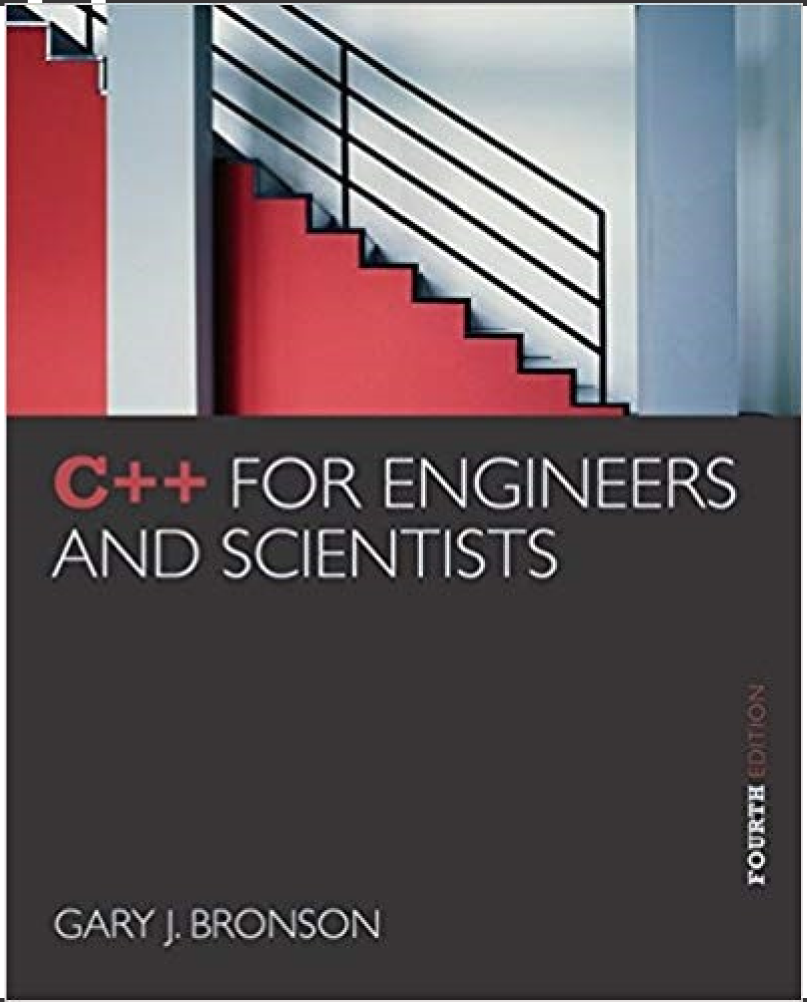
ELEG 1043
Computer Applications in Engineering



Chapter 11: Introduction to Matlab

C++ FOR ENGINEERS
AND SCIENTISTS

Acknowledgement

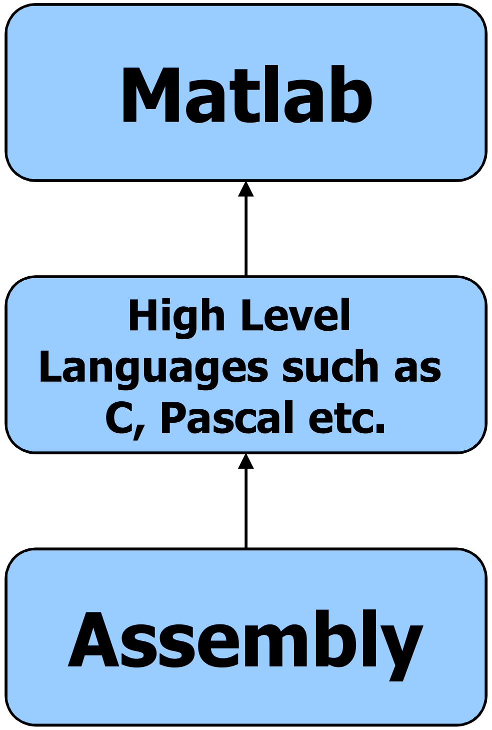
- Some of the slides or images are from various sources. The copyright of those materials belongs to their original owners.

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?



