

# ELEG 1043

## Computer Applications in Engineering





# Chapter 11: Introduction to Matlab

**C++** FOR ENGINEERS  
AND SCIENTISTS

# Acknowledgement

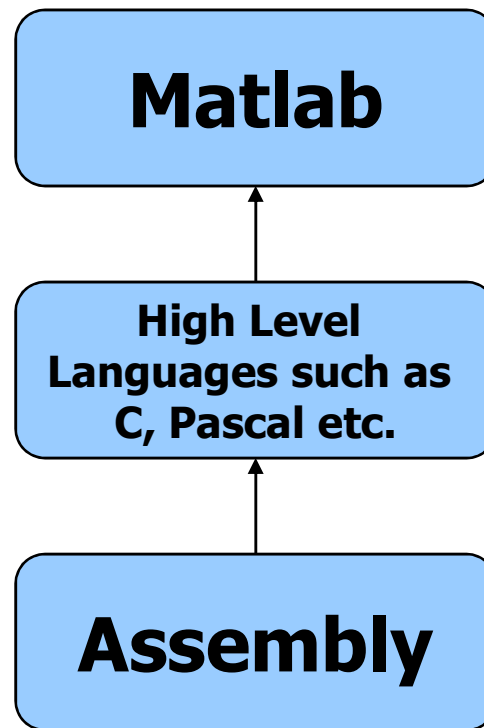
- Some of the slides or images are from various sources. The copyright of those materials belongs to their original owners.

# Objectives

- In this chapter you will learn about:
  - What is Matlab?
  - Matlab Screen
  - Variables, array, matrix, indexing
  - Operators (Arithmetic, relational, logical )
  - Display Facilities
  - Flow Control
  - Using of M-File
  - Debugging

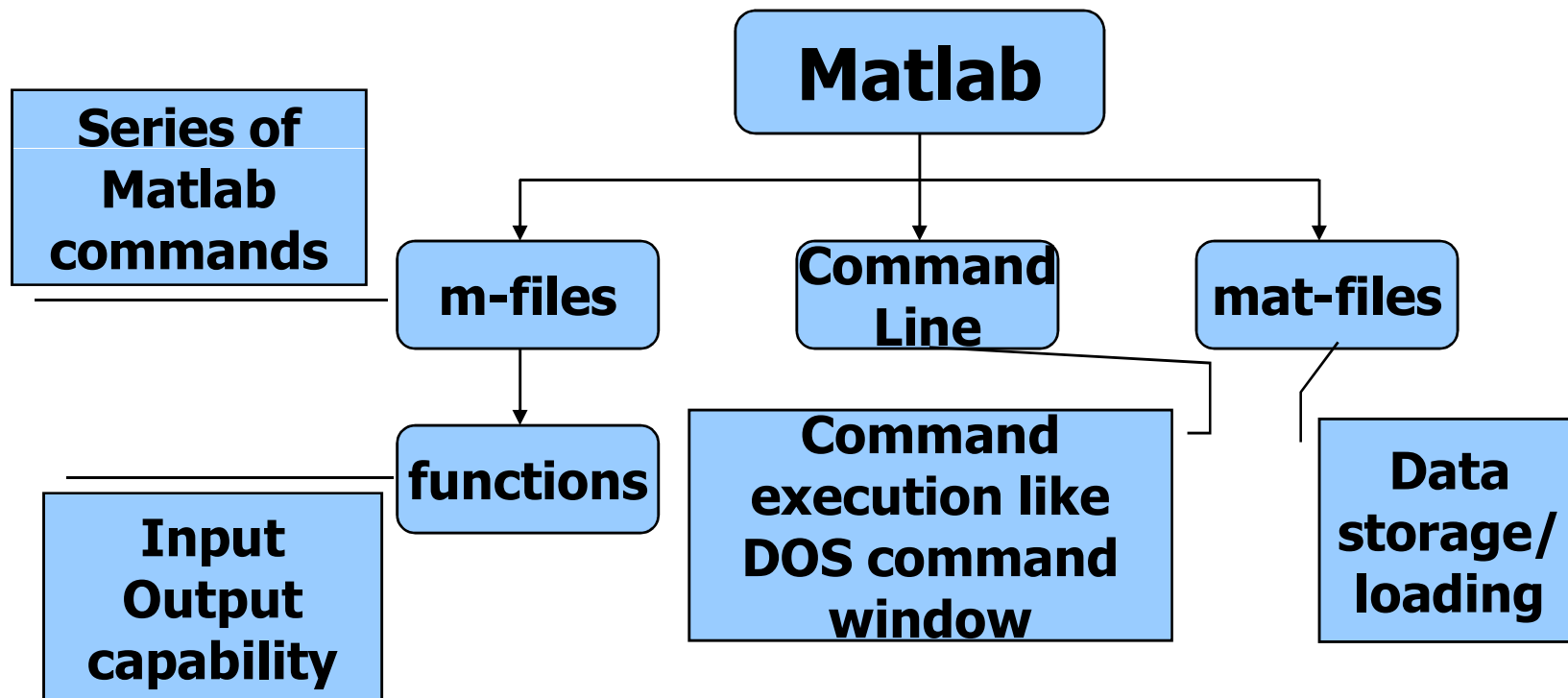
# What is Matlab?

