DIGITAL SIGNAL PROCESSING LAB MANUAL 6

Dr. Ahsan Latif, Ms. Anosh Fatima

WINTER SEMESTER 2022

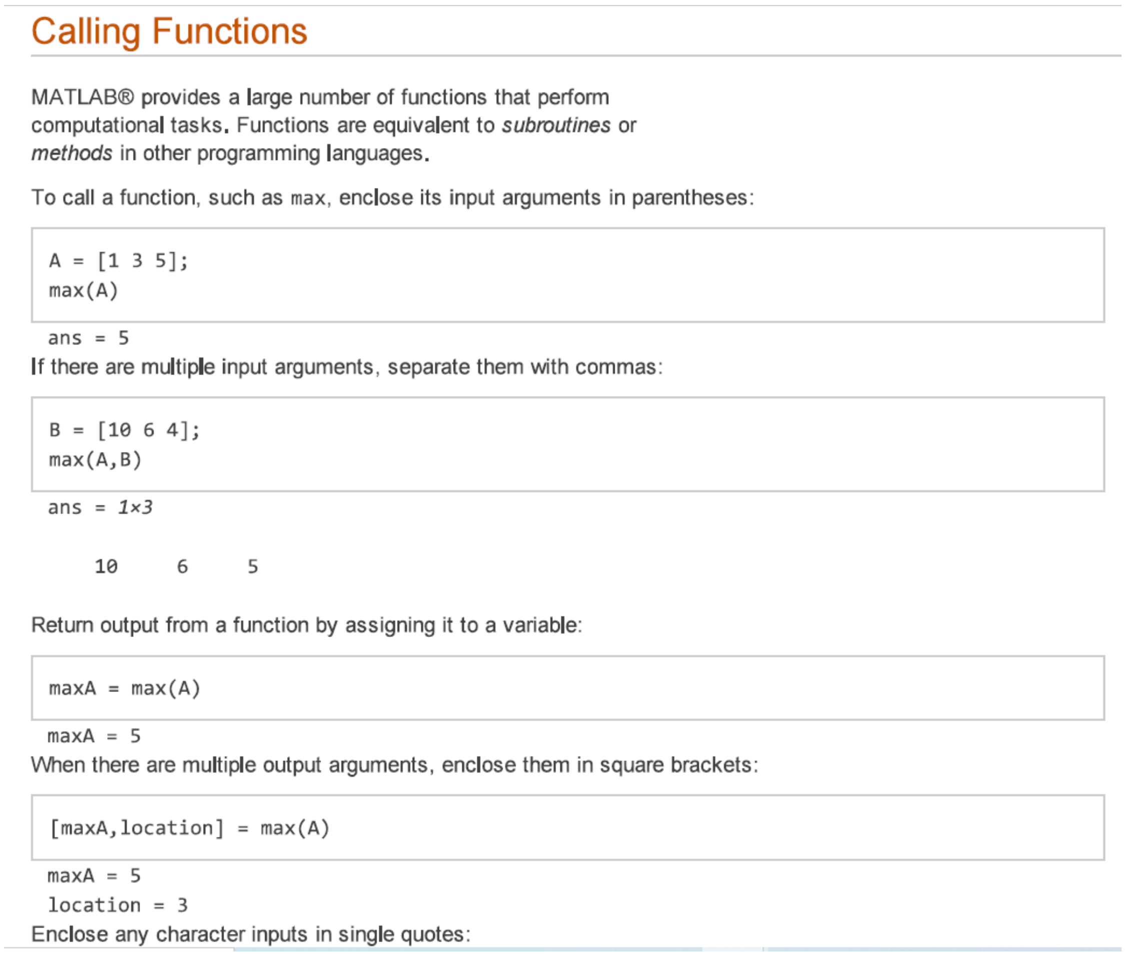
UNIVERSITY OF AGRICULTURE, FAISALABAD (UAF)

MATLAB Fundamentals

Function Calling

TASK

- Open New Script from Toolbar
- Write code (all examples and Exercises) in Editor Window
- Click Run from Tool Bar to see output in command window.
- Save all MATLAB Files in Separate folder called DSP Labs
- Write name of file: your group number and lab number. E.g., G2Lab3
- Click add to the path
- Check results (output)
- Submit MATLAB code to TA online. Solve each exercise in separate script.
- ALL Code must be properly commented for submission, explaining each each step/line of exercises.