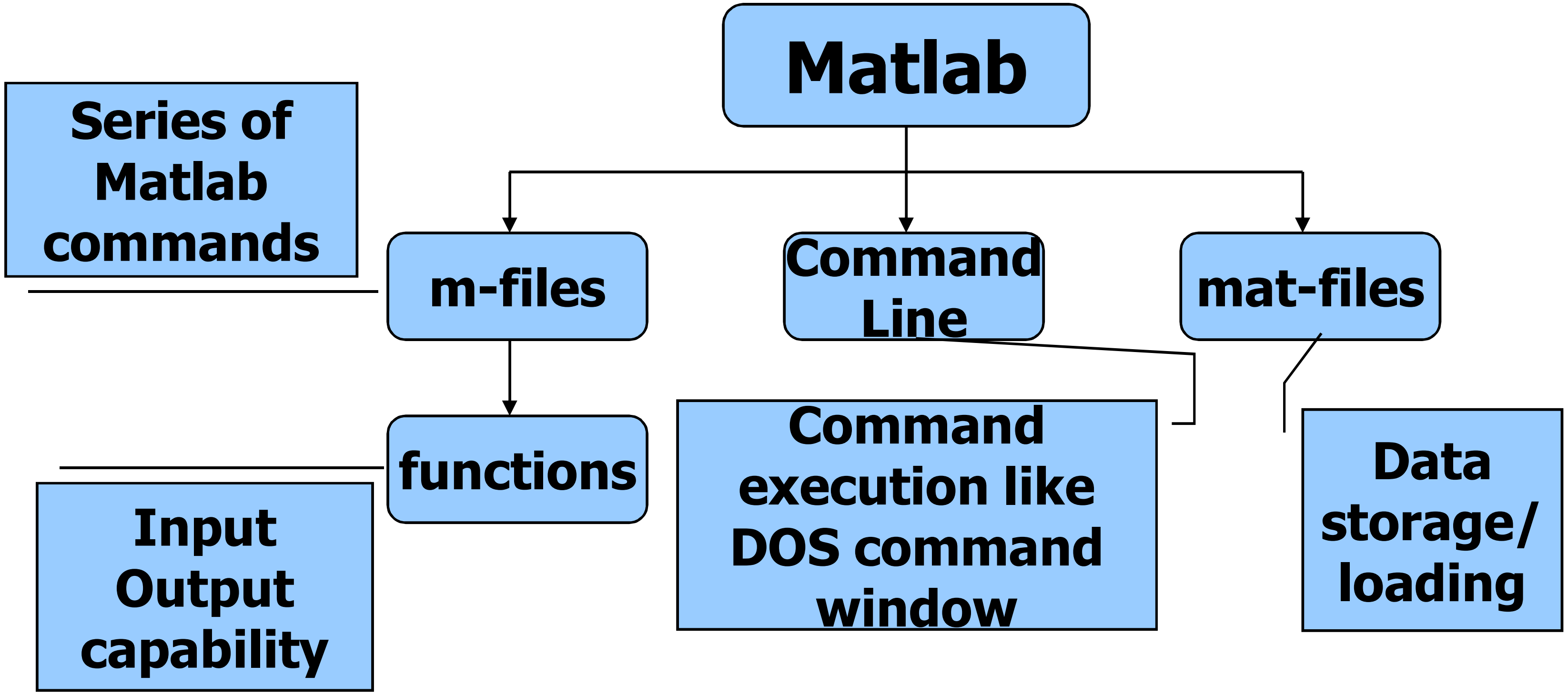
Matlab

High Level Languages such as C, Pascal etc.

Assembly

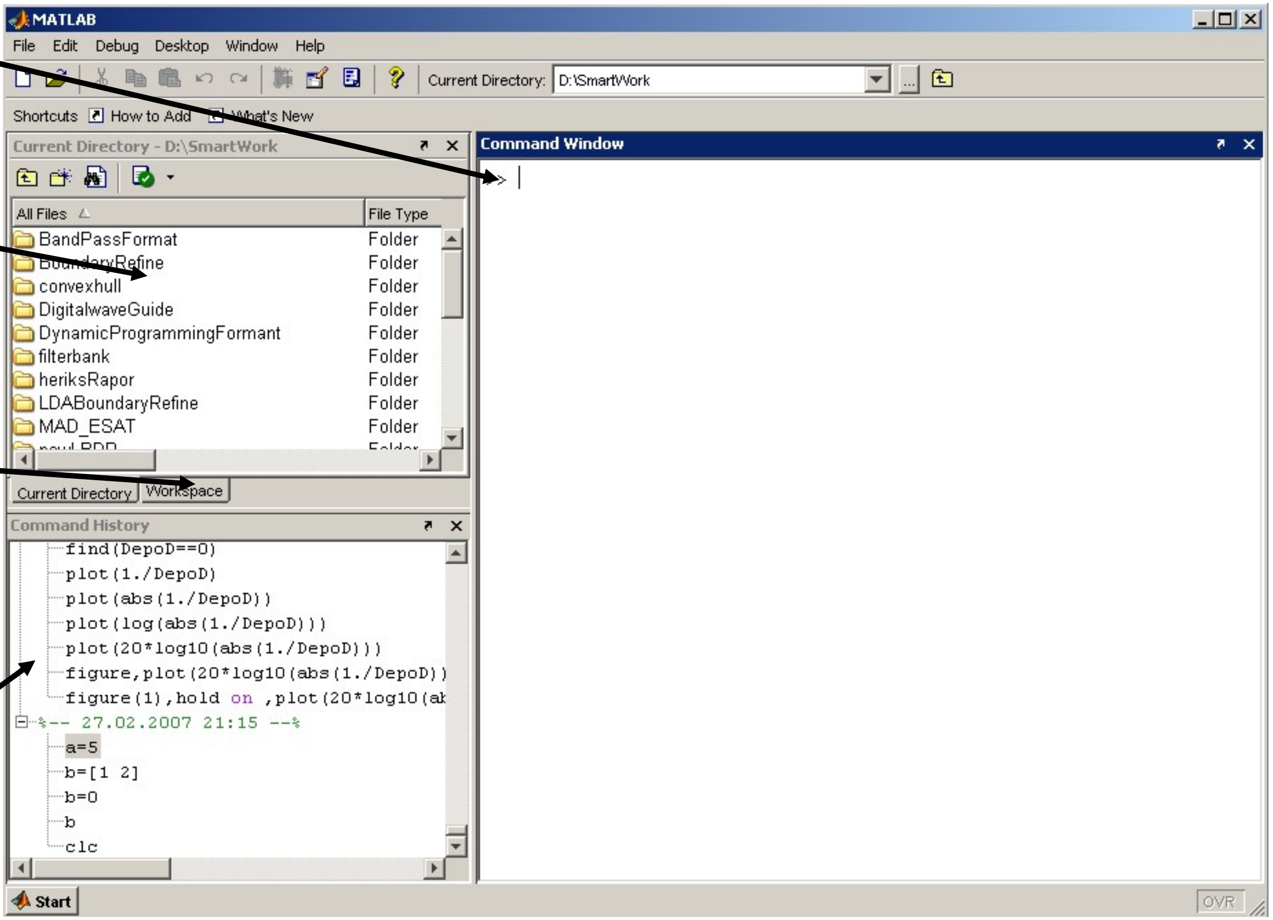
What are we interested in?

