



Introduction to MATLAB Environment

Lecture 1

Dr. Görkem Saygılı

Department of Biomedical Engineering
Ankara University

Introduction to MATLAB, 2017-2018 Spring



MATLAB:

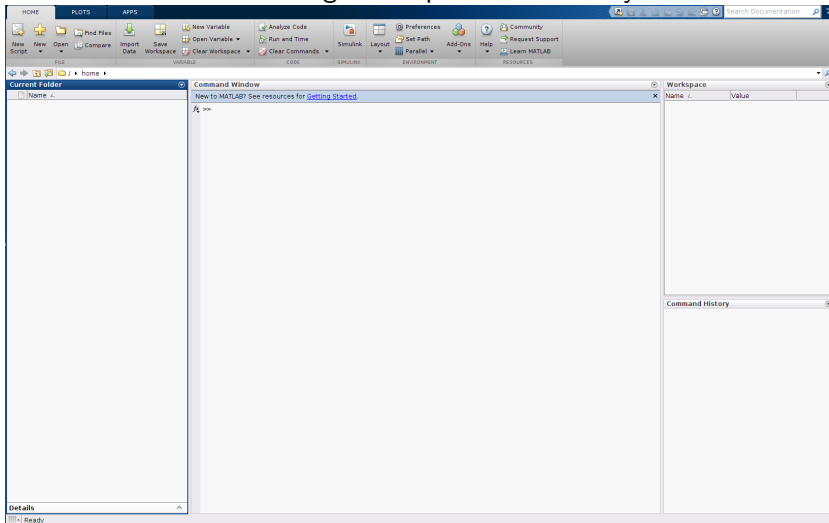
MATLAB is a numerical computing environment and programming language which was invented by Cleve Moler (from University of Mexico) in late 1970s.

MATLAB is now a product of MathWorks (established in 1984). Recently, the company has 100 products and over 1 million users. MATLAB is widely used by universities and companies all over the world.

MATLAB Desktop:



MATLAB has the following desktop environment by default:





MATLAB Desktop Windows:

Default MATLAB desktop has four windows:

Workspace: The variables that are stored in MATLAB environment are displayed in the Workspace window.

Current Folder: Current Folder window shows the recent working directory of MATLAB.

Command Window: Command Window is the input screen for the commands in MATLAB.

Command History: Command History window stores the commands that has been inputted using the Command Window.

MATLAB Desktop Example:



The screenshot displays the MATLAB Desktop interface. The top menu bar includes HOME, PLOTS, and APPS. The ribbon contains various toolbars for file operations, workspace management, code execution, and environment settings. The Command Window shows the following commands and output:

```
>> x = 2+3  
  
x =  
  
5  
  
>> y = 1e3*x  
  
y =  
  
1005  
  
f1 >> |
```

The Workspace window shows the following variables:

Name	Value
x	5
y	1005

The Command History window shows the following commands:

```
x = 2+3  
y = 1e3*x
```



Quit & Abort:

To quit MATLAB, you may either type:

```
>> quit
```

to the command line.

You may also use the combination of “command + q” for Mac or “Cntrl + q” for Windows and Linux.

In some cases, processing a command may take longer than usual (such as in case of an infinite loop). MATLAB aborts the current process when you hit “Ctrl + c” combination.



Issuing Commands:

To issue a command, you need to enter it to the Command Window and hit the Enter key afterwards:

```
>> x = 2+3  
  
x =  
  
    5  
  
>>
```

From now on, MATLAB stores variable x as 5 in its workspace. Whenever you type x in the Command Window, MATLAB will display x with its value as 5.



Variables Not In Workspace:

```
>>> x = 2+3
```

```
x =
```

```
5
```

```
>>> x
```

```
x =
```

```
5
```

```
>>> y
```

```
Undefined function or variable 'y'.
```

```
>>>
```




Supressing Printing (The Use of Semicolon):

You might not always want to display the value of a variable after you assign a value to it. To prevent MATLAB from displaying the value of the variable after the assignment, you can use a semicolon at the end of your assign command before hitting Enter:

```
>> x = 2+3;  
>> x  
  
x =  
  
    5  
  
>> |
```



Lower Case and Upper Case Letters for Variables:

While initializing variables, you need to be careful about the lower case and upper case letters inside your variable. MATLAB is case sensitive!

```
>> x = 2+3;
>> x

x =

    5

>> X
Undefined function or variable 'X'.

Did you mean:
>> x|
```

Making Calculations:



You can do calculations using the Command Window:

```
>> x = 2+3;
```

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

```
y =
```

```
    25
```

```
>> y = pi*x
```

```
y =
```

```
    15.7080
```

```
>> y = 1e3+x
```

```
y =
```

```
    1005
```

```
>> |
```



