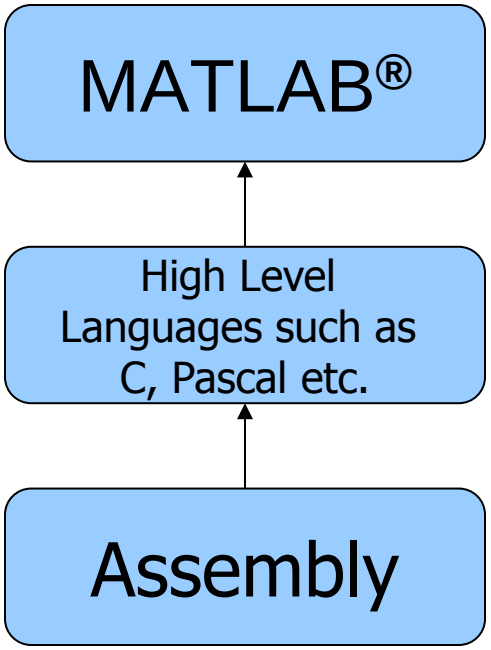
Introduction to MATLAB®

Outline:

- What is MATLAB®?
- MATLAB® Screen
- Variables, array, matrix, indexing
- Operators (Arithmetic, relational, logical)
- Display Facilities
- Flow Control
- Using of M-File
- Writing User Defined Functions
- Conclusion

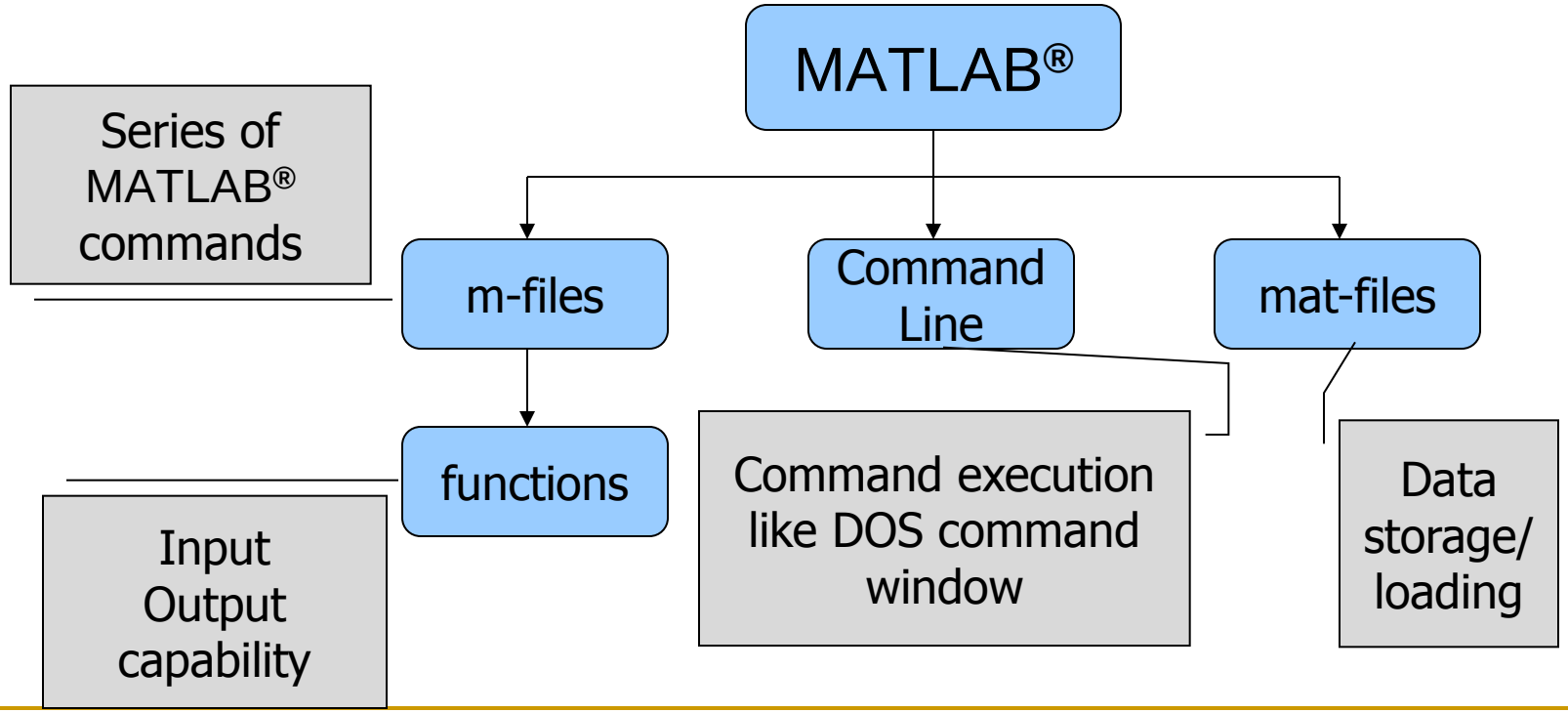
What is MATLAB®?