- 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 using diary


MATLAB
File Edit Debug Desktop Window Help
Current Directory: D:\SmartWork
Shortcuts How to Add What's New
Current Directory - D:\SmartWork
Command Window
All Files
File Type
Current Directory
Workspace
Command History
Start
OVR

All Files	File Type
BandPassFormat	Folder
BoundaryRefine	Folder
convexhull	Folder
DigitalwaveGuide	Folder
DynamicProgrammingFormant	Folder
filterbank	Folder
heriksRapor	Folder
LDABoundaryRefine	Folder
MAD_ESAT	Folder

```
find(DepoD==0)
plot(1./DepoD)
plot(abs(1./DepoD))
plot(log(abs(1./DepoD)))
plot(20*log10(abs(1./DepoD)))
figure,plot(20*log10(abs(1./DepoD))
figure(1),hold on ,plot(20*log10(ak
%-- 27.02.2007 21:15 --%
a=5
b=[1 2]
b=0
b
clc
```

```
>> |
```

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
>>
```

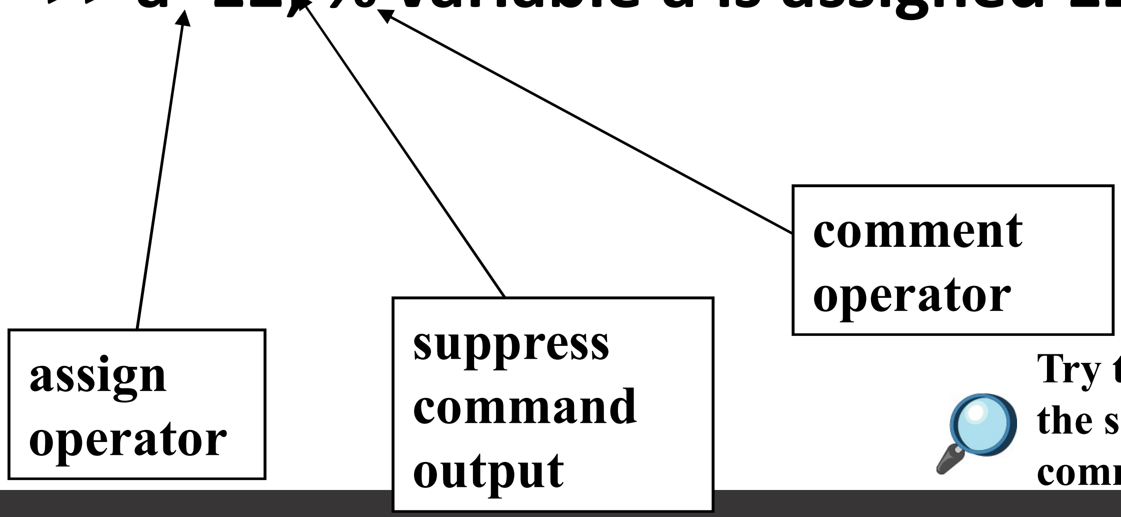
- **Delete variable(s) from workspace**

```
>> 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 `x = [1 2 5 1]`

```
x =
     1   2   5   1
```

- A matrix `t = [1 2 3; 5 1 4; 3 2 -1]`

```
t =
     1     2     3
     5     1     4
     3     2    -1
```

- Transpose `y = x’`

```
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
```

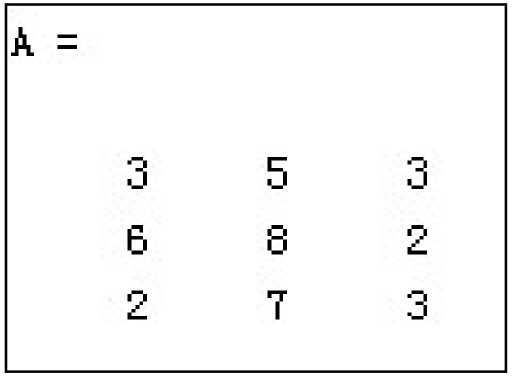
- rand(M,N) MxN matrix of uniformly distributed random numbers on (0,1)

```
x = rand(1,3)
x =
 0.9501  0.2311 0.6068
```