Variables in the Workspace:

Apart from the information you can obtain from the Workspace Window, you can also use “whos” built-in function of MATLAB to observe the attributes of the variables that are stored in the workspace:

```
>> whos
  Name      Size      Bytes  Class  Attributes
  x         1x1         8  double
  y         1x1         8  double
>>
```



Saving Variables:

When MATLAB is closed, all of the variables in the workspace will be lost. To avoid that and save the variables that will be needed in the future, there is a built-in function in MATLAB: `save()`. This function saves all the variables in the current workspace in a file named “matlab.mat”. You can also save the variables you want specifically as:

```
>> save('x_var', 'x');  
>> save|
```



Loading Variables:

After you saved your variable(s), you may want to load them back to your workspace. There is another built-in MATLAB function, which is named as `load()`:

```
>> load('x_var');  
>> |
```

After hitting Enter, you will see the previously saved variable (which is `x` in this case), in your workspace.



Next Line:

Especially when a command is too long, you might want to continue typing your command from the next line. To do that, you can use three dots, “...” and hit Enter to continue from the next line:

```
>> x = ...  
2+3  
  
x =  
    5  
  
>> |
```



Reusing Previous Comments:

The previously used commands are stored in the Command History window:

```
Command History
save
clc
clear all
clc
- save('x_var', 'x');
clc
load x_var
clc
save('x_var', 'x');
clc
```

You can also use upper arrow key on your keyboard to reuse the previously-typed commands.



Syntax and Semantics:

You should follow the syntactic rules while entering your commands. If you don't, you end up with syntax errors as follows:

```
Command Window
New to MATLAB? See resources for Getting Started.
>> 5 = x
    5 = x
    |
Error: The expression to the left of the equals sign is not a valid target for an assignment.
fx >>
```

Name	Value
x	5
y	1005

“=” is an assignment operator. Hence, constants cannot be on the left side of “=” operator.

Syntax is the form of a statement.

Semantics is the meaning of a statement.

Variable Names:



There are certain rules to be followed while creating variables in terms of their names. The following figure shows some syntax errors while initializing variables:

```
>> 1x = 5
    1x = 5
      ↑
Error: Unexpected MATLAB expression.

>> x?1 = 6
    x?1 = 6
      ↑
Error: Unexpected MATLAB operator.

>> x*7 = 10
    x*7 = 10
      ↑
Error: The expression to the left of the equals sign is not a valid target for an assignment.

>> x_1 = 10

x_1 =

    10

>>
```

MATLAB help:



There is a built-in “help” function in MATLAB to help its users about the details of a command:

```
>> help quit
quit Quit MATLAB session.
quit terminates MATLAB after running the script FINISH.M,
if it exists. The workspace information will not be saved
unless FINISH.M calls SAVE. If an error occurs while
executing FINISH.M, quitting is cancelled.

quit FORCE can be used to bypass an errant FINISH.M that
will not let you quit.

quit CANCEL can be used in FINISH.M to cancel quitting.
It has no effect anywhere else.

Example
Put the following lines of code in your FINISH.M file to
display a dialog that allows you to cancel quitting.

    button = questdlg('Ready to quit?', ...
        'Exit Dialog','Yes','No','No');
    switch button
    case 'Yes',
        disp('Exiting MATLAB');
        %Save variables to matlab.mat
        save
    case 'No',
        quit cancel;
    end

Note: When using Handle Graphics in FINISH.M make sure
to use UIWAIT, WAITFOR, or DRAWNOW so that figures are
visible.

See also exit.

Reference page for quit
```

>>



MATLAB lookfor:

You might not always remember the name of the commands you want to use. MATLAB has a built-in command to find out the command which you could not remember. However, you still need to give MATLAB a hint!

```
>> lookfor whil
break
continue
end
while
dbdown
dbup
rsimdemo1
setenableprop
trueorfalse
linlft
localtonemap
>>
```

- Terminate execution of WHILE or FOR loop.
- Pass control to the next iteration of FOR or WHILE loop.
- Terminate scope of FOR, WHILE, SWITCH, TRY, and IF statements.
- Repeat statements an indefinite number of times.
- Reverse workspace shift performed by DBUP, while in debug mode
- Shift current workspace to workspace of caller, while in debug mode
- Runs ten RSim simulations while altering damping coefficient.
- Sets the enable property of uicontrols while maintaining the
- checks logic value while tolerating numerical values.
- Obtains a linear model from a Simulink model while removing
- Render HDR image for viewing while enhancing local contrast



```
>> lookfor statement
```

```
case
```

```
elseif
```

```
end
```

```
for
```

```
if
```

```
otherwise
```

```
while
```

```
echo
```

```
ext\_open\_intrf
```

```
>> help otherwise
```

```
otherwise Default SWITCH statement case.
```

```
otherwise is part of the SWITCH statement syntax, whose general form is:
```

```
SWITCH switch_expr
  CASE case_expr,
    statement, ..., statement
  CASE {case_expr1, case_expr2, case_expr3,...}
    statement, ..., statement
  ...
  otherwise,
    statement, ..., statement
END
```

The **otherwise** part is executed only if none of the preceding case expressions match the switch expression.