Calling Functions

MATLAB® provides a large number of functions that perform computational tasks. Functions are equivalent to *subroutines* or *methods* in other programming languages.

To call a function, such as `max`, enclose its input arguments in parentheses:

```
A = [1 3 5];
max(A)
```

```
ans = 5
```

If there are multiple input arguments, separate them with commas:

```
B = [10 6 4];
max(A,B)
```

```
ans = 1×3

    10     6     5
```

Return output from a function by assigning it to a variable:

```
maxA = max(A)
```

```
maxA = 5
```

When there are multiple output arguments, enclose them in square brackets:

```
[maxA,location] = max(A)
```

```
maxA = 5
location = 3
```

Enclose any character inputs in single quotes:

```
disp('hello world')
```

```
hello world
```

To call a function that does not require any inputs and does not return any outputs, type only the function name:

```
clc
```

The `clc` function clears the Command Window.

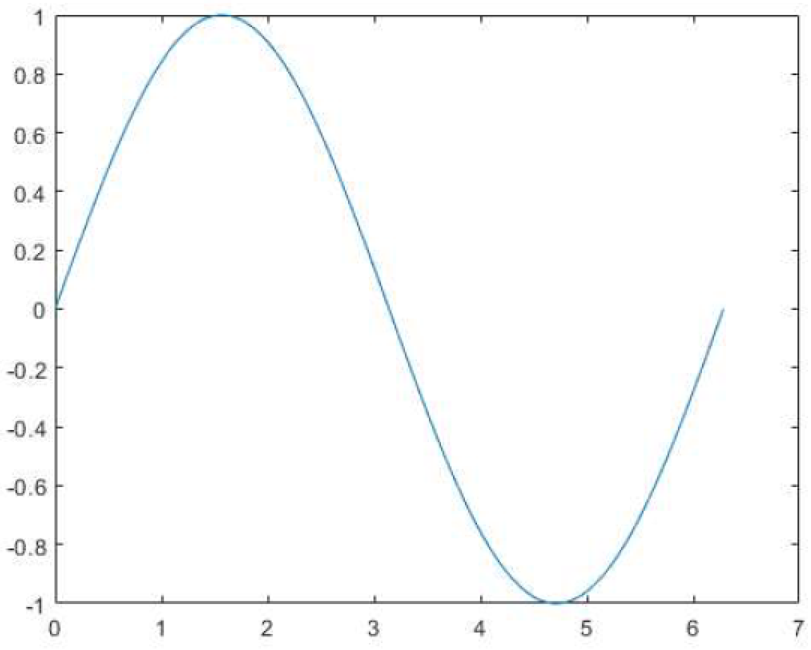
2D and 3D Graphs Plotting

2-D and 3-D Plots

Line Plots

