Computer Aided Design (CAD)

Lecture 5

- Arrays (2).
- Functions

Dr.Eng. Basem ElHalawany

Schedule (Draft)

Topics	Estimated Duration (# Lectures)
Introduction	1
Introduction to Matlab Environment	1
Matlab Programing (m-files)	5 **(3/5)**
Modeling using Matlab Simulink Tool	4
Communication Systems Simulation (Applications)	3
Midterm	8th Week
Introduction to FPGA + Review on Digital Logic/Circuits	2
VHDL Modeling Language	4
VHDL Application	2
Introduction to OPNET Network Simulator	3
Course Closeout / Feedback/ project (s) Delivery	1

The Lecture is based on :

A. **Matlab by Example: Programming Basics, Munther Gdeisat**

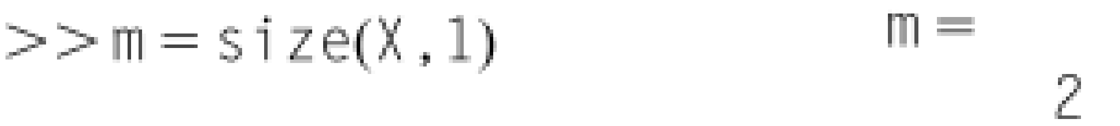
4 Arrays in Matlab

4.1.6 Finding the Size of an Array

- ➢ Matlab enables you to determine the number of rows and columns in an array.

✓ To find the number of rows in X, type

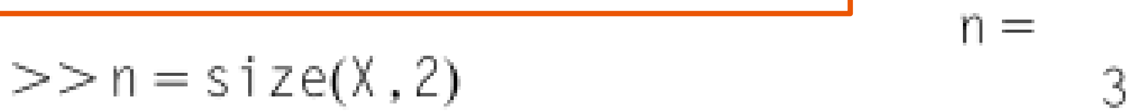
$$\mathbf{X} = \begin{bmatrix} 1 & 2 & 4 \\ 7 & 3 & 5 \end{bmatrix}$$



```
>>m=size(X,1)          m =

                           2
```

- Here the “1” keyword in the size function indicates that we wish to know the **first dimension** of the array X, that is, the number of rows.

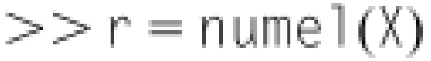
✓ To find the number of columns in X, type



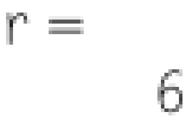
```
>>n=size(X,2)          n =

                           3
```

- Here the “2” keyword in the size function indicates that we wish to know the **second dimension** of the array X, that is, the number of columns.

- ✓ To find the total number of elements in the array X, type

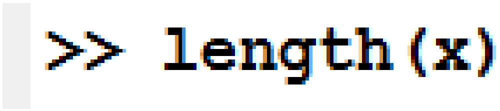


```
>> r = numel(X)
```

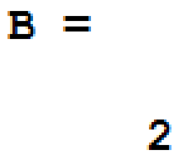


```
r =

     6
```

- ✓ To find the number of dimension of the array X, type



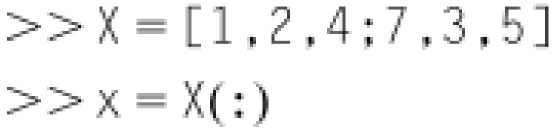
```
>> length(x)
```



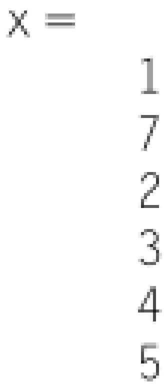
```
B =

     2
```

4.1.7 Converting an Array to a Column Vector

- You can convert an array to a column vector using the colon (:) operator.
- Note that the elements have been extracted from the array X, in a column-by-column fashion.



```
>>X=[1,2,4;7,3,5]
>>x=X(:)
```



```
x =
      1
      7
      2
      3
      4
      5
```

4.1.8 Arrays Concatenation

Arrays can be concatenated (combined) together to produce larger arrays.

Example 8

Concatenate the two arrays

$$\mathbf{X} = \begin{bmatrix} 1 & 2 & 4 \\ 7 & 3 & 5 \end{bmatrix} \quad \text{and} \quad \mathbf{Z} = \begin{bmatrix} 1 & 2 & 5 \\ 8 & 3 & 4 \\ 9 & 6 & 7 \end{bmatrix}$$