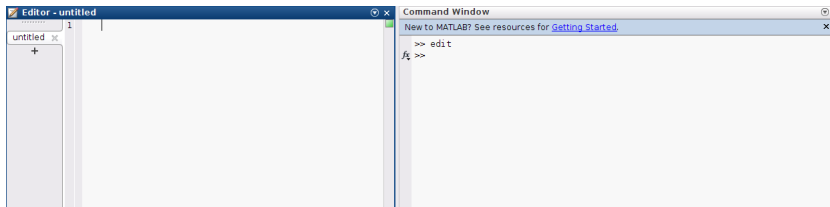
See also [switch](#), [case](#).

[Reference page for otherwise](#)



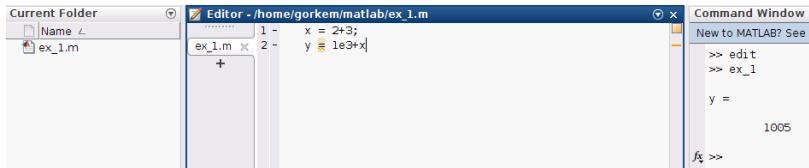
MATLAB editor:

So far, you used the Command Window which is of course behaves like a very advanced calculator, yet you may want to exploit the real power of MATLAB as a programming language. Editor is there to fulfill your needs:





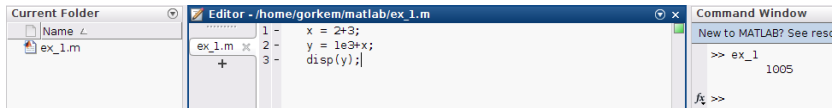
Command Window might be adequate for a couple of commands. But when you need to write bigger codes, you better use MATLAB editor. You can also use editor for simple calculations:



The commands that are typed in the editor is now saved as “ex_1.m” in the current folder of MATLAB.



To display a result you can use the `disp()` function in MATLAB:



The screenshot shows the MATLAB environment. The Editor window displays a script named `ex_1.m` with the following code:

```
1 - x = 2+3;  
2 - y = 1e3+x;  
3 - disp(y);
```

The Command Window shows the execution of the script:

```
New to MATLAB? See res  
>>> ex_1  
      1005  
fx >>
```

```
>> help disp  
disp Display array.  
disp(X) displays array X without printing the array name or  
additional description information such as the size and class name.  
In all other ways it's the same as leaving the semicolon off an  
expression except that nothing is shown for empty arrays.  
  
If X is a string or character array, the text is displayed.  
  
See also fprintf, sprintf, int2str, num2str, rats, format, details.  
  
Reference page for disp  
Other functions named disp
```




MATLAB Path:

To run an m-file, that m-file has to be either in the current folder or in the path of MATLAB:

```
Current Folder: /home/gorkem/matlab/ex1/ex_1.m
Editor: /home/gorkem/matlab/ex1/ex_1.m
1 - x = 2+3;
2 - y = 1e3*x;
3 - disp(y);
Command Window:
New to MATLAB? See resources for Getting Started.
>> ex_1
Undefined function or variable 'ex_1'.
```

```
Current Folder: /home/gorkem/matlab/ex1/ex_1.m
Editor: /home/gorkem/matlab/ex1/ex_1.m
1 - x = 2+3;
2 - y = 1e3*x;
3 - disp(y);
Command Window:
New to MATLAB? See resources for Getting Started.
>> addpath('ex1');
>> ex_1
1005
```

By using `addpath()` function in MATLAB, you can add the folder that contains the targetted script.



Algorithm & Source Code:

Algorithm is a terminology generally used in computer science to describe a step by step procedure to solve a particular problem.

Computer programs are written for executing an algorithm in a computer.

Computer programs are written by means of typing programming language specific commands/text in a file such as a script of MATLAB which is called the source code of the program.



Comments:

You may not always write commands that a computer should interpret and execute in a source code. You may also want to write text inside your code to let other people understand it easily. You can do this by typing “%” first and then typing the explanatory text.