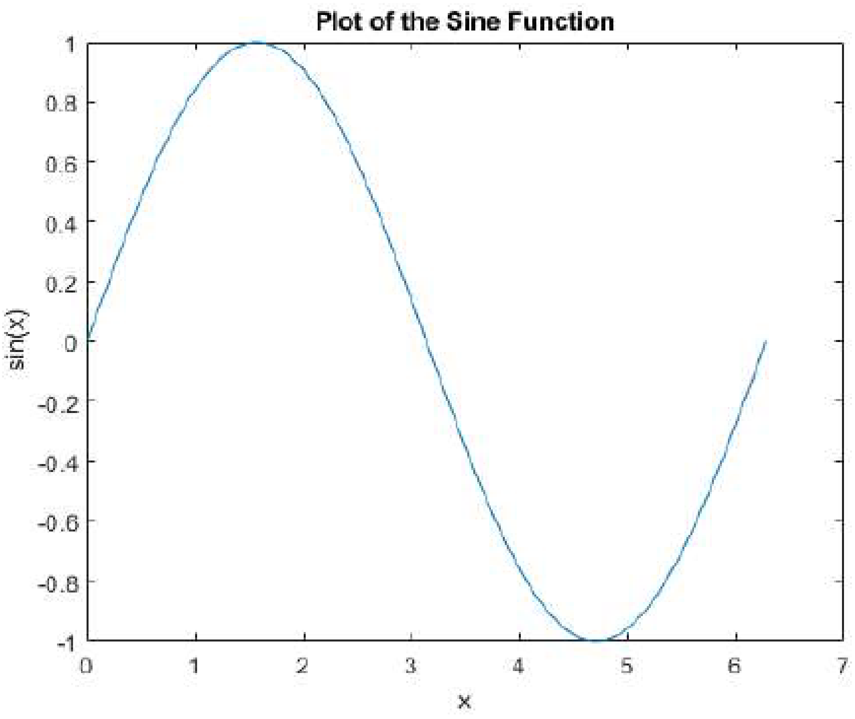
To create two-dimensional line plots, use the `plot` function. For example, plot the value of the sine function from 0 to 2π:

```
x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y)
```



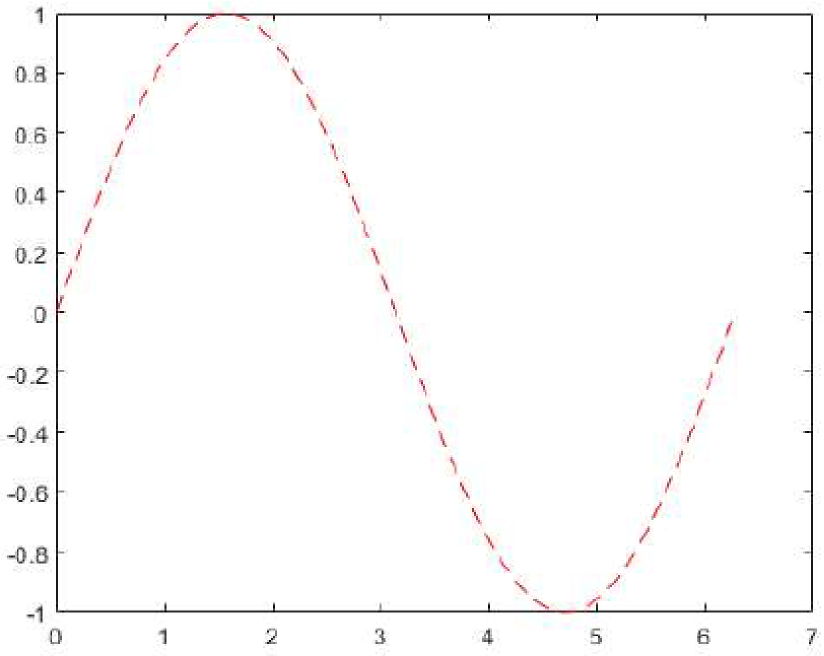
You can label the axes and add a title.

```
xlabel('x')
ylabel('sin(x)')
title('Plot of the Sine Function')
```



By adding a third input argument to the `plot` function, you can plot the same variables using a red dashed line.

```
plot(x,y,'r--')
```



`'r--'` is a *line specification*. Each specification can include characters for the line color, style, and marker. A marker is a symbol that appears at each plotted data point, such as a `+`, `o`, or `*`. For example, `'g:*'` requests a dotted green line with * markers.

Notice that the titles and labels that you defined for the first plot are no longer in the current *figure* window. By default, MATLAB® clears the figure each time you call a plotting function, resetting the axes and other elements to prepare the new plot.