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



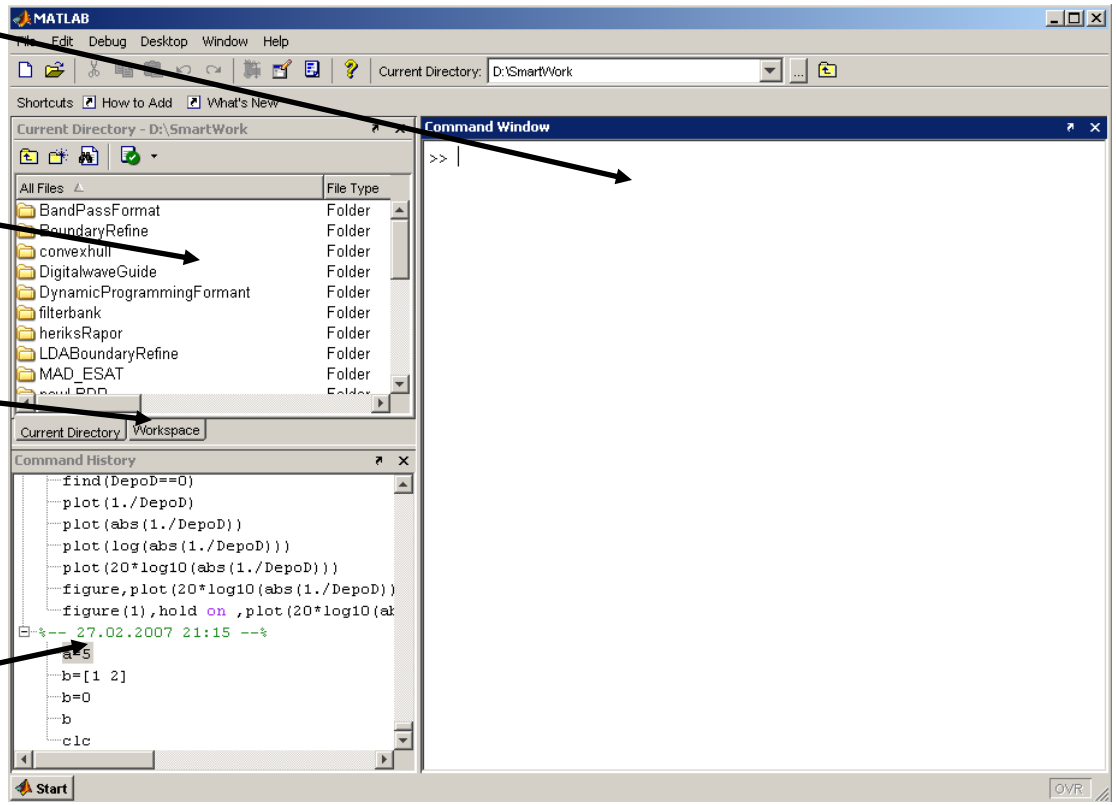
What are we interested in?

- MATLAB® is too broad for our purposes in this course.
- The features that will be enough for you



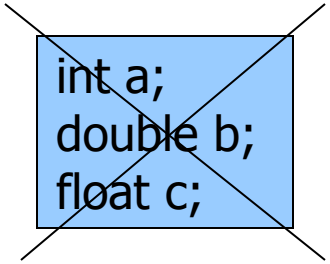
MATLAB® Screen

- 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



Variables

- No need for types. i.e.,



- 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

Array, Matrix

- a vector `x = [1 2 5 1]`

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

- a matrix `x = [1 2 3; 5 1 4; 3 2 -1]`

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

- transpose `y = x'`

```
y =
     1
     2
     5
     1
```

Long Array, Matrix

```
e = 1:10
```

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

```
r = 2:-0.5:-1
```

```
r =
    2.0   1.5   1.0   0.5   0.0  -0.5  -1.0
```

```
t = [1:4; 5:8]
```

```
t =
     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
```