A screenshot of the MATLAB environment. The Editor window on the left shows a script named 'ex_1.m' with four lines of code: a comment, two assignment statements, and a display statement. The Command Window on the right shows the execution of 'ex_1' resulting in the value '1005'.

```
Editor - /home/gorkem/matlab/ex1/ex_1.m
-----
1 - % this is a simple code
2 - x = 2+3;
3 - y = 1e3+x;
4 - disp(y);

Command Window
-----
New to MATLAB? See resources for Getting Started.

>> ex_1
      1005

fx >>
```



Vectors:

So far, we simply assign a value to a variable. This may not be enough when we want to compute over a list of numbers. For this purpose, we can create a vector using square brackets as follows:

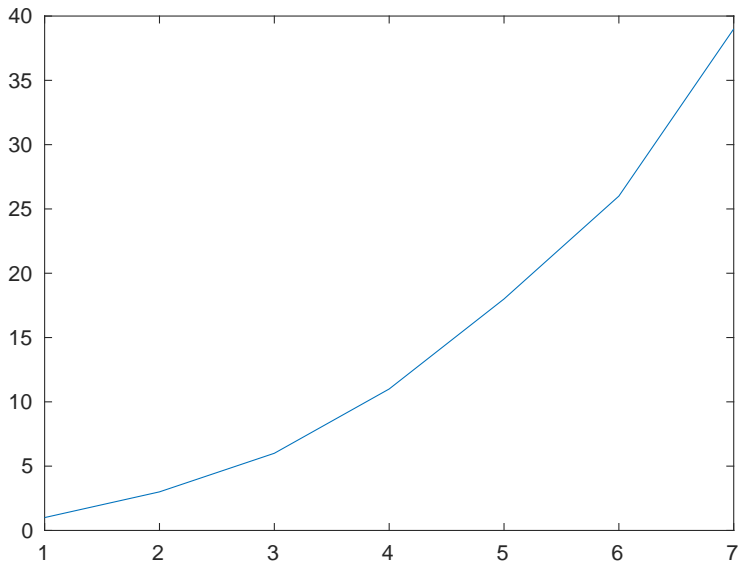
```
>> x = [1 2 3 4 5 6 7]
x =
     1     2     3     4     5     6     7
>> y = [1, 3, 6, 11, 18, 26, 39]
y =
     1     3     6    11    18    26    39
>> |
```



Figures:

One of the most important features of MATLAB is its advanced built-in functions for plotting figures. `plot()` function is widely used for plotting purposes:

```
>> plot(x,y)
>> |
```

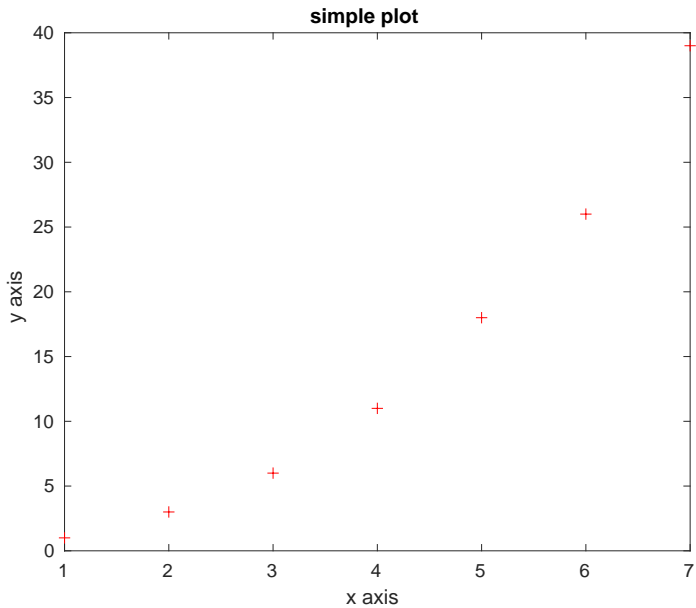




Attributes of Plot:

You can change the color and data point representations inside your plot. Furthermore, you can add labels to figure axes by using `xlabel()` and `ylabel()` functions and also you can add a title to your plot using the `title()` function:

```
>> plot(x,y, 'r+');  
>> xlabel('x axis');  
>> ylabel('y axis');  
>> title('simple plot');  
>>
```





Images:

Using MATLAB, you can read, write and process images.

```
>> moon_step = imread('moon_step.jpg');  
>> imshow(moon_step)  
Warning: Image is too big to fit on screen; displaying at 67%  
> In images.internal.initSize (line 71)  
   In imshow (line 327)  
>>
```

The following image is downloaded from <https://images.nasa.gov/details-6901250.html>