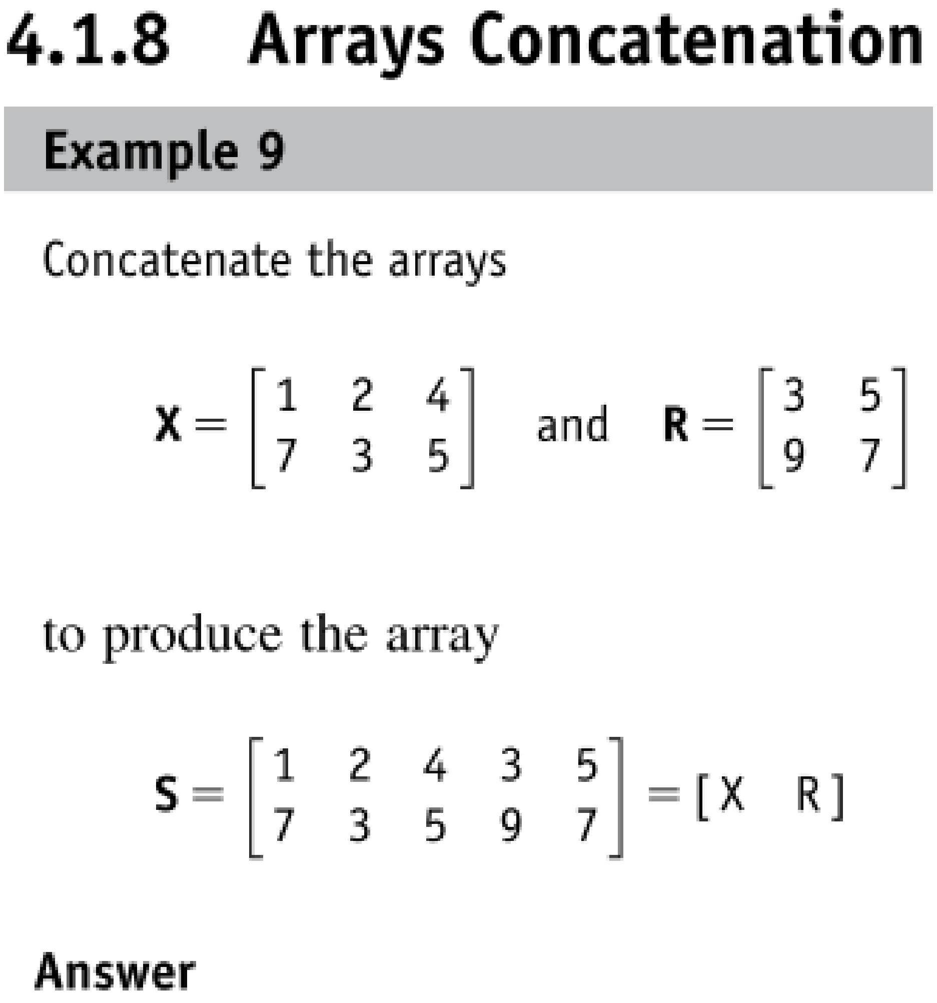
to produce the array

$$\mathbf{F} = \begin{bmatrix} 1 & 2 & 5 \\ 8 & 3 & 4 \\ 9 & 6 & 7 \\ 1 & 2 & 4 \\ 7 & 3 & 5 \end{bmatrix} = \begin{bmatrix} Z \\ X \end{bmatrix}$$

Answer

```
>>X=[1,2,5;8,3,4;9,6,7];
>>Z=[1,2,4;7,3,5];
>> F = [ Z ; X ] ;
```

- Note that here we have used the semicolon (;) to combine X and Z arrays in the vertical direction.



4.1.8 Arrays Concatenation

Example 9

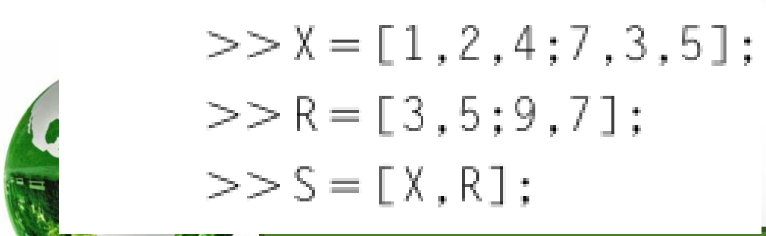
Concatenate the arrays

$$\mathbf{X} = \begin{bmatrix} 1 & 2 & 4 \\ 7 & 3 & 5 \end{bmatrix} \quad \text{and} \quad \mathbf{R} = \begin{bmatrix} 3 & 5 \\ 9 & 7 \end{bmatrix}$$

to produce the array

$$\mathbf{S} = \begin{bmatrix} 1 & 2 & 4 & 3 & 5 \\ 7 & 3 & 5 & 9 & 7 \end{bmatrix} = [\mathrm{X} \quad \mathrm{R}]$$

Answer



```
>>X=[1,2,4;7,3,5];
>>R=[3,5;9,7];
>>S=[X,R];
```

- Note that here we have used the comma (,) to combine X and R arrays in the horizontal direction.

Lesson 4.3 Accessing Elements in Arrays

4.3.1.1 Row-and-Column Indexing Method

$$\mathbf{X} = \begin{bmatrix} 3 & 4 & 8 & 12 \\ 2 & 5 & 7 & 11 \\ 1 & 6 & 9 & 10 \end{bmatrix} \qquad \begin{bmatrix} X_{1,1} & X_{1,2} & X_{1,3} & X_{1,4} \\ X_{2,1} & X_{2,2} & X_{2,3} & X_{2,4} \\ X_{3,1} & X_{3,2} & X_{3,3} & X_{3,4} \end{bmatrix}$$

We refer to an element in the array **X** as $X_{m,n}$,

m refers to the row number and n refers to the column number.

```
>> X = [3,4,8,12;2,5,7,11;1,6,9,10];
```

To access the element $X_{1,1}$, type at the Matlab **Command Prompt**

```
>> X(1,1)            ans =
                          3
```

To access the element $X_{2,3}$, type at the Matlab **Command Prompt**

```
>> f = X(2,3)            f =
                              7
```

Lesson 4.3 Accessing Elements in Arrays

4.3.1.1 *Row-and-Column Indexing Method*

- To access the last element in the first row of X, type

```
>> s = X(1,end);
```

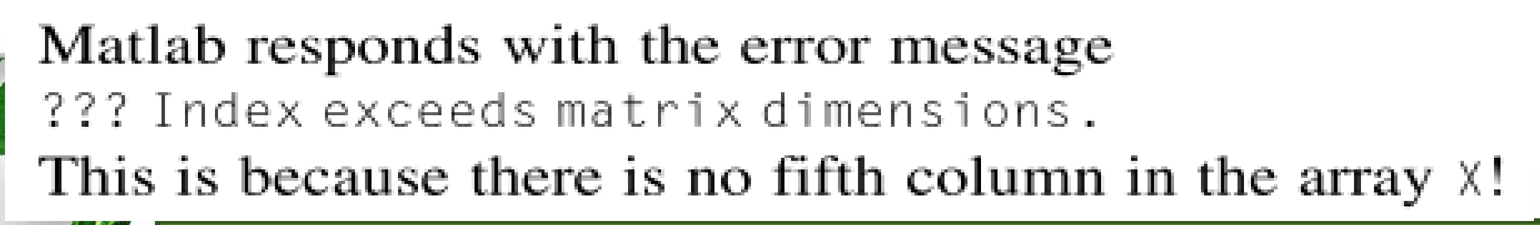
Or alternatively use the command
```
>> s = X(1,4);
```

- To access the last element in the third column of X,

```
>> t = X(end,3);
```

Let us try to access the element $X_{1,5}$ as follows:

```
>> X(1,5)
```



Matlab responds with the error message

```
??? Index exceeds matrix dimensions.
```

This is because there is no fifth column in the array X!