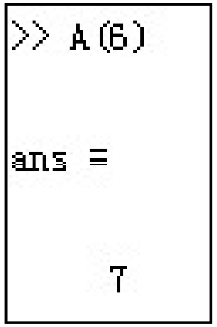
Matrix Index

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

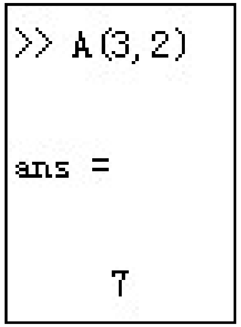
Given:



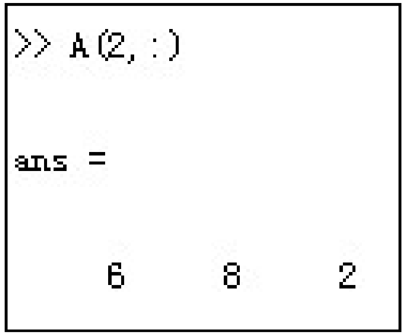
```
A =

     3     5     3
     6     8     2
     2     7     3
```



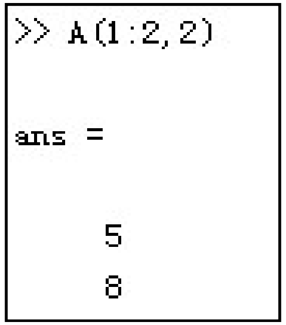
```
>> A(6)

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

- `x = [1 2], y = [4 5], z=[ 0 0]`

```
A = [x y]
       1   2   4   5
B = [x ; y]
      1 2
      4 5
```

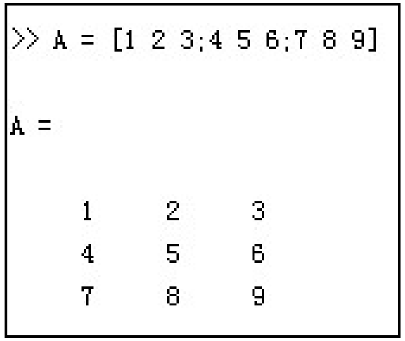
```
     C = [x y ;z]
Error:
??? Error using ==> vertcat CAT arguments dimensions are not consistent.
```

Operators (arithmetic)

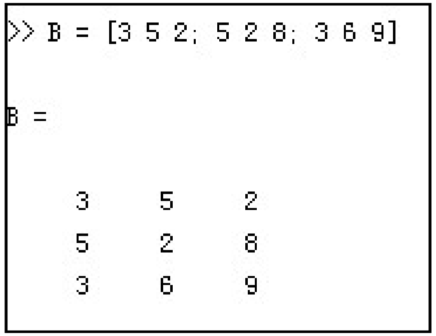
- + addition
- - subtraction
- * multiplication
- / division
- ^ power
- ‘ matrix transpose

Matrices Operations

Given A and B:

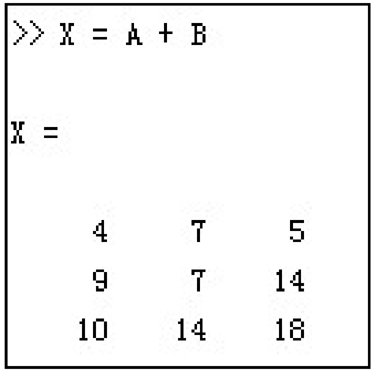


```
>> A = [1 2 3;4 5 6;7 8 9]

A =

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



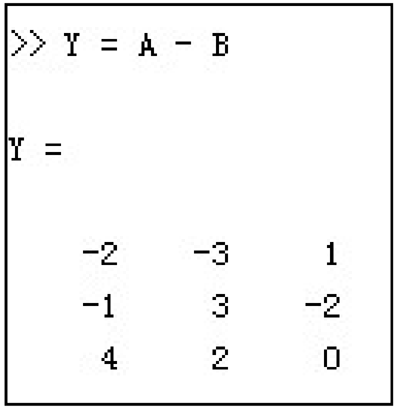
```
>> B = [3 5 2; 5 2 8; 3 6 9]

B =

     3     5     2
     5     2     8
     3     6     9
```

Addition



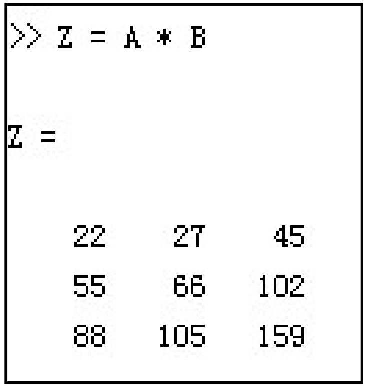
```
>> X = A + B

X =

     4     7     5
     9     7    14
    10    14    18
```