- randn(M,N) MxN matrix of normally distributed random numbers with *N*(0,1)

```
x = randn(1,3)
x =
 -0.0029   0.9199   0.1498
```

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
‘ complex conjugate

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

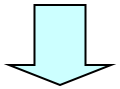
./ element-by-element division

.^ element-by-element power

.' array transpose (without conjugation)

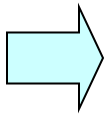
The use of "." – "Element" Operation

```
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 * ==> Inputs must be a scalar and a square matrix..
B=x*y
Error:
Error using ^ ==> Inner matrix dimensions must agree.
```

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

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

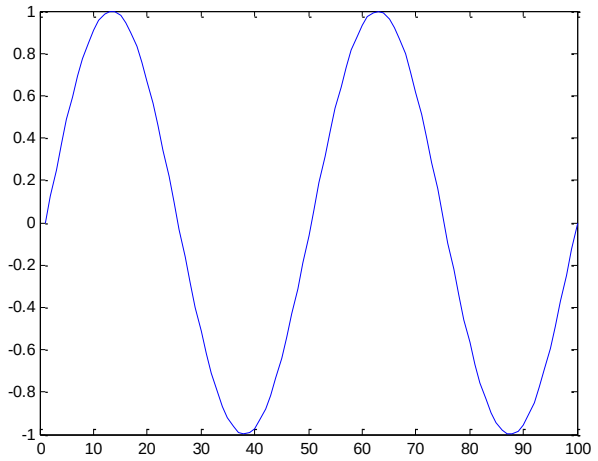
```
>>x=linspace(0,4*pi,100);
```

- Calculate sin(.) of the x-array

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

- Plot the y-array

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



Plot the function $e^{-x/3}\sin(x)$ between $0 \le x \le 4\pi$

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

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

- Calculate sin(.) of the x-array

```
>>f1=sin(x);
```

- Calculate $e^{-x/3}$ of the x-array

```
>>f2=exp(-x/3);
```

- Multiply the arrays f1 and f2

```
>>y=f1.*f2;
```

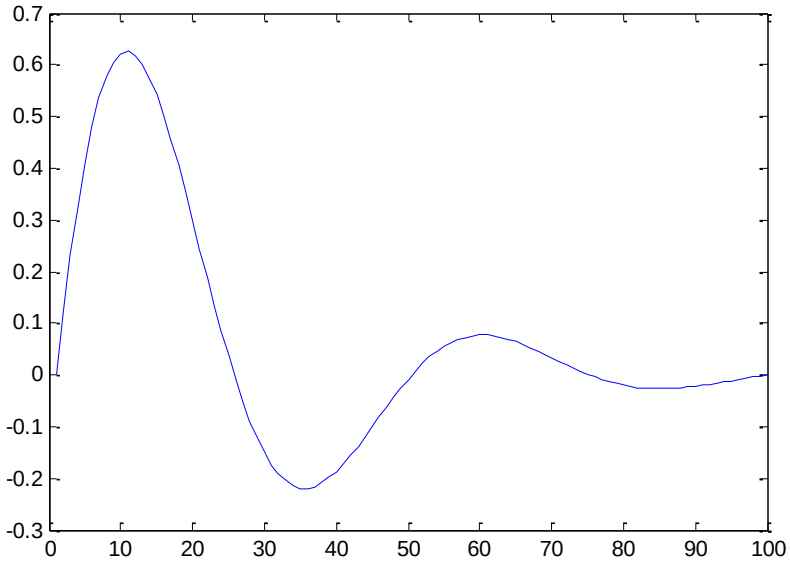
Plot the function $e^{-x/3}\sin(x)$ between $0 \leq x \leq 4\pi$