4.3.1.2 Linear-Indexing Method

The linear indices of the elements of **X** are

$$\begin{bmatrix} X_1 & X_4 & X_7 & X_{10} \\ X_2 & X_5 & X_8 & X_{11} \\ X_3 & X_6 & X_9 & X_{12} \end{bmatrix}$$

$$\mathbf{X} = \begin{bmatrix} 3 & 4 & 8 & 12 \\ 2 & 5 & 9 & 11 \\ 1 & 6 & 7 & 10 \end{bmatrix}$$

```
>> a = X(1)

                    a =
                        3
```

4.3.2 Accessing Rows in an Array

- You can use the colon operator (:) to access a row in an array.
- To access the first row, type

```
>>a = X(1,:)
```

Matlab responds with

```
a =

    3    4    8   12
```

- To access the last row, type

```
>>b = X(end,:)
```

```
b =

    1    6    9   10
```

- To access the last two rows, type

```
>>B = X(end-1:end,:)
```

```
B =

    2    5    7   11
    1    6    9   10
```

- To access the first and the third rows, type

```
>>C = X([1,3],:)
```

```
C =

    3    4    8   12
    1    6    9   10
```

4.3.3 Accessing Columns in an Array

- You can use the colon operator (:) to access a column in an array.
- To access the first column, type

```
>>a = X(: ,1)
```

Matlab responds with

```
a =
      3
      2
      1
```

- To access the last column , type

```
>>b = X(: ,end)
```

```
b =
     12
     11
     10
```

- To access the first and second columns, type

```
>>C = X(: ,[1, 2])
```

```
C =
      3   4
      2   5
      1   6
```

4.3.4 Accessing a Group of Elements in an Array Using Their Indices

$$\mathbf{X} = \begin{bmatrix} 3 & 4 & \boxed{8} & 12 \\ 2 & 5 & \boxed{7} & 11 \\ 1 & 6 & 9 & 10 \end{bmatrix}$$

```
>> r = X([1,2],3)
```

$$\mathbf{X} = \begin{bmatrix} 3 & 4 & 8 & 12 \\ 2 & \boxed{5} & \boxed{7} & \boxed{11} \\ 1 & 6 & 9 & 10 \end{bmatrix}$$

```
>> e = X(2,[2,3,4])
```

$$\mathbf{X} = \begin{bmatrix} 3 & 4 & 8 & 12 \\ 2 & \boxed{5} & \boxed{7} & \boxed{11} \\ 1 & \boxed{6} & \boxed{9} & \boxed{10} \end{bmatrix}$$

```
>> G = X([2,3],[2,3,4])
```

4.3.5 Accessing Elements in an Array Using Their Values

- To find the indices of the elements whose values is greater than 7, type

$$E = \begin{bmatrix} 7 & 3 & 9 \\ 1 & 0 & 2 \\ 5 & 8 & 4 \end{bmatrix}$$

```
>>[a,b]=find(E>7);
>>[a,b]
```

```
ans =
        3   2
        1   3
```

The output of Matlab means that the elements E(3,2) and E(1,3) are greater than 7.

- To find the indices of the elements in the array E whose value is less than 3, type

```
>>[c,d]=find(E<3);
>>[c,d]
```

```
ans =
        2   1
        2   2
        2   3
```

E(2,1), E(2,2) and E(2,3)

- To find the values of the elements in E that have values that are less than 3, type

```
>>f=find(E<3);
>>r=E(f)
```

```
r =
     1
     0
     2
```

Lesson 4.5 Plotting Arrays

4.5.2 3D Plot an Array with the `mesh` Function

Let us plot the function $\mathbf{Z} = \mathbf{X}^2 - \mathbf{Y}^2$,

- X is in the range of [- 2, 2] and
- Y is in the range of [- 3, 3].

```
    x = -2:1:2;
    y = -3:1:3;
[X,Y] = meshgrid(x,y);
```

Matlab produces the arrays X and Y as follows:

```
X =                              Y =

      -2   -1   0   1   2              -3   -3   -3   -3   -3
      -2   -1   0   1   2              -2   -2   -2   -2   -2
      -2   -1   0   1   2              -1   -1   -1   -1   -1
      -2   -1   0   1   2               0    0    0    0    0
      -2   -1   0   1   2               1    1    1    1    1
      -2   -1   0   1   2               2    2    2    2    2
      -2   -1   0   1   2               3    3    3    3    3
```

To evaluate Z using Matlab, type at the **Command Prompt**

```
Z = X.^2 - Y.^2
```

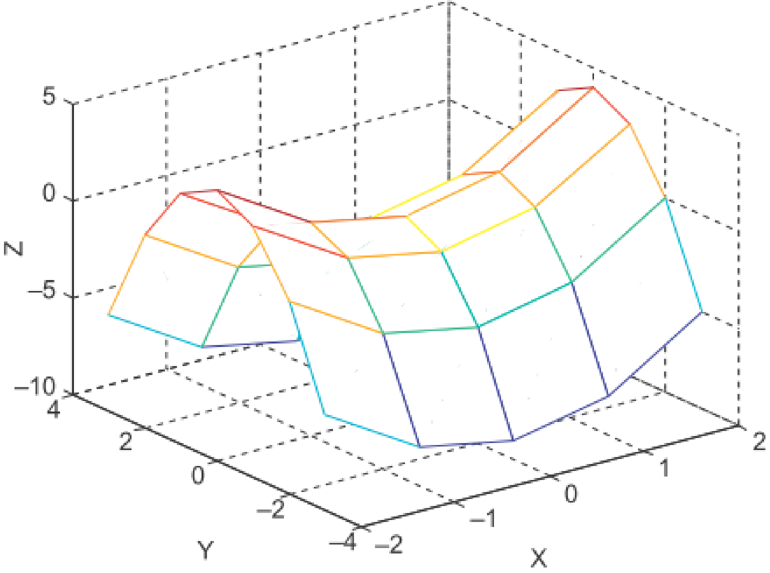
```
Z =

    -5   -8   -9   -8   -5
     0   -3   -4   -3    0
     3    0   -1    0    3
     4    1    0    1    4
     3    0   -1    0    3
     0   -3   -4   -3    0
    -5   -8   -9   -8   -5
```

