MATLAB

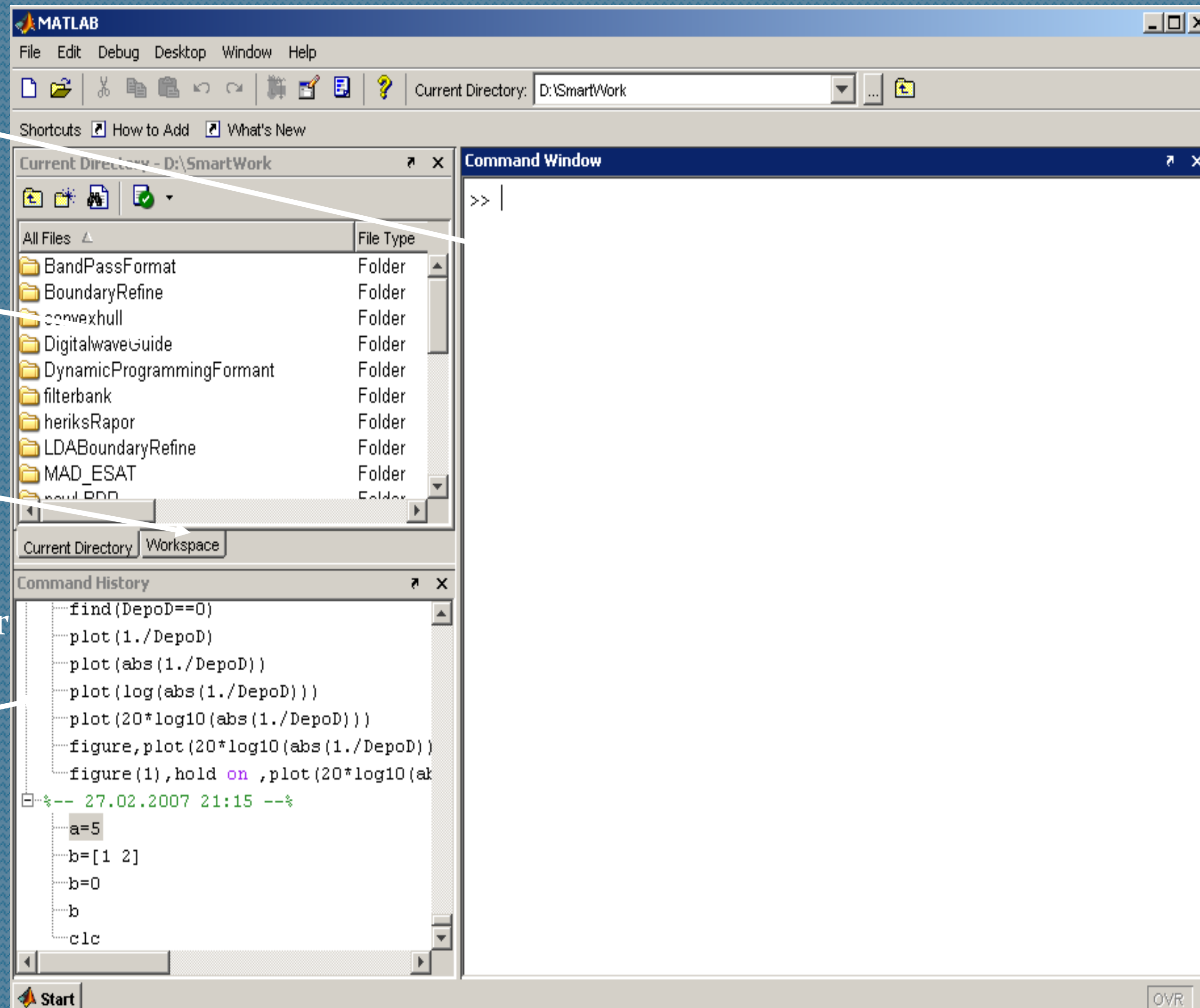
POOJA SHARMA

Dept. of Physics and Astrophysics
University of Delhi

- MATLAB stands for Matrix Laboratory
- Works with WIN/Linux/Mac etc.
- Not free
- The language is based on matrices.
- The variables do not have to be specified as float/ int etc.
- Case - sensitive

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 Files (.m)

- Use predefined functions or write your own functions
- There are two kinds of files : Script files ‘M-files’ and Function files.
- Reside on the `current directory` or the `search path`
  - add with File/Set Path
- Use the Editor/Debugger to edit, run

Mathematical Functions

- Elementary functions (sin, cos, sqrt, abs, exp, log10, round)
  - type `help elfun`
- Advanced functions (bessel, beta, gamma, erf)
- Other in-built functions (sum, diff, int, eig, eigvec, abs, etc.)

