c\dot{x} + kx = 0$$

Take, $x_1 = x$, and $x_2 = \dot{x}$. So

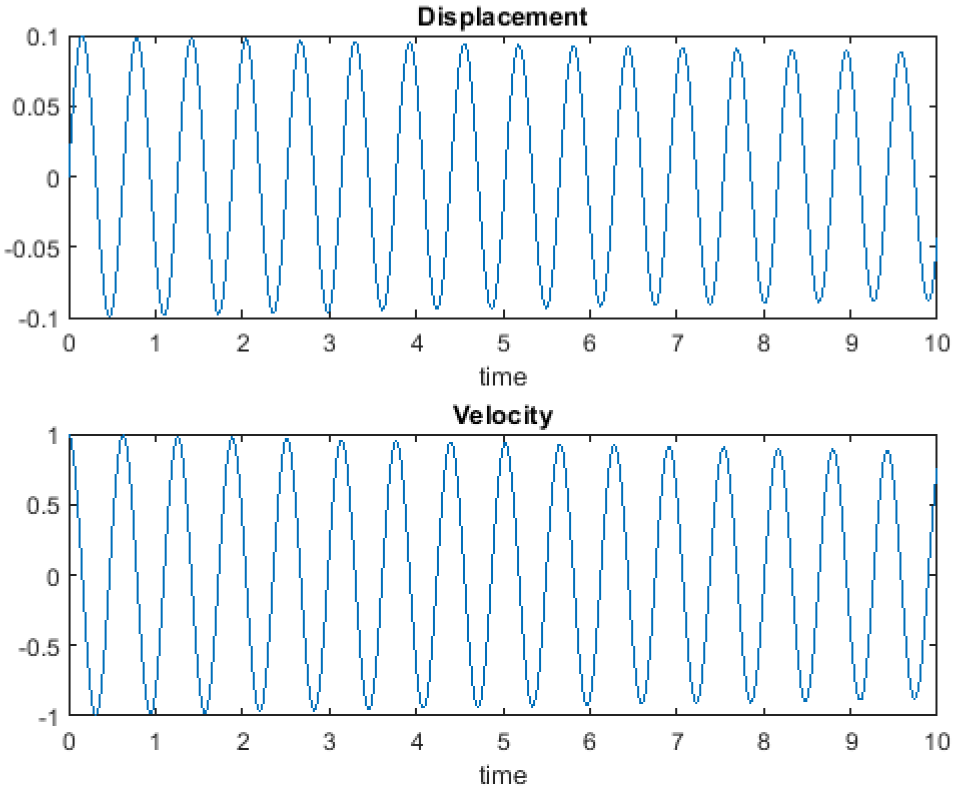
$$\dot{x}_1 = \dot{x} = x_2 \text{ and } \dot{x}_2 = \ddot{x} = -\frac{k}{m}x_1 - \frac{c}{m}x_2$$

So the above second degree equation can be written as a system of first order equations as below.

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Now this system can be solved easily in MATLAB. We will write an anonymous function that represents the RHS of the above equation. Then 'ode45' solves the differential equation. 'ode45' function takes 3 inputs - the function to be solved, the time interval between which solution is required, and initial condition. Output of this function is a set of time values and corresponding function values. This is how we represent it in MATLAB.

```
m = 2; k = 200; c = 0.05;
anonym_function = @(t,x) [x(2);-(k/m)*x(1) - (c/m)*x(2)];
[time, y_out] = ode45(anonym_function, [0,10], [0;1]);
figure
subplot(2,1,1)
plot(time, y_out(:,1));title("Displacement"); xlabel("time")
subplot(2,1,2)
plot(time, y_out(:,2)); title("Velocity"); xlabel("time")
```



The result is a slowly decaying function. Increase the amount of damping and see what changes.

Exercise:

- Solve a forced second order single degree of freedom spring mass system. Assume any forcing function. For simplicity, you can choose the forcing function to be sinusoidal.
- Solve other variations of this problem. For example, solve a 2 degrees of freedom system or whatever you like.

Alternative Languages:

As MATLAB is a commercial language, getting a licensed copy outside academia might be costly. Many open source alternatives exist that will work equally well. "**Octave**" is a free software that is almost identical to MATLAB. Another alternative is "**R**", the open source statistical computing software. All matrix computations that you can do in MATLAB, can also be done in **R**, though the syntax is different. The same is true for "**Python**".

Another recent language that is faster than MATLAB and with similar syntax is "**Julia**". **Julia** is open source and probably the best for high performance computing. It all boils down to availability and personal choice while choosing a language to work with.

References:

There are many resources (books, videos, documents, etc.) to learn MATLAB. Here we mention only three that are our personal favorites.

- MATLAB getting started guide (https://in.mathworks.com/help/pdf_doc/matlab/getstart.pdf)
- Higham, Desmond J., and Nicholas J. Higham. *MATLAB guide*. Vol. 150. Siam, 2016.

- Attaway, Stormy. *Matlab: a practical introduction to programming and problem solving*. Butterworth-Heinemann, 2013.

Last Modified: 15 August, 2019 (The material in this article was covered in a one hour tutorial class at IIT Kharagpur.)