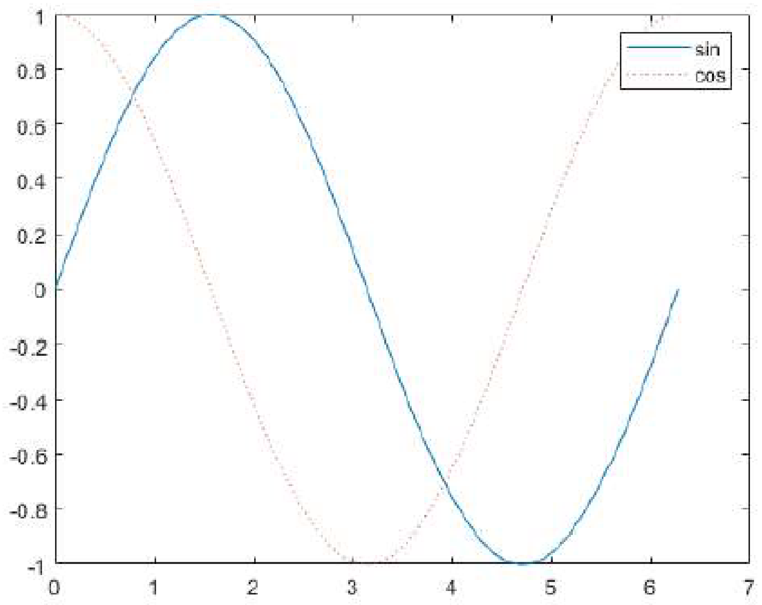
To add plots to an existing figure, use `hold on`. Until you use `hold off` or close the window, all plots appear in the current figure window.

```
x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y)

hold on

y2 = cos(x);
plot(x,y2,':')
legend('sin','cos')

hold off
```



3-D Plots

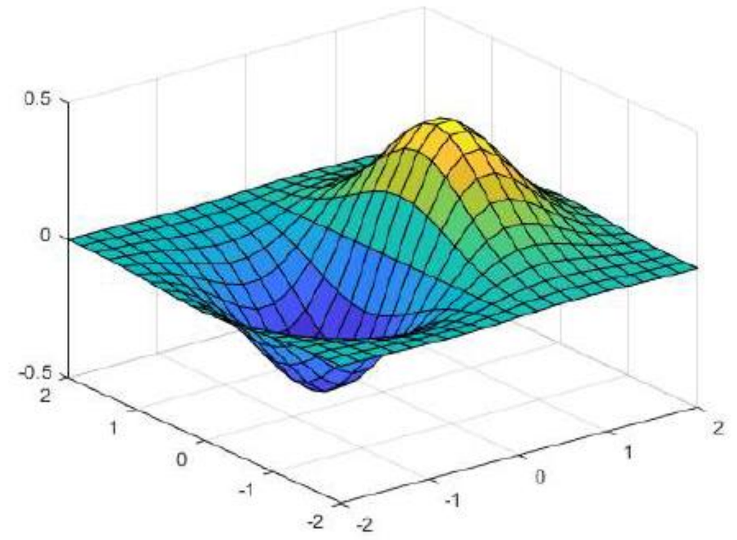
Three-dimensional plots typically display a surface defined by a function in two variables, *z = f(x,y)* .

To evaluate *z*, first create a set of (*x*,*y*) points over the domain of the function using `meshgrid`.

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

Then, create a surface plot.

```
surf(X,Y,Z)
```



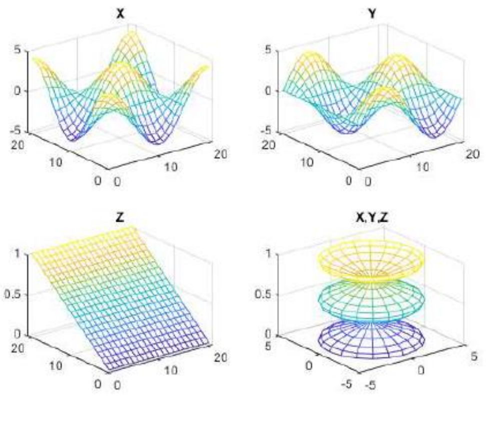
Both the surf function and its companion mesh display surfaces in three dimensions. surf displays both the connecting lines and the faces of the surface in color. mesh produces wireframe surfaces that color only the lines connecting the defining points.

Subplots

You can display multiple plots in different subregions of the same window using the subplot function.