Operators (relational, logical)

==	equal
~=	not equal
<	less than
<=	less than or equal
>	greater than
>=	greater than or equal

&	AND
|	OR
~	NOT

pi	3.14159265…
j	imaginary unit, $\sqrt{-1}$
i	same as j

Operators (arithmetic)

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

.*	element-by-element mult
./	element-by-element div
.^	element-by-element power
.‘	transpose

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
```

Simple Matrix Commands

- 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
```

More vector commands

- ```
  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
```

- ```
  B = [1:4; 5:8]
  ```

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

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.

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
```

Matrix operations (Element by Element)

.* element-by-element multiplication

./ element-by-element division

.^ element-by-element power

Examples: Element by element Operations

```
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
Erorr:
 ??? Error using ==> mpower  Matrix must be square.
B=x*y
Erorr:
??? Error using ==> mtimes Inner matrix dimensions must agree.
```

Basic Plotting

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

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

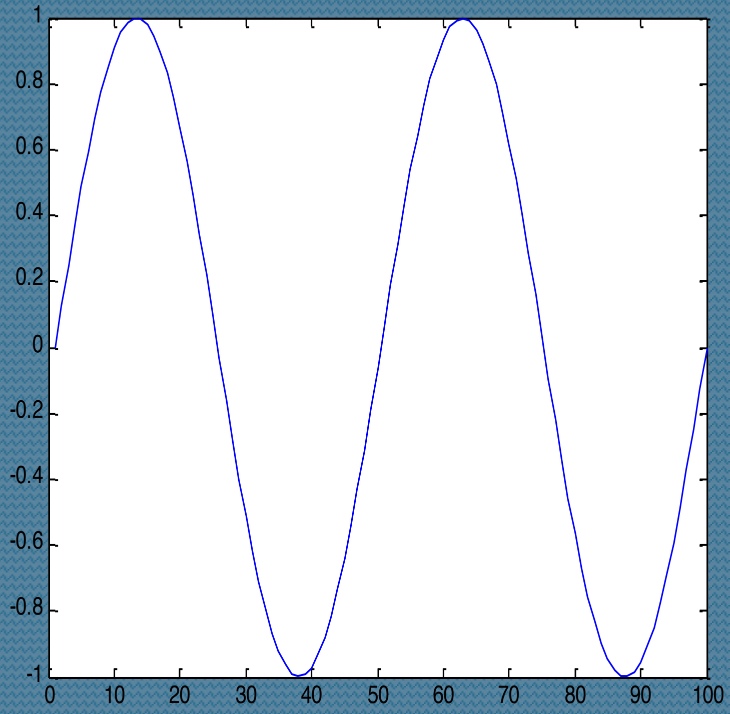
```
>>x=linspace(0,4*pi,100);
```

- Calculate sin(.) of the x-array

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

- Plot the y-array

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



Plotting the function $e^{-x/3}\sin(x)$ 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);
```