Subtraction



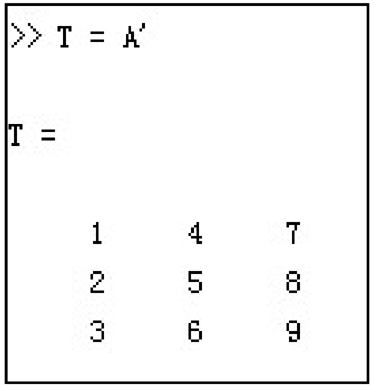
```
>> Y = A - B

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

```
A =

     3     2     1
     5     1     0
     2     1     7
```

```
B =

     1     3     1
     4     9     5
     2     7     2
```

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

Enter matrix B into the Matlab workspace

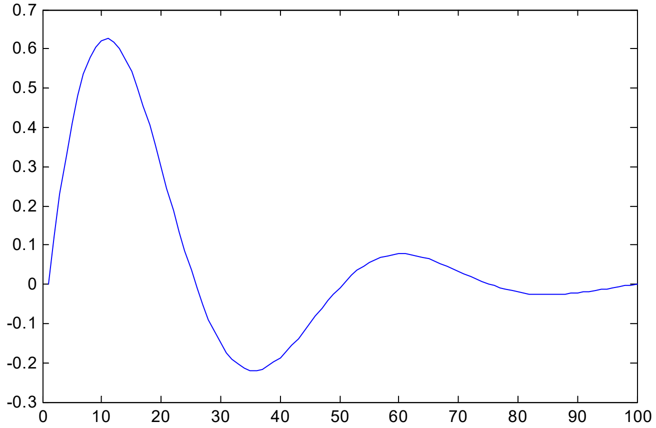
Create matrices A and B and try out the the matrix operators in this slide

Display Facilities

- **plot(.)**

Example:

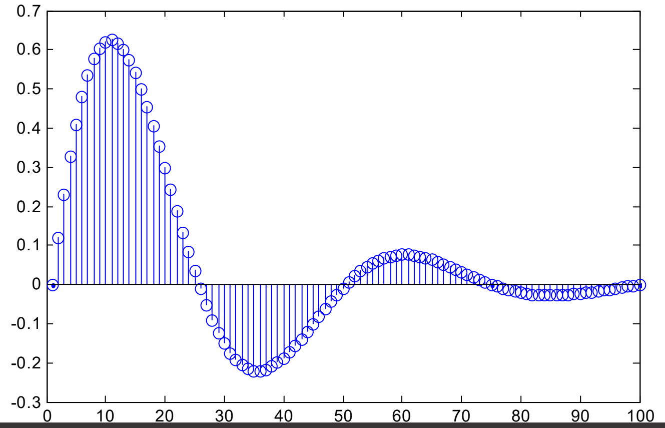
```
>>x=linspace(0,4*pi,100);
>>y=sin(x);
>>plot(y)
>>plot(x,y)
```



- **stem(.)**

Example:

```
>>stem(y)
>>stem(x,y)
```



Basic Task: Plot the function sin(x) between $0 \leq x \leq 4\pi$

- Create an x-array of 100 samples between 0 and 4π.

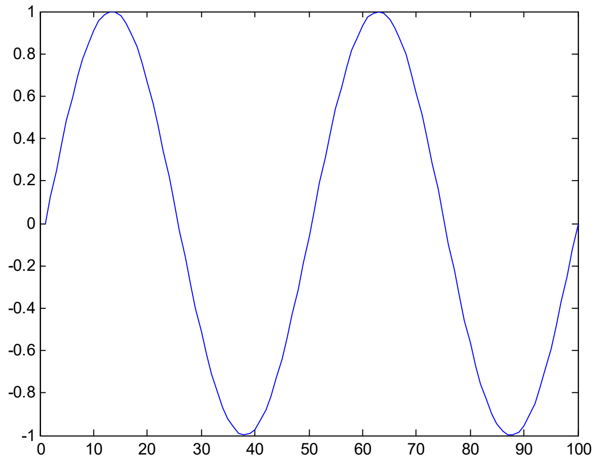
```
>>x=linspace(0,4*pi,100);
```

- Calculate sin(.) of the x-array

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

- Plot the y-array

```
>>plot(y)
```



Operators (relational, logical)

- == Equal to
- ~= 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
```