- Matlab is basically a high level language which has many specialized toolboxes for making things easier for us
- How high?



# What are we interested in?

- Matlab is too broad.
- Features



# Matlab Screen

- **Command Window**

- type commands

- **Current Directory**

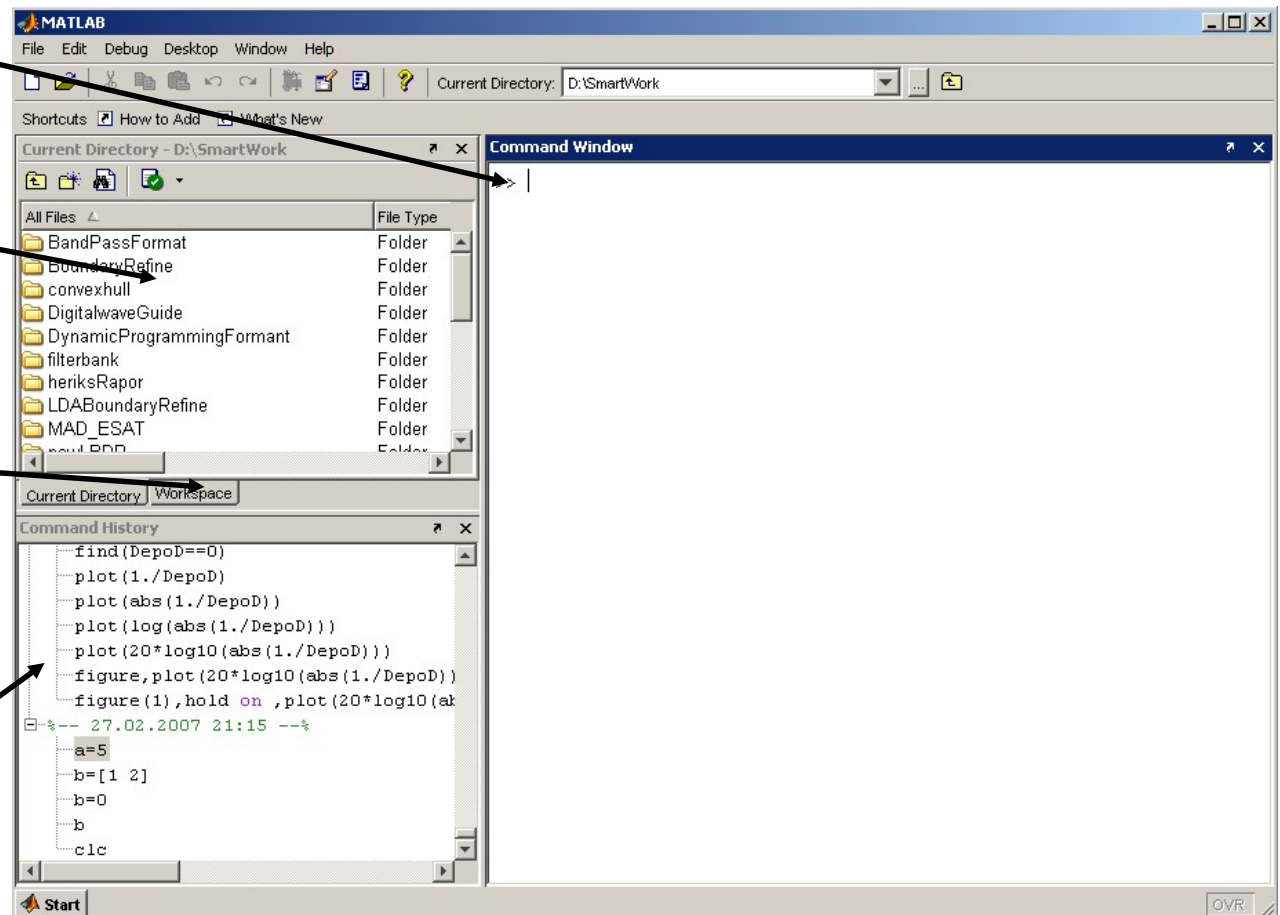
- View folders and m-files

- **Workspace**

- View program variables
- Double click on a variable to see it in the Array Editor

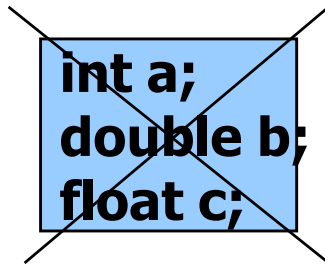
- **Command History**

- view past commands
- save a whole session



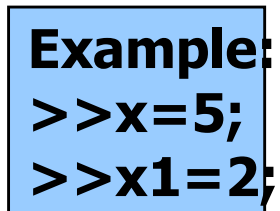
# Variables

- No need for types. i.e.,



```
int a;  
double b;  
float c;
```

- All variables are created with **double precision** unless specified and they are matrices.



```
Example:  
>>x=5;  
>>x1=2;
```

- After these statements, the variables are **1x1 matrices** with double precision



# Workspace

- The workspace is Matlab's memory
- Can **manipulate variables** stored in the workspace

```
>> a=12;
```

```
>> b=10;
```

```
>> c=a+b
```

```
c =
```

```
22
```

# Workspace

- Display contents of workspace

```
>> whos
```

Name	Size	Bytes	Class
a	1x1	8	double array
b	1x1	8	double array
c	1x1	8	double array

Grand total is 3 elements using 24 bytes

```
>>
```

- Delete variable(s) from workspace

```
>> clear a b; % delete a and b from workspace
```

```