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

```
>>y1=exp(-x/3);
```

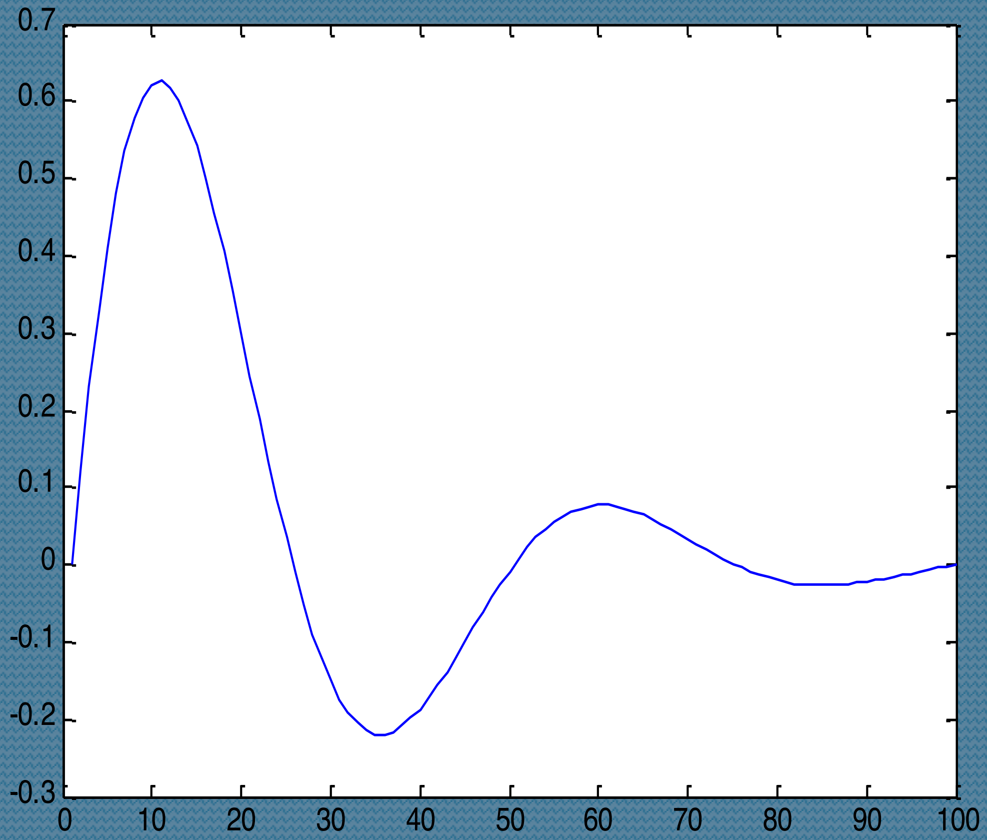
- Multiply the arrays y and y1

```
>>y2=y*y1;
```

cont..

Plot the y2-array

```
>>plot(y2)
```



Title, X-Label, Y-label

- title(.)

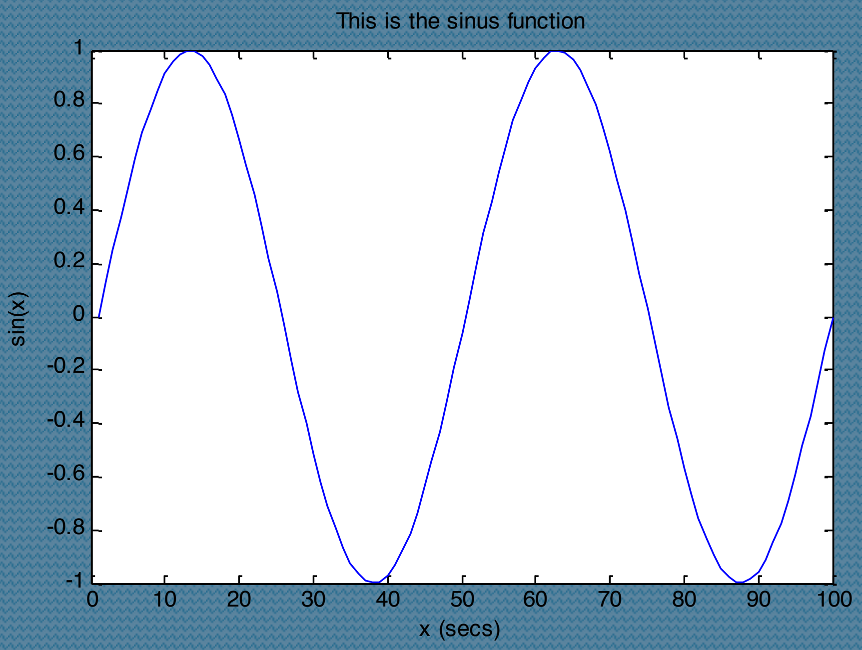
```
>>title('This is the sinus function')
```

- xlabel(.)

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

- ylabel(.)

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



Symbolic Calculations

The symbolic calculations can be done very easily in MATLAB using very few commands.