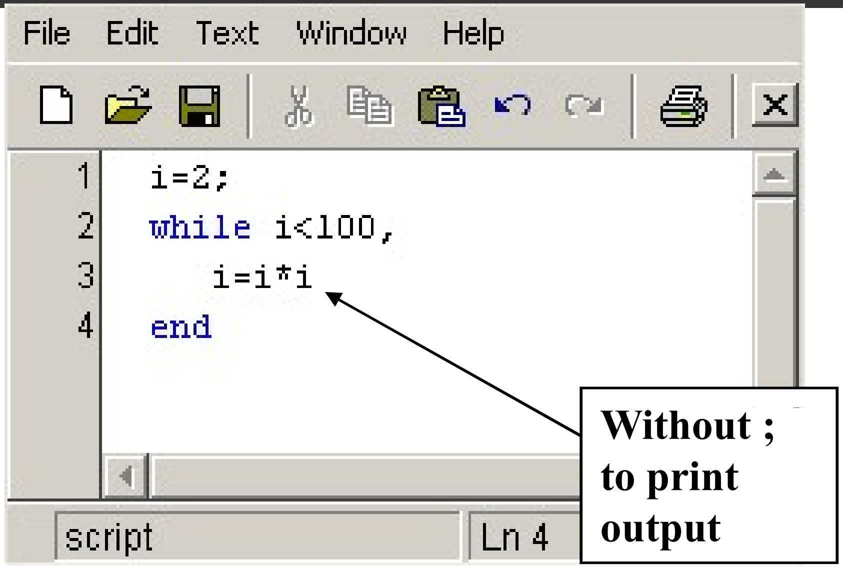
While Loop Syntax

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

Dummy Example

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

While Loop Syntax

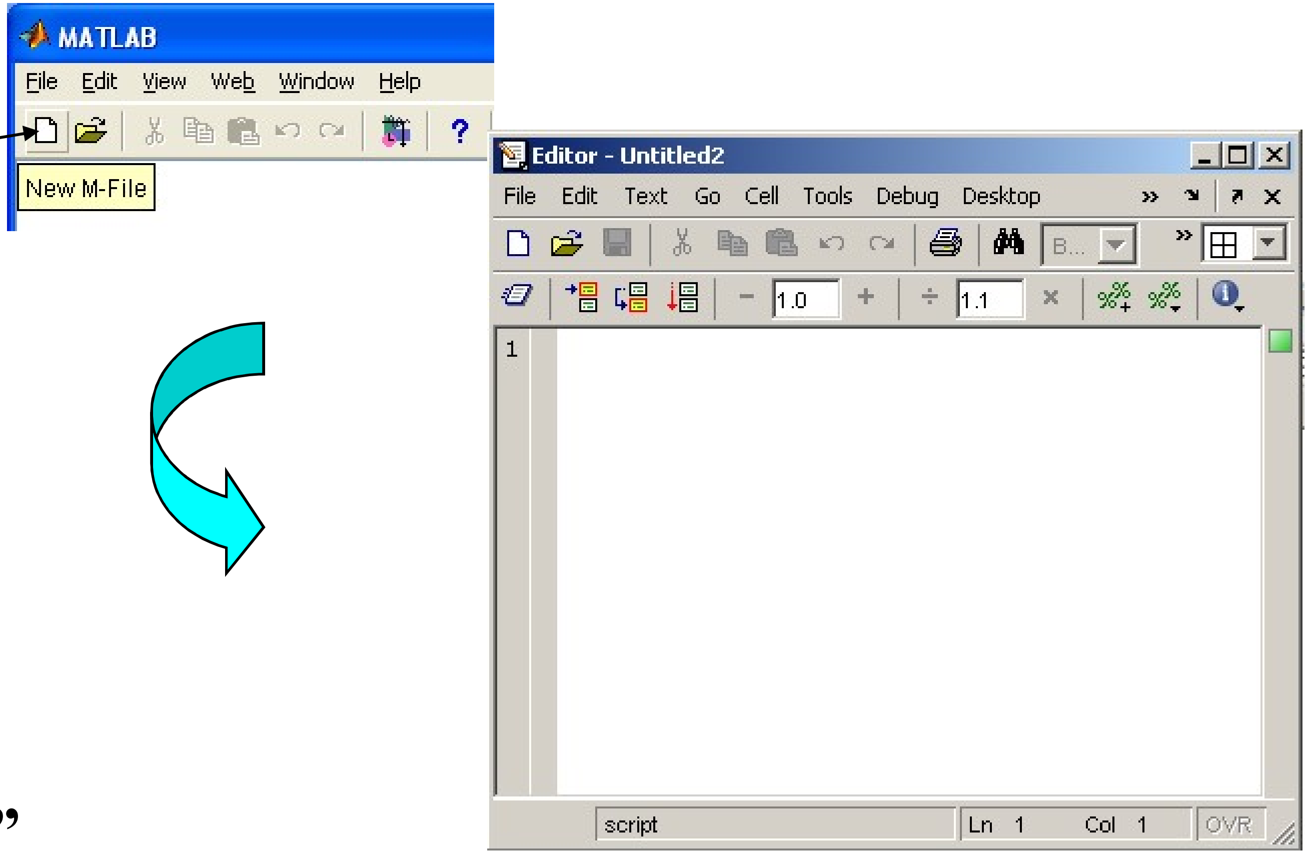


```
i=2;
while i<100,
   i=i*i
end
```