>> whos
```

```
>> clear all; % delete all variables from workspace
```

```
>> whos
```

# Variables

- Don't have to **declare type**
- Don't even have to **initialise**
- **Just assign** in command window

>>

>> a=12; %variable a is assigned 12

assign  
operator

suppress  
command  
output

comment  
operator



Try the same line without  
the semicolon and  
comments

# Variables

- View variable contents by simply **typing the variable name** at the command prompt

```
>> a
```

```
a =
```

```
12
```

```
>>
```

```
>> a*2
```

```
a =
```

```
24
```

```
>>
```

# Array, Matrix

- A vector  $\mathbf{x} = [1 \ 2 \ 5 \ 1]$

$\mathbf{x} =$   
1    2    5    1

- A matrix  $\mathbf{t} = [1 \ 2 \ 3; \ 5 \ 1 \ 4; \ 3 \ 2 \ -1]$

$\mathbf{t} =$   
1        2        3  
5        1        4  
3        2        -1

- Transpose         $\mathbf{y} = \mathbf{x}'$          $\mathbf{y} =$   
1  
2  
5  
1

# The **:** operator

- VERY important operator in Matlab
- Means 'to'

```
>> 1:10
```

```
ans =
```

```
1 2 3 4 5 6 7 8 9 10
```

```
>> 1:2:10
```

```
ans =
```

```
1 3 5 7 9
```



Try the following

```
>> x=0:pi/12:2*pi;
```

```
>> y=sin(x)
```

# The **:** operator

```
>>A(3,2:3)
```

```
ans =
```

```
1    7
```

```
>>A(:,2)
```

```
ans =
```

```
2
```

```
1
```

```
1
```

A =

3	2	1
5	1	0
2	1	7



What'll happen if you type A(:,,:) ?