4.5.2 3D Plot an Array with the `mesh` Function

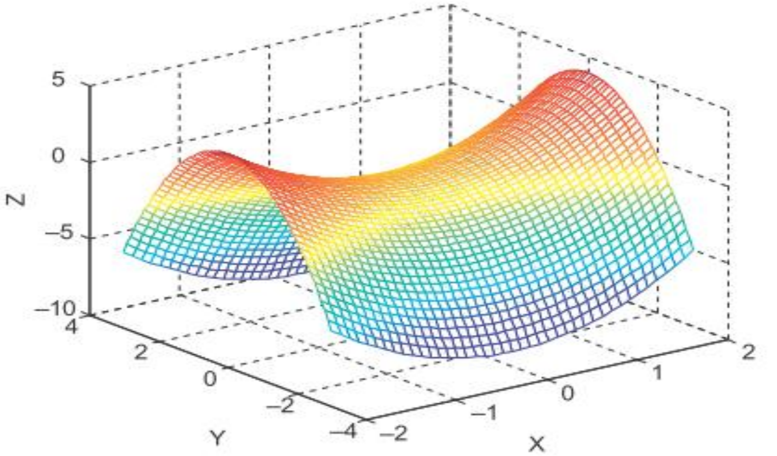
To plot the array `Z` versus `X` and `Y`, type at the **Command Prompt**

```
  mesh(X,Y,Z)
```

```
clear; clc; close all
x = -2:1:2;
y = -3:1:3;
[X,Y]=meshgrid(x,y);
Z = X.^2-Y.^2;
mesh(X,Y,Z)
xlabel('X')
ylabel('Y')
zlabel('Z')
```



To improve the resolution of the 3D plot, you need to increase the number of points within the X and Y arrays, and then recompute the Z array.

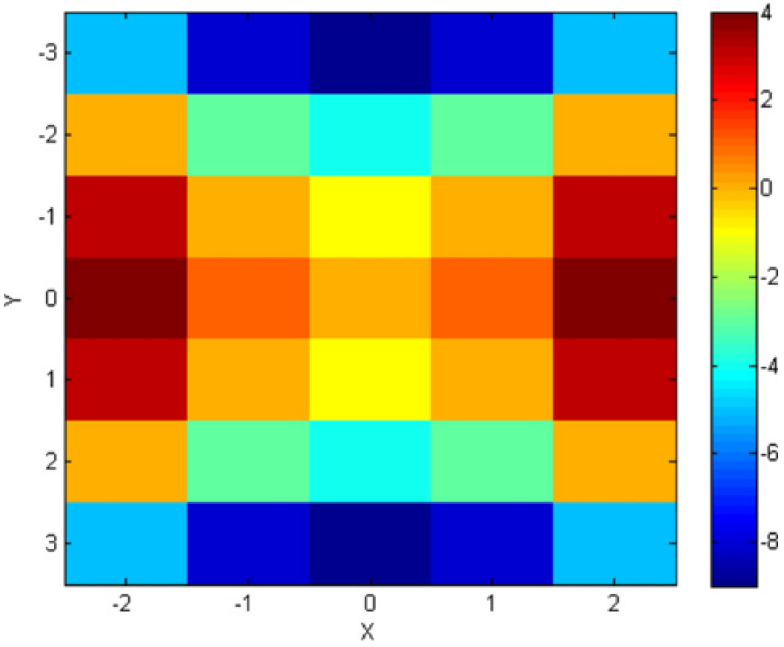


4.5.6 Background for 2D Plotting of Arrays

Example 1

Plot the function $\mathbf{Z} = \mathbf{X}^2 - \mathbf{Y}^2$, where $\mathbf{X}$ is in the range of $[-2, 2]$ and $\mathbf{Y}$ is in the range of $[-3, 3]$ as a 2D plot.

$$\mathbf{Z} = \begin{bmatrix} -5 & -8 & -9 & -8 & -5 \\ 0 & -3 & -4 & -3 & 0 \\ 3 & 0 & -1 & 0 & 3 \\ 4 & 1 & 0 & 1 & 4 \\ 3 & 0 & -1 & 0 & 3 \\ 0 & -3 & -4 & -3 & 0 \\ -5 & -8 & -9 & -8 & -5 \end{bmatrix}$$

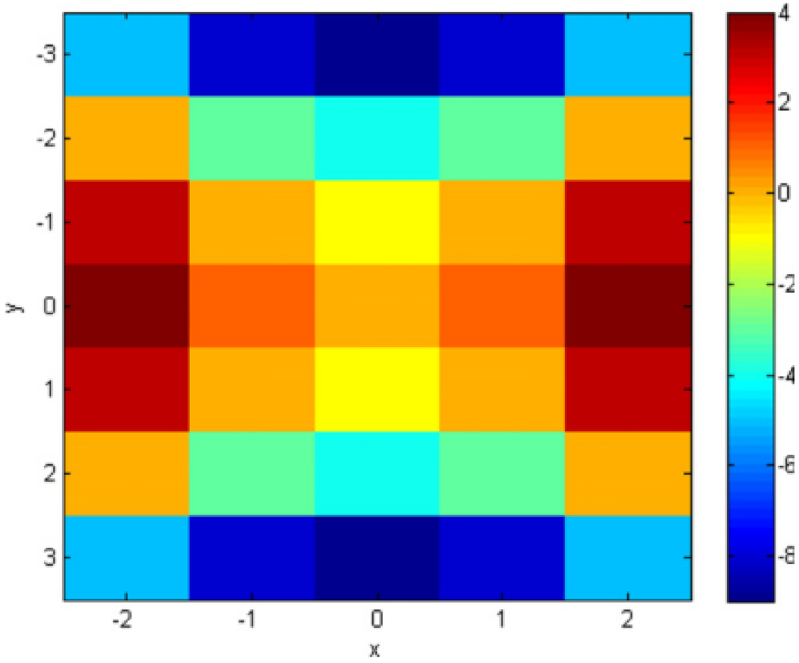


- there is a relationship between the values of Z, the color bar, and the color of the 2D graph.