Without ; to print output

```
i =

   4

i =

  16

i =

 256
```

Use of M-File

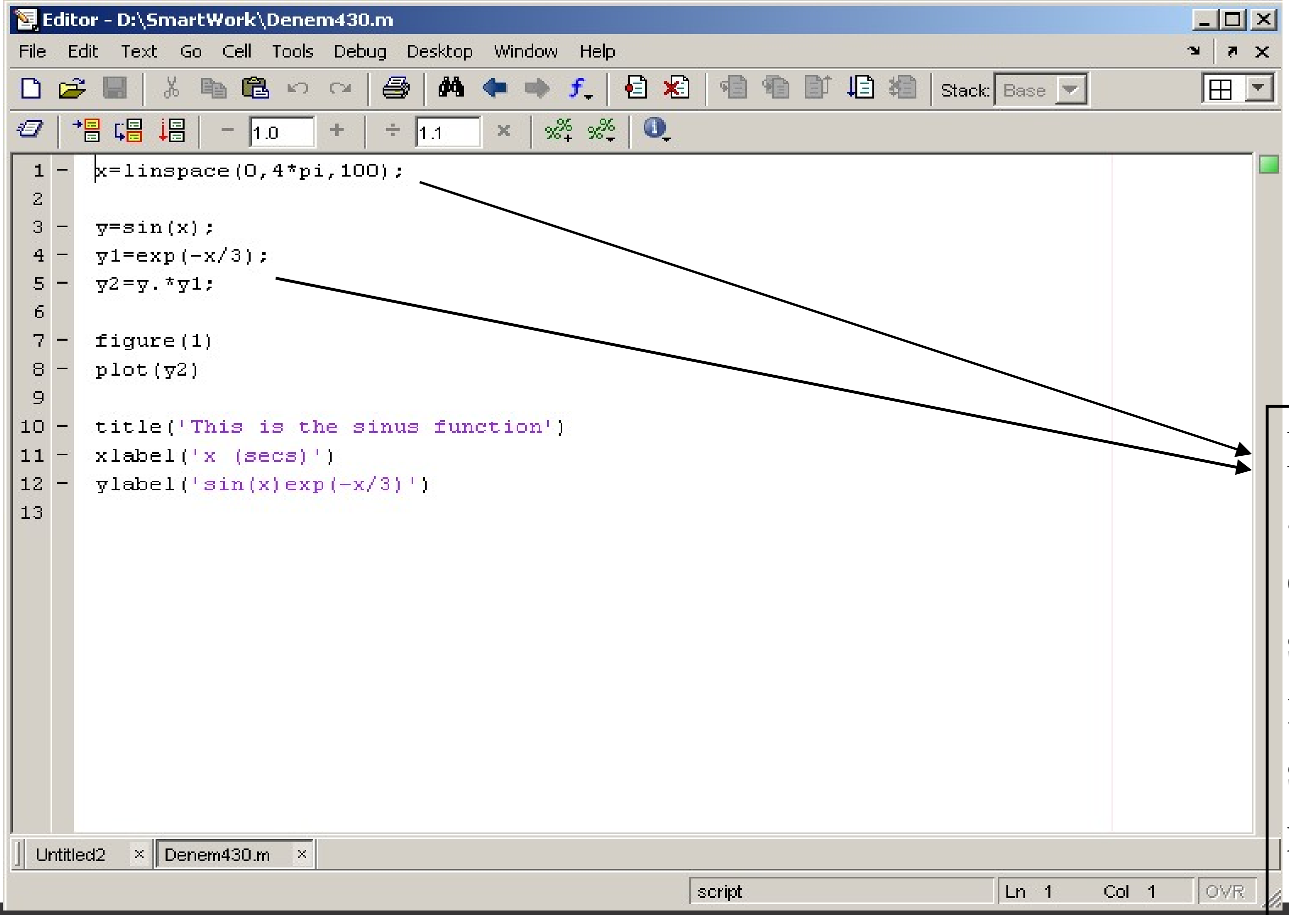

MATLAB
File Edit View Web Window Help
New M-File
Editor - Untitled2
File Edit Text Go Cell Tools Debug Desktop
B...
1.0
1.1
1
script
Ln 1
Col 1
OVR

Click to create a new M-File

- Extension “.m”
- A text file containing script or function or program to run

Use of M-File

Save file as *Denem430*.m



Editor - D:\SmartWork\Denem430.m
File Edit Text Go Cell Tools Debug Desktop Window Help
Stack: Base
1.0
1.1
Untitled2
Denem430.m
script
Ln 1
Col 1
OVR

```
x=linspace(0,4*pi,100);

y=sin(x);
y1=exp(-x/3);
y2=y.*y1;

figure(1)
plot(y2)

title('This is the sinus function')
xlabel('x (secs)')
ylabel('sin(x)exp(-x/3)')
```

If you include ";" at the end of each statement, result will not be shown immediately

Writing User Defined Functions

- Functions are **m-files** which can be executed by **specifying some inputs and supply some desired outputs.**
- The code telling the Matlab that an m-file is actually a function is