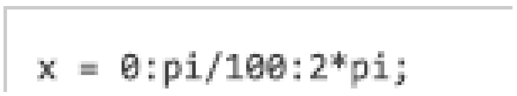
The first two inputs to subplot indicate the number of plots in each row and column. The third input specifies which plot is active. For example, create four plots in a 2-by-2 grid within a figure window.

```
t = 0:pi/10:2*pi;
[X,Y,Z] = cylinder(4*cos(t));
subplot(2,2,1); mesh(X); title('X');
subplot(2,2,2); mesh(Y); title('Y');
subplot(2,2,3); mesh(Z); title('Z');
subplot(2,2,4); mesh(X,Y,Z); title('X,Y,Z');
```



Exercise 1

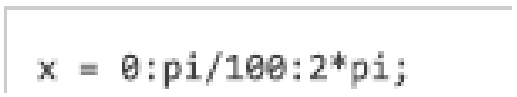
1. Create a matrix A of 5 numbers and display its maximum value using function and store result in variable ‘maximumA’.
2. Create a matrix B of 3 numbers and display its maximum value using function and store result in variable ‘maximumB’.
3. Find out maximum values in both matrices A and B using the max function with multiple input arguments and and store result in variable maximumAB.
4. Create a matrix C of 7 numbers and display its maximum value using max function and display location of maximum number.
5. Create a graph between x and cos(x) when



```
x = 0:pi/100:2*pi;
```

Properly label both axes and display title of graph.

6. Create a graph between x and tan(x) when

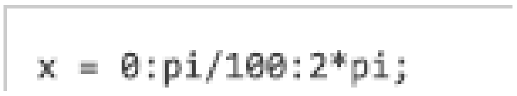


```
x = 0:pi/100:2*pi;
```

Properly label both axes and display title of graph.

Graph line must be dashed and in blue color.

7. Create a graph between x and cos(x) when



```
x = 0:pi/100:2*pi;
```

Properly label both axes and display title of graph.

Graph line must be in form of * and in green color.

8. Create any surface plot and use meshgrid function for it. Also create its 3 subplots. (Read example above for reference)
9. Create a matrix D of 5 numbers and display its minimum value using function and display its location.
10. Use matrix D of 5 numbers and display its mean value using function.
11. Create a matrix D of 5 numbers and display its maximum value using function and display its location.
12. Calculate volume and area of cube and plot it.
(For reference, read sphere example explained below.)

All MATLAB® functions have supporting documentation that includes examples and describes the function inputs, outputs, and calling syntax. There are several ways to access this information from the command line:

- Open the function documentation in a separate window using the `doc` command.

  ```
  doc mean
  ```

- Display function hints (the syntax portion of the function documentation) in the Command Window by pausing after you type the open parentheses for the function input arguments.

  ```
  mean(
  ```

- View an abbreviated text version of the function documentation in the Command Window using the `help` command.

  ```
  help mean
  ```

Scripts

The simplest type of MATLAB® program is called a *script*. A script is a file that contains multiple sequential lines of MATLAB commands and function calls. You can run a script by typing its name at the command line.

Scripts

To create a script, use the `edit` command,

```
edit mysphere
```

This command opens a blank file named `mysphere.m`. Enter some code that creates a unit sphere, doubles the radius, and plots the results:

```
[x,y,z] = sphere;
r = 2;
surf(x*r,y*r,z*r)
axis equal
```

Next, add code that calculates the surface area and volume of a sphere:

```
A = 4*pi*r^2;
V = (4/3)*pi*r^3;
```

Whenever you write code, it is a good practice to add comments that describe the code. Comments enable others to understand your code and can refresh your memory when you return to it later. Add comments using the percent (%) symbol.

```
% Create and plot a sphere with radius r.
[x,y,z] = sphere;       % Create a unit sphere.
r = 2;
surf(x*r,y*r,z*r)       % Adjust each dimension and plot.
axis equal              % Use the same scale for each axis.

% Find the surface area and volume.
A = 4*pi*r^2;
V = (4/3)*pi*r^3;
```

Save the file in the current folder. To run the script, type its name at the command line:

```
mysphere
```