- Multiply the arrays f1 and f2 correctly!

```
>>y=f1.*f2;
```

- Plot the y-array

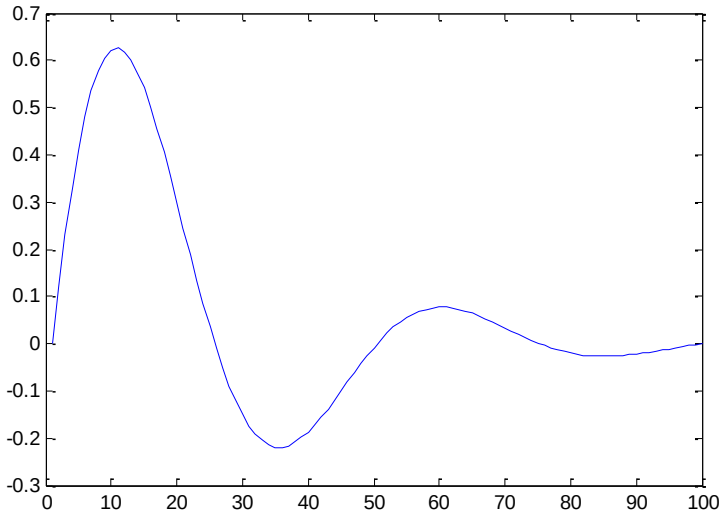
```
>>plot(y)
```



Display Facilities

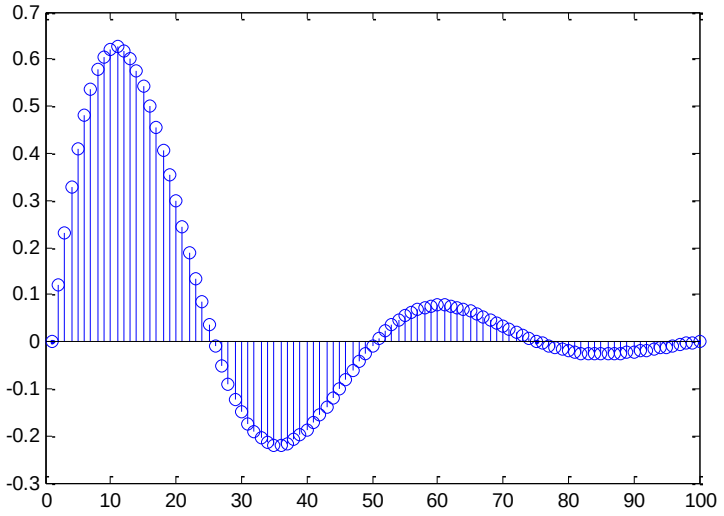
- plot(.)

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



- stem(.)

Example:
>>stem(y)
>>stem(x,y)



Display Facilities

- title(.)

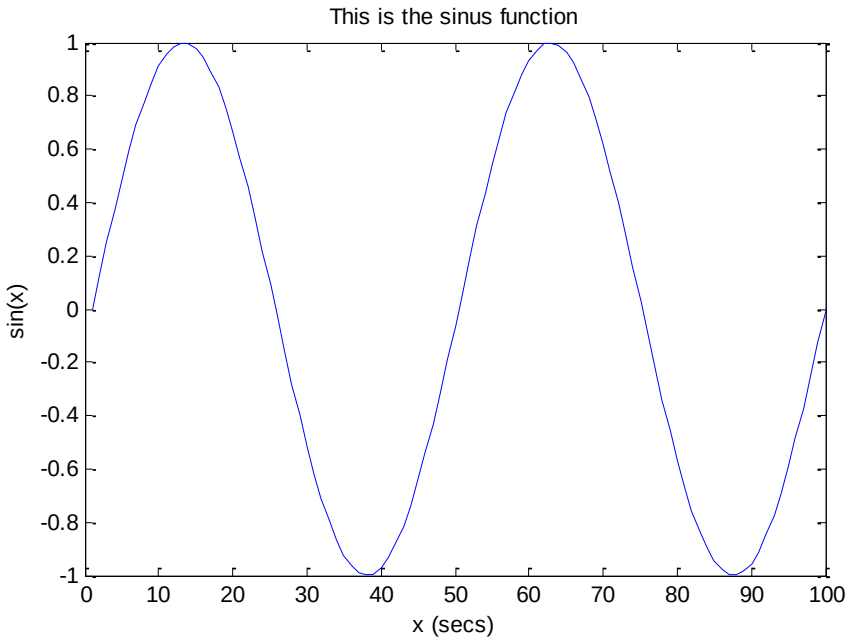
```
>>title('This is the sinus function')
```

- xlabel(.)

```
>>xlabel('x (secs)')
```

- ylabel(.)