# Long Array, Matrix

- **`t = 1:10`**

**`t =`**  
**`1 2 3 4 5 6 7 8 9 10`**

- **`k = 2:-0.5:-1`**

**`k =`**  
**`2 1.5 1 0.5 0 -0.5 -1`**

- **`X = [1:4; 5:8]`**

**`x =`**  
**`1 2 3 4`**  
**`5 6 7 8`**



# Generating Vectors from functions

- `zeros(M,N)`    **MxN matrix of zeros**

```
x = zeros (1 , 3)
```

```
x =
```

```
0            0            0
```

---

- `ones(M,N)`    **MxN matrix of ones**

```
x = ones (1 , 3)
```

```
x =
```

```
1            1            1
```

---

- `rand(M,N)`    **MxN matrix of uniformly  
distributed random  
numbers on (0,1)**

```
x = rand (1 , 3)
```

```
x =
```

```
0.9501    0.2311    0.6068
```

# Matrix Index

- The matrix indices begin from **1** (not 0 (as in C))
- The matrix indices must be **positive integer**

Given:

```
A =  
  
     3     5     3  
     6     8     2  
     2     7     3
```

```
>> A(6)  
  
ans =  
  
     7
```

```
>> A(3,2)  
  
ans =  
  
     7
```

```
>> A(2,:)   
  
ans =  
  
     6     8     2
```

```
>> A(1:2,2)  
  
ans =  
  
     5  
     8
```

A(-2), A(0)

**Error: ??? Subscript indices must either be real positive integers or logicals.**

A(4,2)

**Error: ??? Index exceeds matrix dimensions.**

# Concatenation of Matrices

- $\mathbf{x} = [1 \ 2]$ ,  $\mathbf{y} = [4 \ 5]$ ,  $\mathbf{z} = [0 \ 0]$

$\mathbf{A} = [\mathbf{x} \ \mathbf{y}]$

1    2    4    5

$\mathbf{B} = [\mathbf{x} \ ; \ \mathbf{y}]$

1 2

4 5

$\mathbf{C} = [\mathbf{x} \ \mathbf{y} \ ; \ \mathbf{z}]$

**Error:**

**??? Error using ==> vertcat CAT arguments dimensions are not consistent.**

# Operators (arithmetic)

- + addition
- - subtraction
- \* multiplication
- / division
- ^ power
- ' matrix transpose

# Matrices Operations

**Given A and B:**

```
>> A = [1 2 3;4 5 6;7 8 9]
```

A =

1	2	3
4	5	6
7	8	9

```
>> B = [3 5 2; 5 2 8; 3 6 9]
```

B =

3	5	2
5	2	8
3	6	9

**Addition**

```
>> X = A + B
```

X =

4	7	5
9	7	14
10	14	18

**Subtraction**

```
>> Y = A - B
```

Y =

-2	-3	1
-1	3	-2
4	2	0

**Product**

```
>> Z = A * B
```

Z =

22	27	45
55	66	102
88	105	159

**Transpose**

```
>> T = A'
```

T =

1	4	7
2	5	8
3	6	9

# Operators (Element by Element)

- `.*` element-by-element multiplication
- `./` element-by-element division
- `.^` element-by-element power

# The use of “.” – “Element” Operation

```
A = [1 2 3; 5 1 4; 3 2 1]
```

A =

```
1  2  3
5  1  4
3  2 -1
```

```
x = A(1,:)
```

x =

```
1  2  3
```

```
y = A(3 ,:)
```

y =

```
3  4 -1
```

```
b = x .* y
```

b =

```
3  8 -3
```

```
c = x ./ y
```

c =

```
0.33  0.5 -3
```

```
d = x .^2
```

d =

```
1  4  9
```

```
K = x^2
```

Error: ??? Error using ==> mpower Matrix must be square.

```
B = x*y
```

Error: ??? Error using ==> mtimes Inner matrix dimensions must agree.