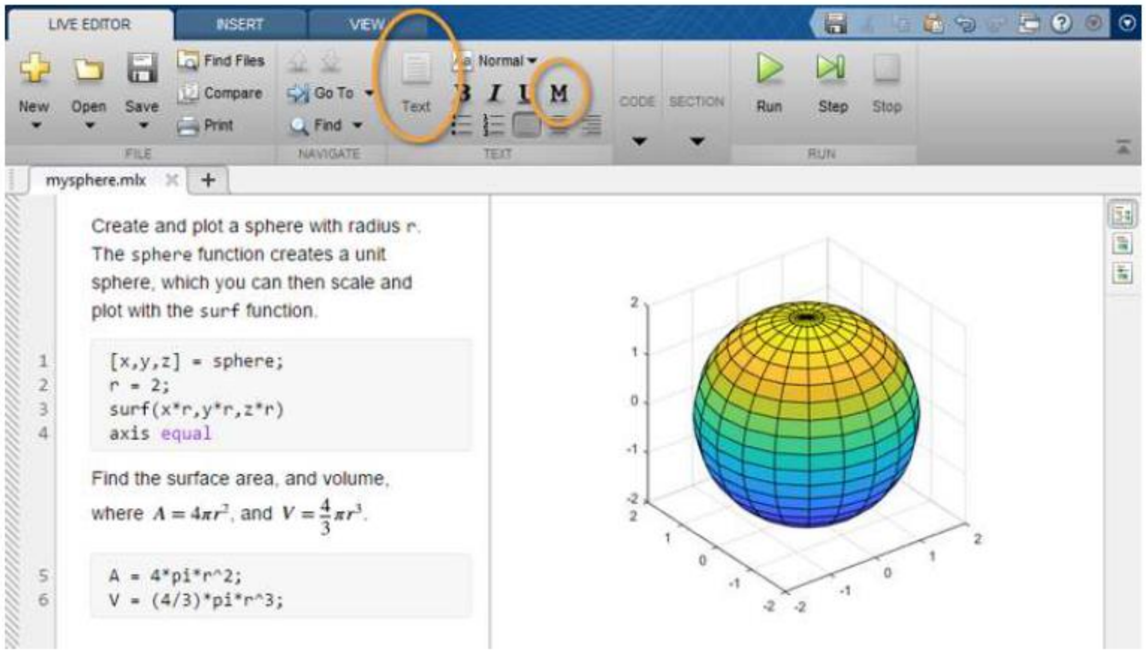
You can also run scripts from the Editor using the **Run** button, ▷.

Live Scripts

Instead of writing code and comments in plain text, you can use formatting options in *live scripts* to enhance your code. Live scripts allow you to view and interact with both code and output and can include formatted text, equations, and images.

For example, convert `mysphere` to a live script by selecting **Save As** and changing the file type to a MATLAB live code file (`*.mlx`). Then, replace the code comments with formatted text. For instance:

- Convert the comment lines to text. Select each line that begins with a percent symbol, and then select **Text**, ▤. Remove the percent symbols.
- Rewrite the text to replace the comments at the end of code lines. To apply a monospace font to function names in the text, select **M**. To add an equation, select **Equation** on the **Insert** tab.



Create and plot a sphere with radius `r`. The `sphere` function creates a unit sphere, which you can then scale and plot with the `surf` function.

```
[x,y,z] = sphere;
r = 2;
surf(x*r,y*r,z*r)
axis equal
```

Find the surface area, and volume, where $A = 4\pi r^2$, and $V = \frac{4}{3}\pi r^3$.

```
A = 4*pi*r^2;
V = (4/3)*pi*r^3;
```

To create a new live script using the `edit` command, include the `.mlx` extension with the file name:

```
edit newfile.mxl
```

Script Locations

MATLAB looks for scripts and other files in certain places. To run a script, the file must be in the current folder or in a folder on the *search path*.

By default, the `MATLAB` folder that the MATLAB Installer creates is on the search path. If you want to store and run programs in another folder, add it to the search path. Select the folder in the Current Folder browser, right-click, and then select **Add to Path**.