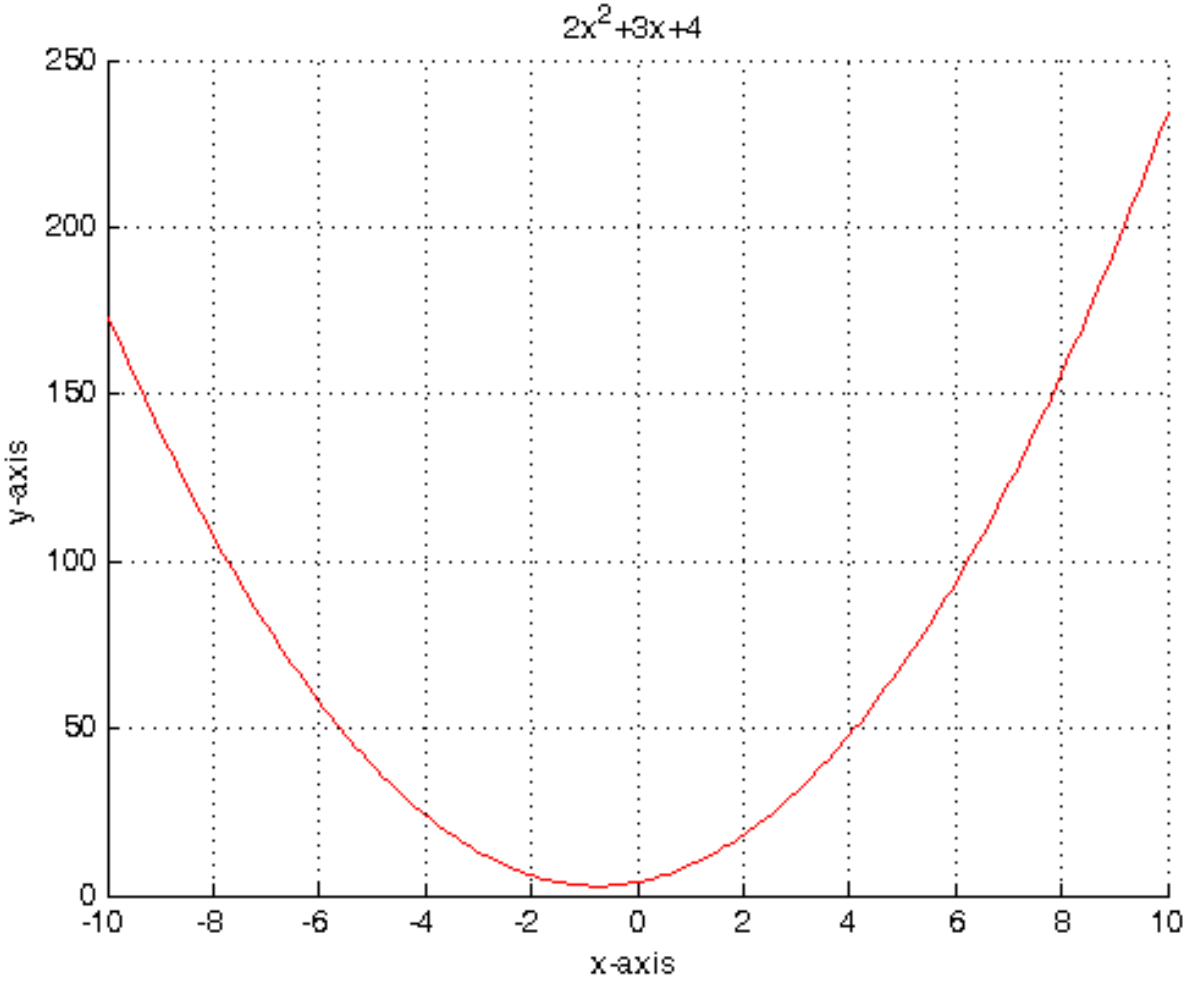
```
>>ylabel('sin(x)')
```



A Plot Example

```
disp('Please, enter inputs for the polynomial:');
disp(' a*(x.^2) + (b*x) +c ');
a= input('enter "a" \n');
b= input('enter "b" \n');
c= input ('enter "c" \n');
x= -10:0.1:10;
y= a*(x.^2) + (b.*x) + c;
plot(x,y,'r-')
xlabel('x-axis')
ylabel('y-axis')
title([num2str(a),'x^2+',num2str(b),'x+',num2str(c)]);
grid on
box off
```

A Plot Example



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
- ….

Control Structures

- If Statement Syntax

```
if (Condition_1)
        MATLAB® Commands
elseif (Condition_2)
        MATLAB® Commands
elseif (Condition_3)
        MATLAB® Commands
else
        MATLAB® Commands
end
```

Some Dummy Examples

```
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
```

Control Structures

- For Loop syntax

```
for i=Index_Array
    MATLAB® Commands
end
```

Some Dummy Examples

```
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
```

Control Structures

- While Loop Syntax

```
while (condition)
    MATLAB® Commands
end
```

Dummy Example

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

Examples

```
% EX-1

t=[-5,5,-4,4,-3,3,-2,2,-1,1];
for k=1:length(t)
   if t(k)<0
      continue
   end
   t(k)= t(k)^2;
end
disp(t)
```

```
% EX-2

counter = 1; init = 4;
while counter < 11
    cnum= [counter init];
    disp(cnum)
    init = init+5;
    counter = counter + 1;
    if counter >= 11
        break
    end
end
```