2D Plot an Array with the `imagesc` Function

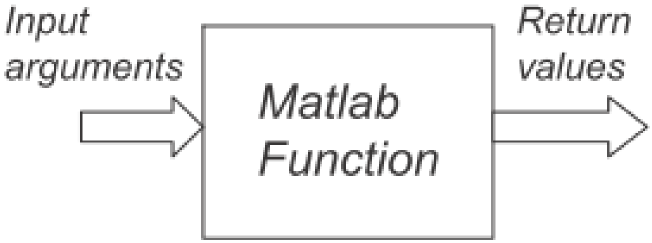
2D Plot an Array with the `imagesc` Function

```
x = −2:1:2;
y = −3:1:3;
[X,Y] = meshgrid(x,y);
Z = X.^2 − Y.^2;
imagesc(x,y,Z)
xlabel('x')
ylabel('y')
colorbar
```



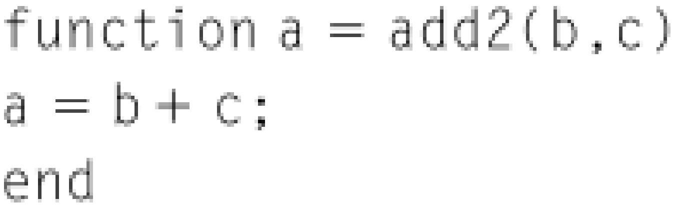
5 Matlab Functions

accepts
one or more
Matlab variables

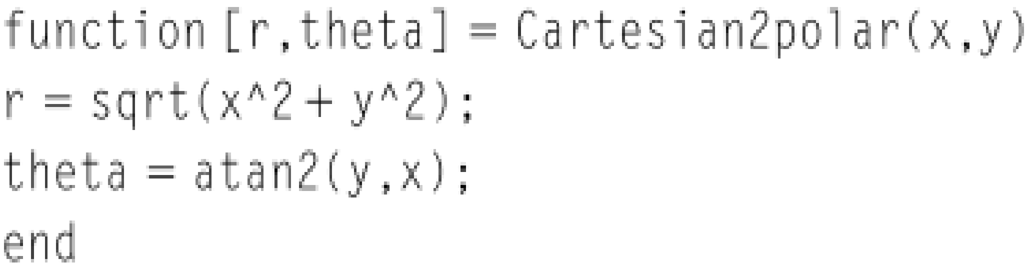


returns
one or more
Matlab variables

operates on them in some way



```
function a = add2(b,c)
a = b + c;
end
```

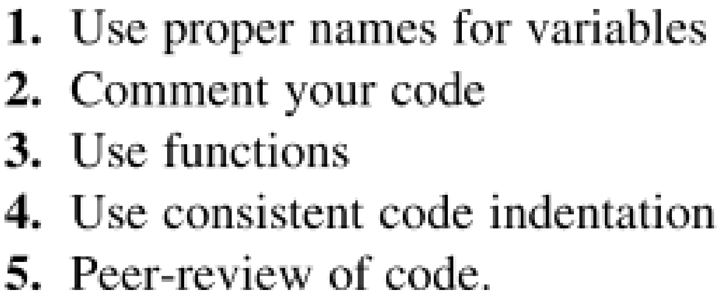


```
function [r,theta] = Cartesian2polar(x,y)
r = sqrt(x^2 + y^2);
theta = atan2(y,x);
end
```

The Purpose of a Function

1 *Improves Code Readability*

- Five techniques to improve the readability of your programs:



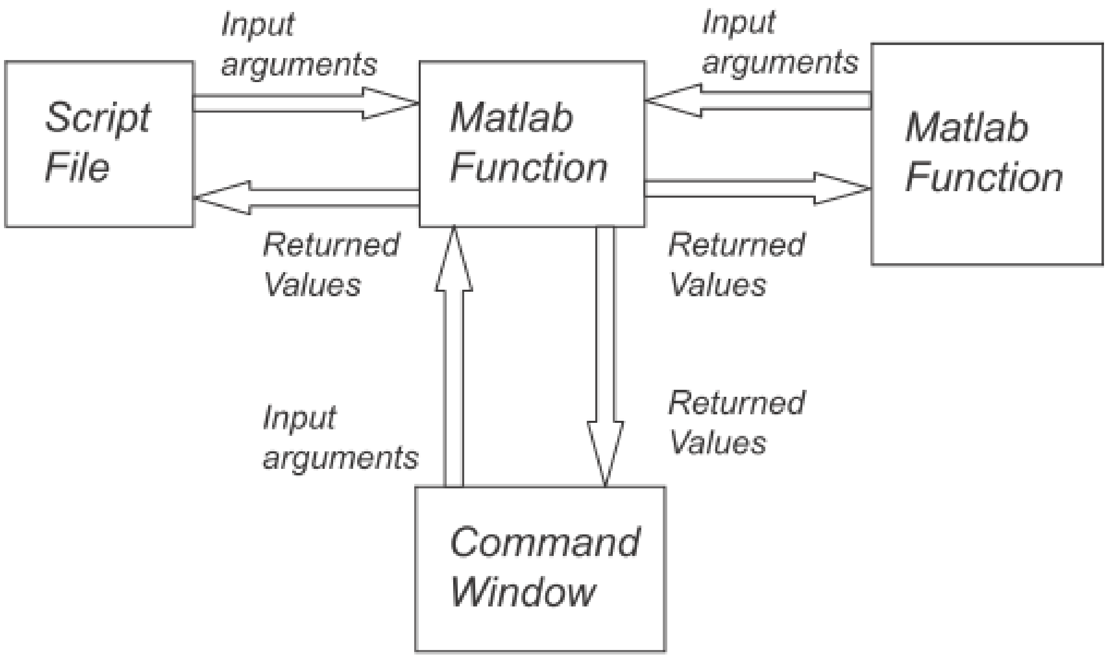
1. Use proper names for variables
2. Comment your code
3. Use functions
4. Use consistent code indentation
5. Peer-review of code.