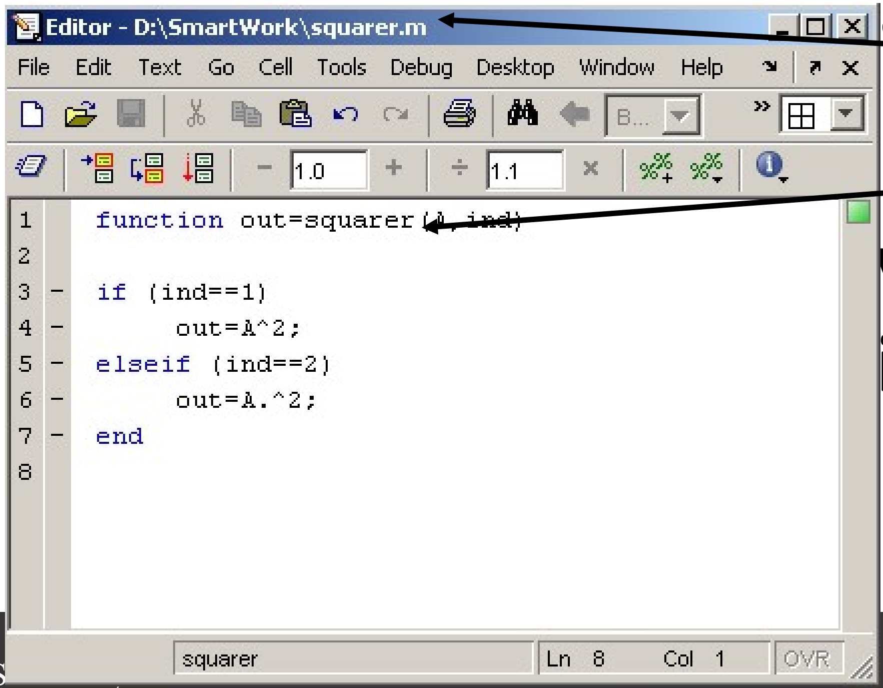
```
function out1=functionname(in1)
function out1=functionname(in1,in2,in3)
function [out1,out2]=functionname(in1,in2)
```

- You should write this command **at the beginning of the m-file** and save the m-file with a file name **same as the function name**

Writing User Defined Functions

- Examples
  - Write a function : out=squarer (A, ind)
    - Which takes the square of the input
      tor is equal

    - y element
      ix if the input



Editor - D:\SmartWork\squarer.m
File Edit Text Go Cell Tools Debug Desktop Window Help
B...
1.0
1.1
squarer
Ln 8 Col 1
OVR

```
function out=squarer(A,ind)

if (ind==1)
      out=A^2;
elseif (ind==2)
      out=A.^2;
end
```

Same Name

Useful Commands

- The two commands used most by Matlab users are

```
>>help functionName
```

```
>>lookfor keyWord
```

Debugging

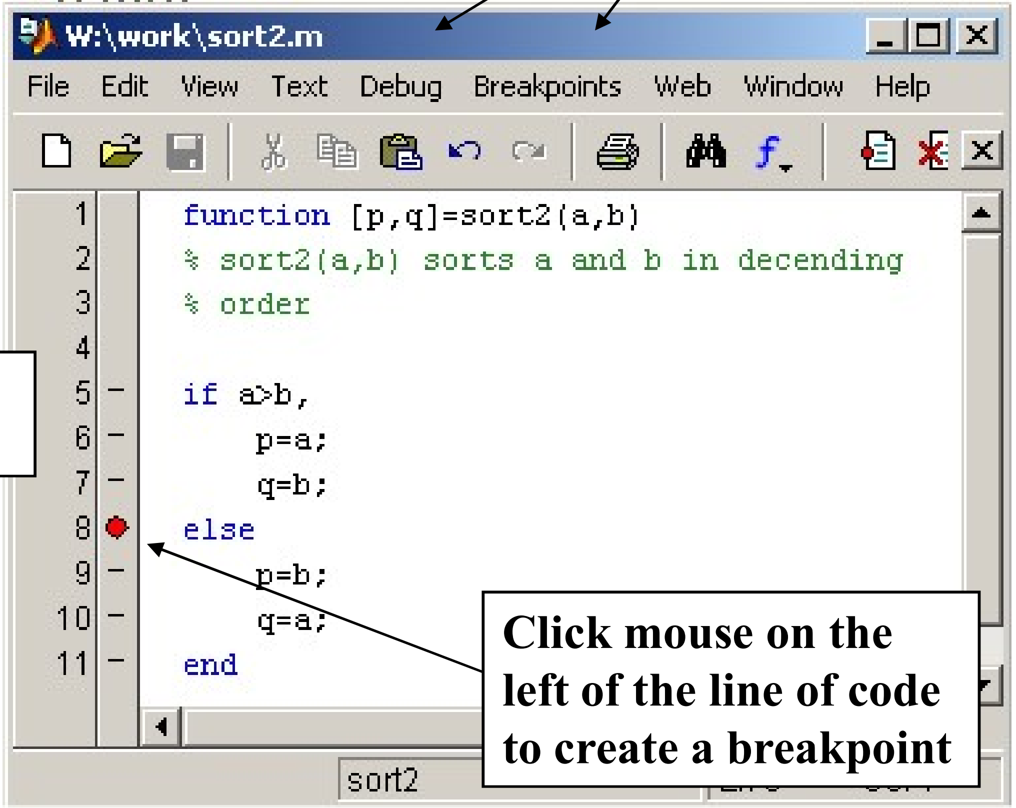
- **Set breakpoints to stop the execution of code**

```
>> [i j]=sort2(2,4)
K>>
K>> whos
 Name      Size                Bytes  Class
  a          1x1                 8  double array
  b          1x1                 8  double array
Grand total is 2 elements using 16 bytes
K>> a
a =
    2
K>> return
i =

    4
j =
    2
```

local function workspace

exit debug mode

Debug menus



```
function [p,q]=sort2(a,b)
% sort2(a,b) sorts a and b in decending
% order

if a>b,
    p=a;
    q=b;
else
    p=b;
    q=a;
end
```

Click mouse on the left of the line of code to create a breakpoint