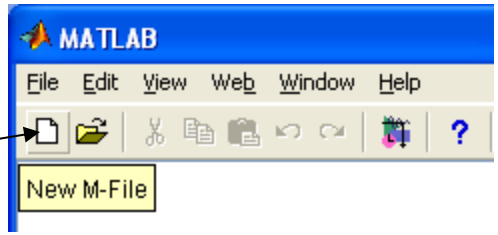
Examples

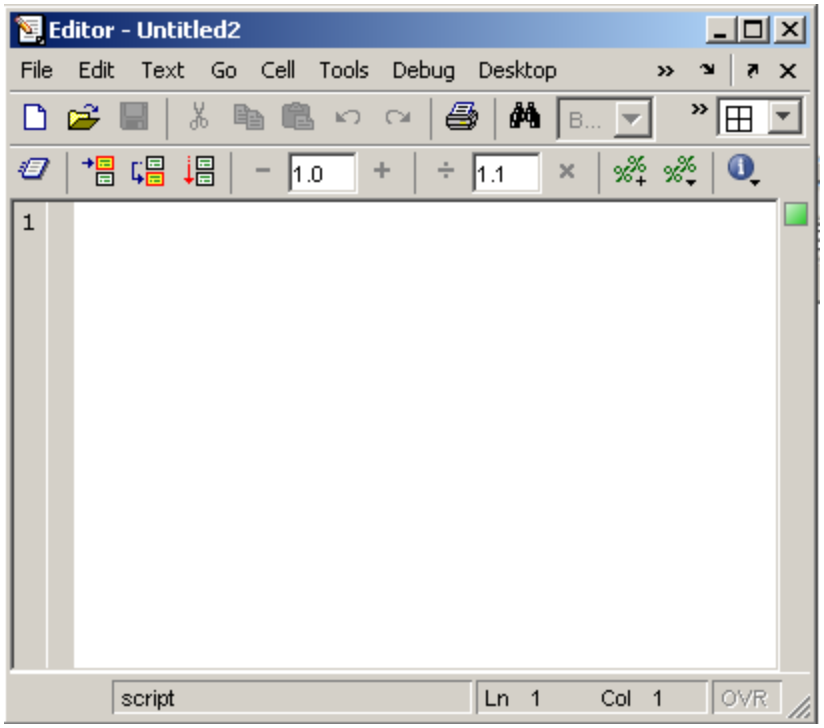
```
% EX-3

a=input('Enter a number:\n');
if a<10
x=a*3;
elseif a<100
    x=a+3;
else
    x=a*10;
end
disp('Output');
disp(x);
```

```
% EX-4

angle=input('Angle Value');
switch fix(angle/90)
    case 0
        disp('Quadrant I')
    case 1
        disp('Quadrant II')
    case 2
        disp('Quadrant III')
    case 3
        disp('Quadrant IV')
    otherwise
        disp('Input should be 0-360')
end
```