2 *Improves Code Reusability*

- A piece of code should be typed only once, then used as many times as required.

5.1.3 Calling a Matlab Function

- You can call a Matlab function from a script file, from the Command Window, or from another function.



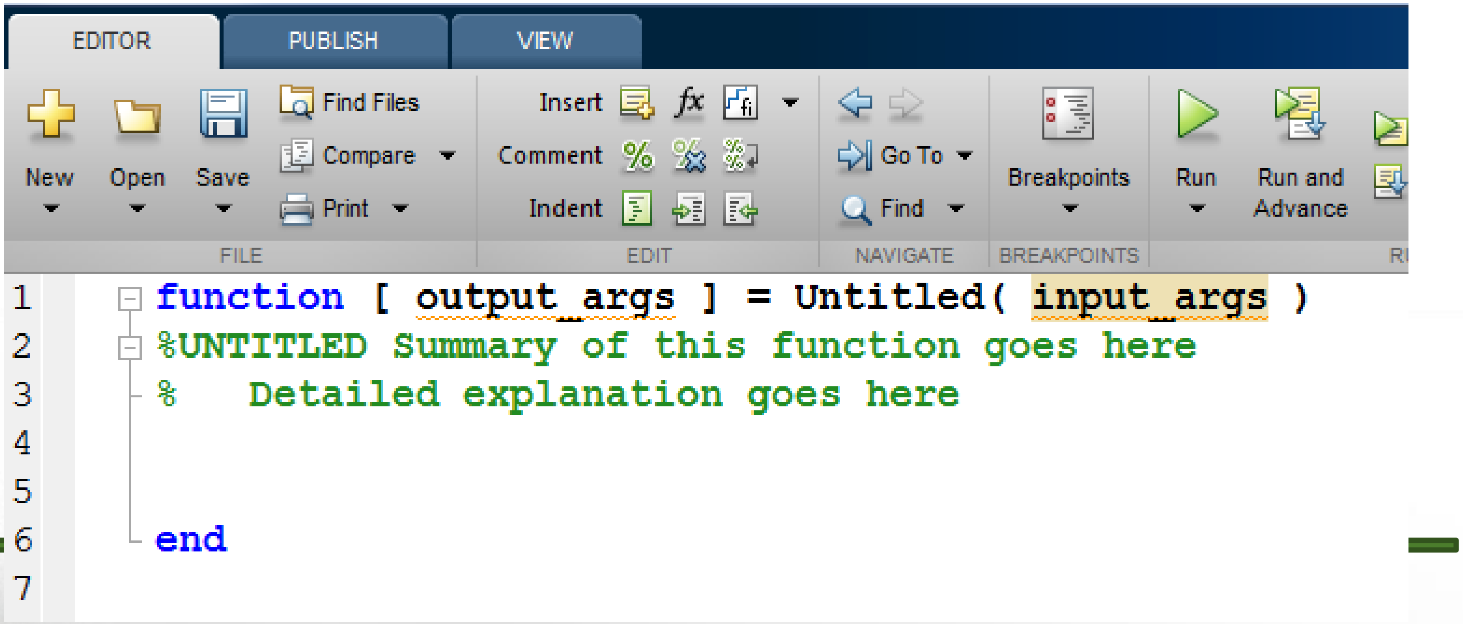
Lesson 5.2 Creating Functions

- Choosing the name of a function is a similar process to that of choosing the name of a script file
- Similar restrictions must be taken into consideration.

To create the add2 function, go to:

Menu→File→New→Function.

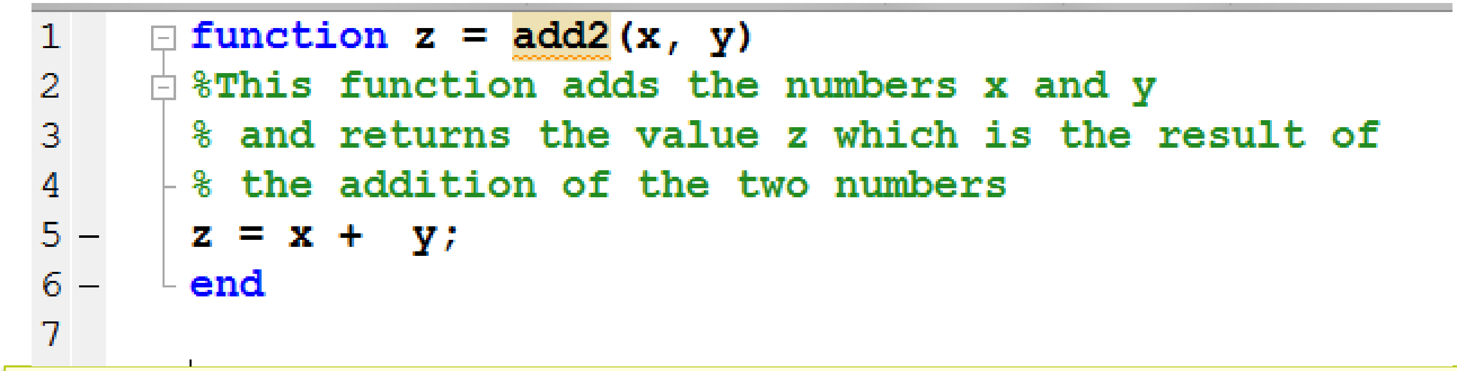
The Matlab Editor pops up and write your function as :



```
function [ output_args ] = Untitled( input_args )
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here


end