# Manipulating Matrices

<b>&gt;&gt; A'</b>	<b>% transpose</b>
<b>&gt;&gt; B*A</b>	<b>% matrix multiplication</b>
<b>&gt;&gt; B.*A</b>	<b>% element by element multiplication</b>
<b>&gt;&gt; B/A</b>	<b>% matrix division</b>
<b>&gt;&gt; B./A</b>	<b>% element by element division</b>
<b>&gt;&gt; [B A]</b>	<b>% Join matrices (horizontally)</b>
<b>&gt;&gt; [B; A]</b>	<b>% Join matrices (vertically)</b>

**A =**

3	2	1
5	1	0
2	1	7

**B =**

1	3	1
4	9	5
2	7	2



Enter matrix  
B into the  
Matlab  
workspace



Create matrices A and B and try out the the matrix operators in this  
slide



# Operators (relational, logical)

- == Equal to
- $\approx$  Not equal to
- < Strictly smaller
- > Strictly greater
- <= Smaller than or equal to
- >= Greater than equal to
- & And operator
- | Or operator

# Flow Control

- if
- for
- while
- break
- ....

# If Statement Syntax

```
if (Condition_1)
    Matlab Commands
elseif (Condition_2)
    Matlab Commands
elseif (Condition_3)
    Matlab Commands
else
    Matlab Commands
end
```

```
if ((a>3) & (b==5))
    Some Matlab Commands;
end
```

```
if (a<3)
    Some Matlab Commands;
elseif (b~=5)
    Some Matlab Commands;
end
```

```
if (a<3)
    Some Matlab Commands;
else
    Some Matlab Commands;
end
```

# For loop syntax

```
for i=Index_Array
    Matlab Commands
end
```

```
for i=1:100
    Some Matlab Commands;
end
```

---

```
for j=1:3:200
    Some Matlab Commands;
end
```

---

```
for m=13:-0.2:-21
    Some Matlab Commands;
end
```

---

```
for k=[0.1 0.3 -13 12 7 -9.3]
    Some Matlab Commands;
end
```

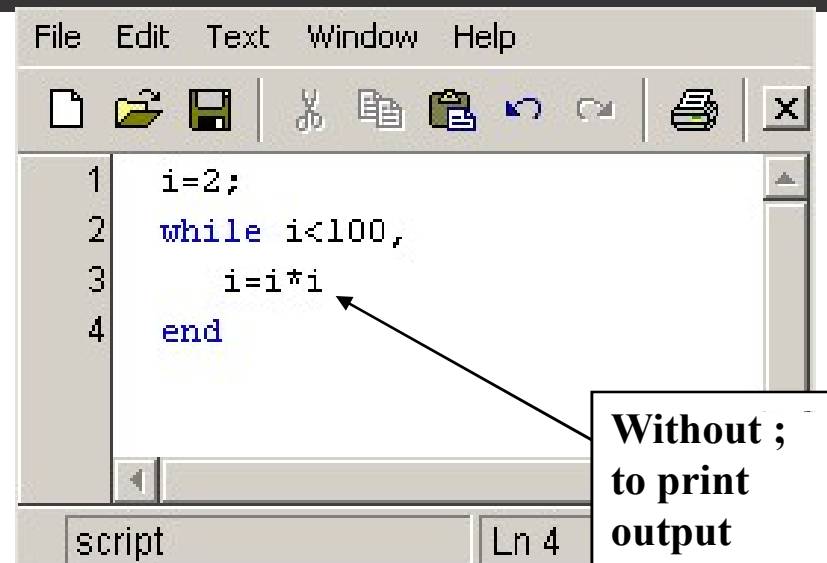
# While Loop Syntax

```
while (condition)
    Matlab Commands
end
```

## Example

```
while ((a>3) & (b==5))
    Some Matlab Commands;
end
```

# While Loop Syntax



```
File Edit Text Window Help
[Icons]
1 i=2;
2 while i<100,
3     i=i*i
4 end
```

Without ;  
to print  
output

script Ln 4

```
i =  
    4  
i =  
    16  
i =  
    256
```