Use of M-File

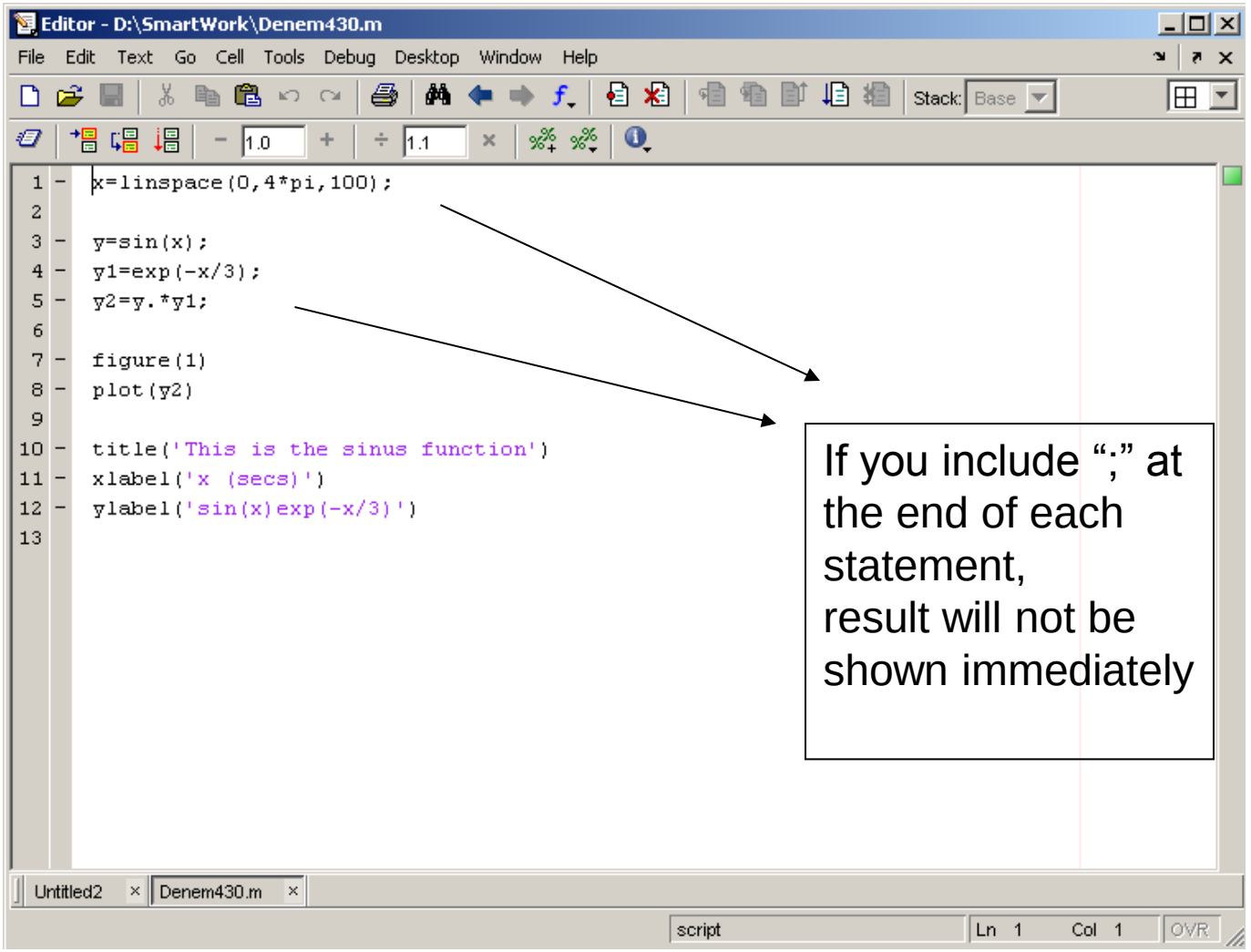
Click to create
a new M-File





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

Use of M-File


Editor - D:\SmartWork\Denem430.m
File Edit Text Go Cell Tools Debug Desktop Window Help
Stack: Base
1.0
1.1
1 x=linspace(0,4*pi,100);
2
3 y=sin(x);
4 y1=exp(-x/3);
5 y2=y.*y1;
6
7 figure(1)
8 plot(y2)
9
10 title('This is the sinus function')
11 xlabel('x (secs)')
12 ylabel('sin(x)exp(-x/3)')
13
Untitled2
Denem430.m
script
Ln 1 Col 1
OVR
If you include ";" at the end of each statement, result will not be shown immediately