```

Lesson 5.2 Creating Functions

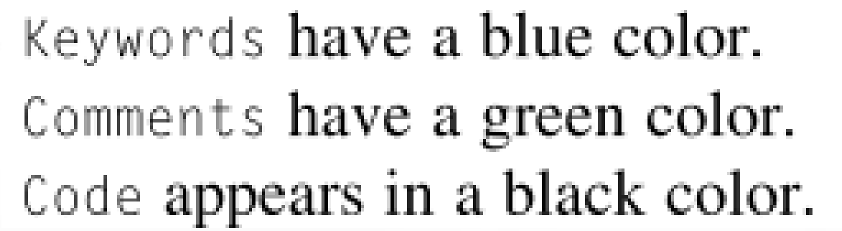
- ➢ Delete everything in the Editor and type the following code in the Editor



```
function z = add2(x, y)
%This function adds the numbers x and y
% and returns the value z which is the result of
% the addition of the two numbers
z = x +  y;
end
```

- ➢ Save the add2 function using the name add2.m.
- ➢ The name of the file MUST be exactly the same as the name of the function and must be followed by the .m extension.

- ➢ Note that the Matlab Editor uses different colored text to simplify the programming process:



Keywords have a blue color.

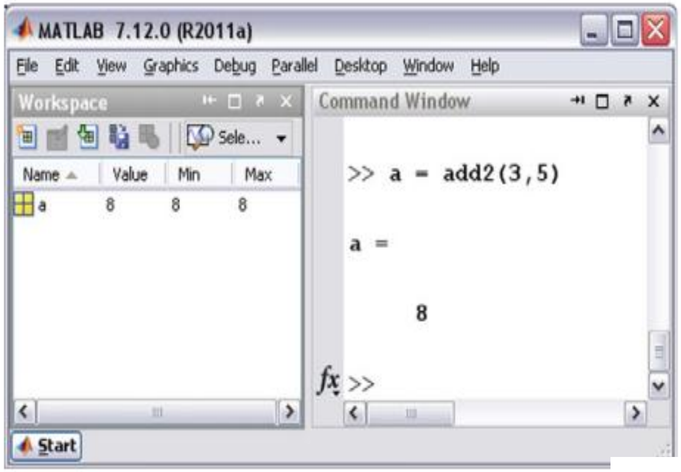
Comments have a green color.

Code appears in a black color.

5.2.4 Calling a Matlab Function

5.2.4.1 Calling a Matlab Function from the Command Window

```
>>a = add2 (3,5);
```

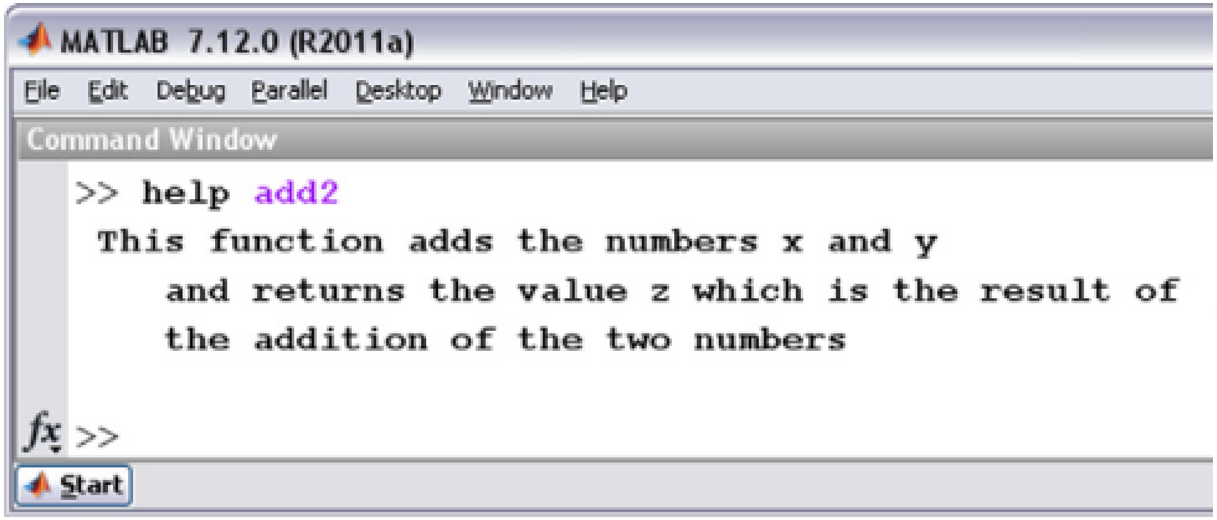


Name	Value	Min	Max
a	8	8	8

```
>> a = add2(3,5)

a =

     8

>>
```

- ➢ A new variable a is created and its value is 8 in the Workspace window .
- ➢ Note that the function arguments x, y and the returned value z that is created by the add2 function do not actually appear in the Workspace window and they do not exist in Matlab memory,

```
>>help add2
```



```
>> help add2
 This function adds the numbers x and y
    and returns the value z which is the result of
    the addition of the two numbers

>>
```

5.2.4 Calling a Matlab Function

5.2.4.2 Calling a Matlab Function from a Script File

```
a = 1;
b = 2;
c = add2(a,b)
```

5.2.4.3 Calling a Matlab Function from Another Function

```
function d = add3(a, b, c)
e = add2(a,b);
d = add2(e, c);
end
```

To call this function, at the **Command Prompt** type

```
>>z = add3(1,2,3)
```

Matlab responds with

```
z  =
      6
```

5.2.5 A Matlab Function Returning Two Values

```
function [addition, subtraction] = add_sub(x,y)
addition = x + y;
subtraction = x − y;
end
```