Writing User Defined Functions

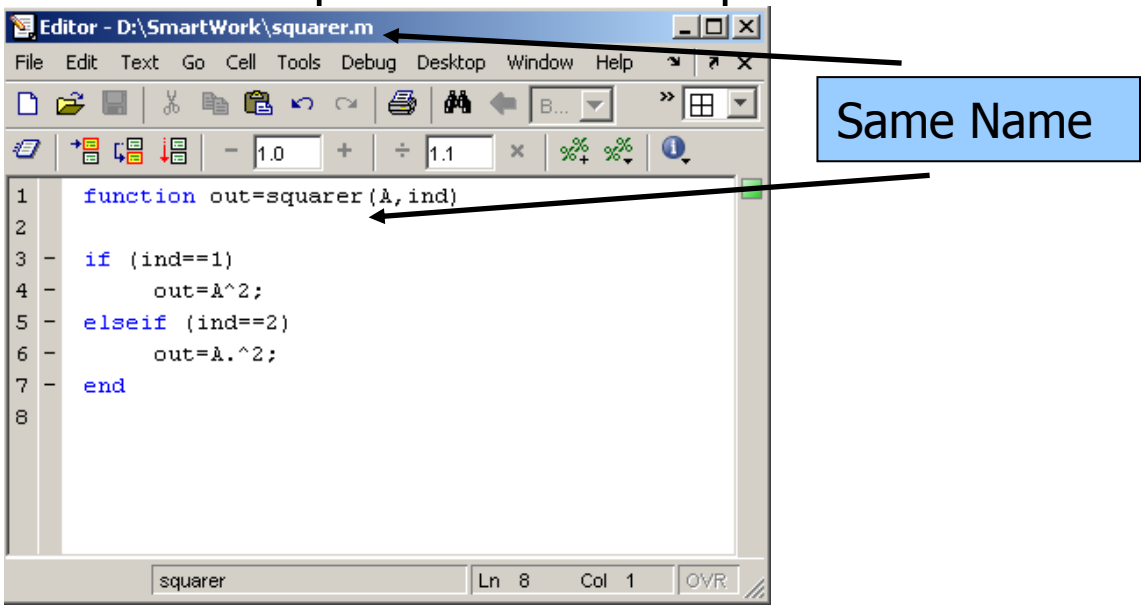
- Functions are m-files which can be executed by specifying some inputs and supply some desired outputs.
- The code telling the MATLAB® that an m-file is actually a function is

```
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 you should save the m-file with a file name same as the function name

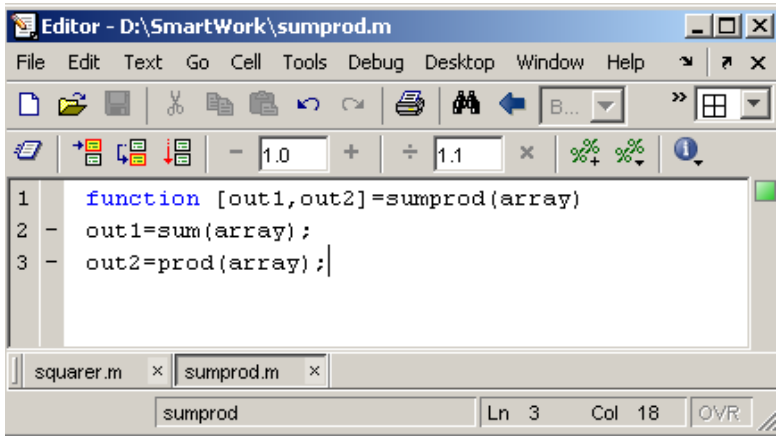
Writing User Defined Functions

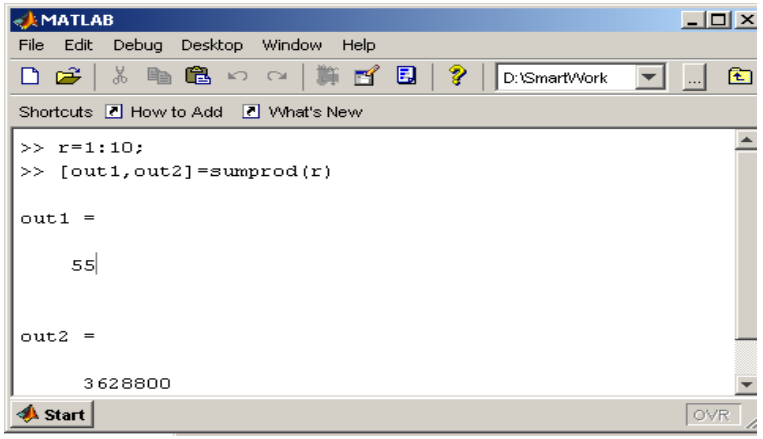
- Examples
  - Write a function : out=squarer (A, ind)
    - Which takes the square of the input matrix if the input indicator is equal to 1
    - And takes the element by element square of the input matrix if the input indicator is equal to 2



Writing User Defined Functions

- Another function which takes an input array and returns the sum and product of its elements as outputs.
- The function sumprod(.) can be called from command window or an m-file as






```
%Sumprod Example
function [out1, out2] = sumprod(array)
out1= sum(array);
out2= prod(array);
end
```

```
% Factorial Example
function a=fact(n)
a=1;
for i=1:n
    a = a* i;
end
disp(a);
```

Notes:

- "%" is the neglect sign for MATLAB® (equivalent of "//" in C). Anything after it on the same line is neglected by MATLAB® compiler.
- Sometimes slowing down the execution is done deliberately for observation purposes. You can use the command "pause" for this purpose

```
pause %wait until any key
pause(3) %wait 3 seconds
```

Useful Commands

- The three commands mostly used for help by MATLAB® users are

```
>>help functionname
```

```
>>doc functionname
```

```
>>lookfor keyword
```

- And off course other helper is..
  - Google search

Questions

- ?

Thank you..