To call this function, at the **Command Prompt** type

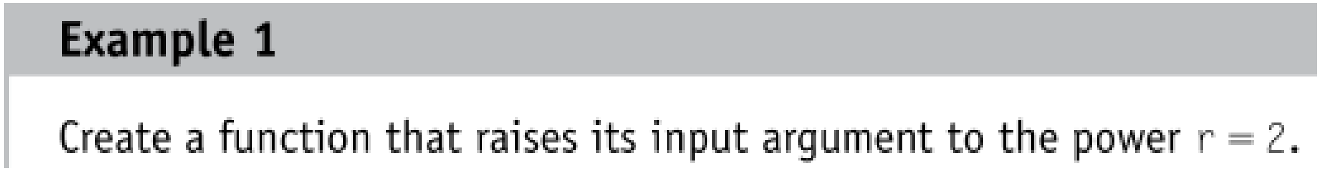
```
> >[r, s]= add_sub (5,3)
```

The result of calling this function is

```
r  =
     8

s  =
     2
```

Lesson 5.3 Scope of Matlab Variables in a Function

- ➢ A variable that is created within a function can be only accessed or modified by this function.
- ➢ This variable is called a local variable.



Example 1

Create a function that raises its input argument to the power r = 2.

```
function c = pow(a)
r = 2;
c = a.^r;
end
```

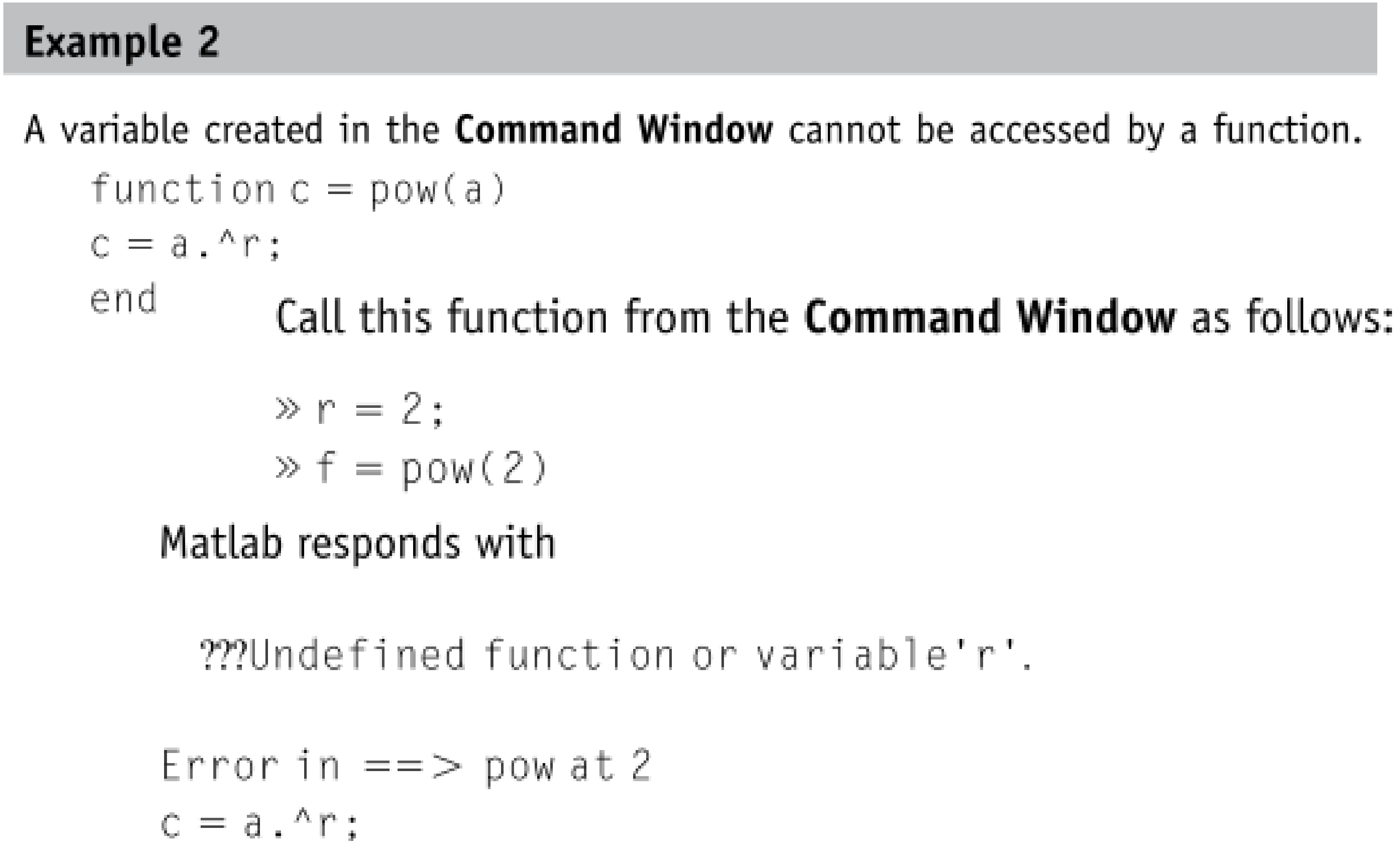
- ➢ The variables a, r, and c are local variables to the function pow and can only be accessed by this function.

```
>>f = pow(3)

f =
     9
```

```
>>r
```

Matlab responds with

```
??? Undefined function or variable 'r'.
```



Example 2

A variable created in the **Command Window** cannot be accessed by a function.

```
function c = pow(a)
c = a.^r;
end
```

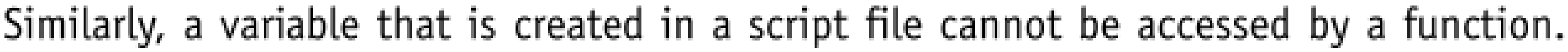
Call this function from the **Command Window** as follows:

```
» r = 2;
» f = pow(2)
```

Matlab responds with

```
???Undefined function or variable'r'.

Error in ==> pow at 2
c = a.^r;
```

- Even though we have created the variable r in the Command Window, the pow function cannot access this variable.



Similarly, a variable that is created in a script file cannot be accessed by a function.