To manipulate a symbolic variable, create an object of type SYM, such as

```
>> x = sym('x')
```

Shortcut for constructing symbolic objects such as:

```
>> syms arg1 arg2 ... Real
```

is short-hand notation for:

```
>>arg1 = sym('arg1','real');
>>arg2 = sym('arg2','real');
```

Solving symbolic equations

- >> syms a b c x
- >>x = solve(a*x^2 + b*x + c);

Gives

x =

-1/2*(b-(b^2-4*a*c)^(1/2))/a
-1/2*(b+(b^2-4*a*c)^(1/2))/a

System of two equations

A system of two quadratic equations in two unknowns produces solution vectors.

```
>> [x,y] = solve('x^2 + x*y + y = 3','x^2 - 4*x + 3 = 0')

x =

 1
 3

 y =

   1
 -3/2
```

Solving symbolic differential equations

Similar notation, with "D" denoting differentiation, is used for ordinary differential equations by the "dsolve" function.

```
>>y = dsolve('Dy = -a*y')

y =

C1*exp(-a*t)
```

Specify an initial condition.y = dsolve('Dy = -a*y','y(0) = 1')

```
>>y = dsolve('Dy = -a*y','y(0) = 1')
 y =

exp(-a*t)
```

The second derivative is denoted by "D2".

```
>> y = dsolve('D2y = -a^2*y', 'y(0) = 1, Dy(pi/a) = 0')

y =

cos(a*t)
```

Differentiate symbolic expression:

```
>>diff(S,'v',n)
```

It differentiates a symbolic expression S with respect to its free variable v, n times.

Example:

```
>>syms x t
```

Then

```
>>diff(sin(x^2))
```

Returns

```
>>2*cos(x^2)*x
```

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_

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

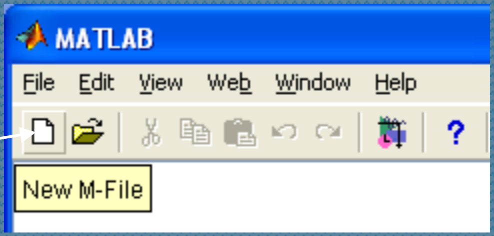
```
while (condition)
   Matlab Commands
end
```

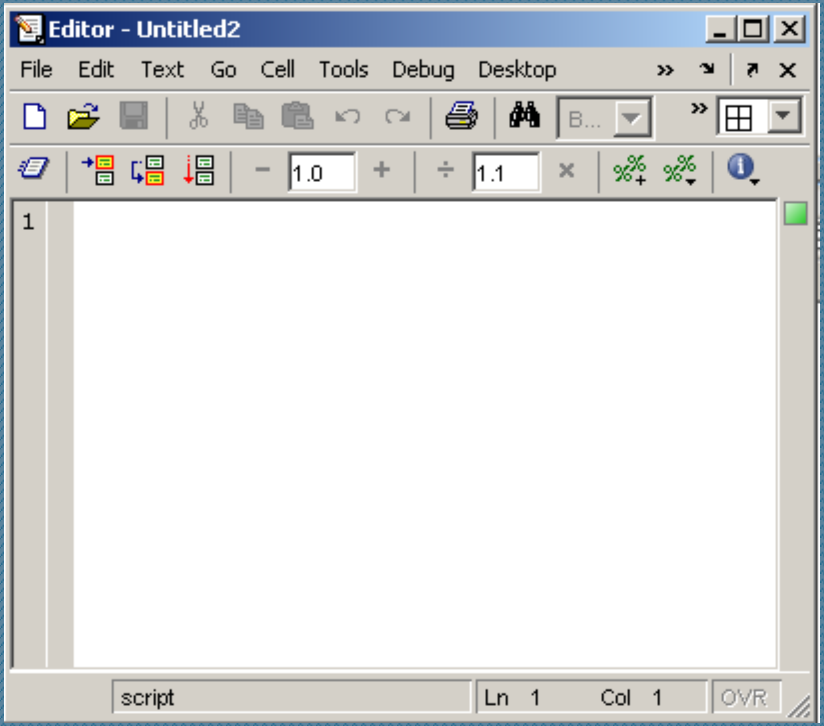
Dummy Example

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

M-File

Click to create a new M-File

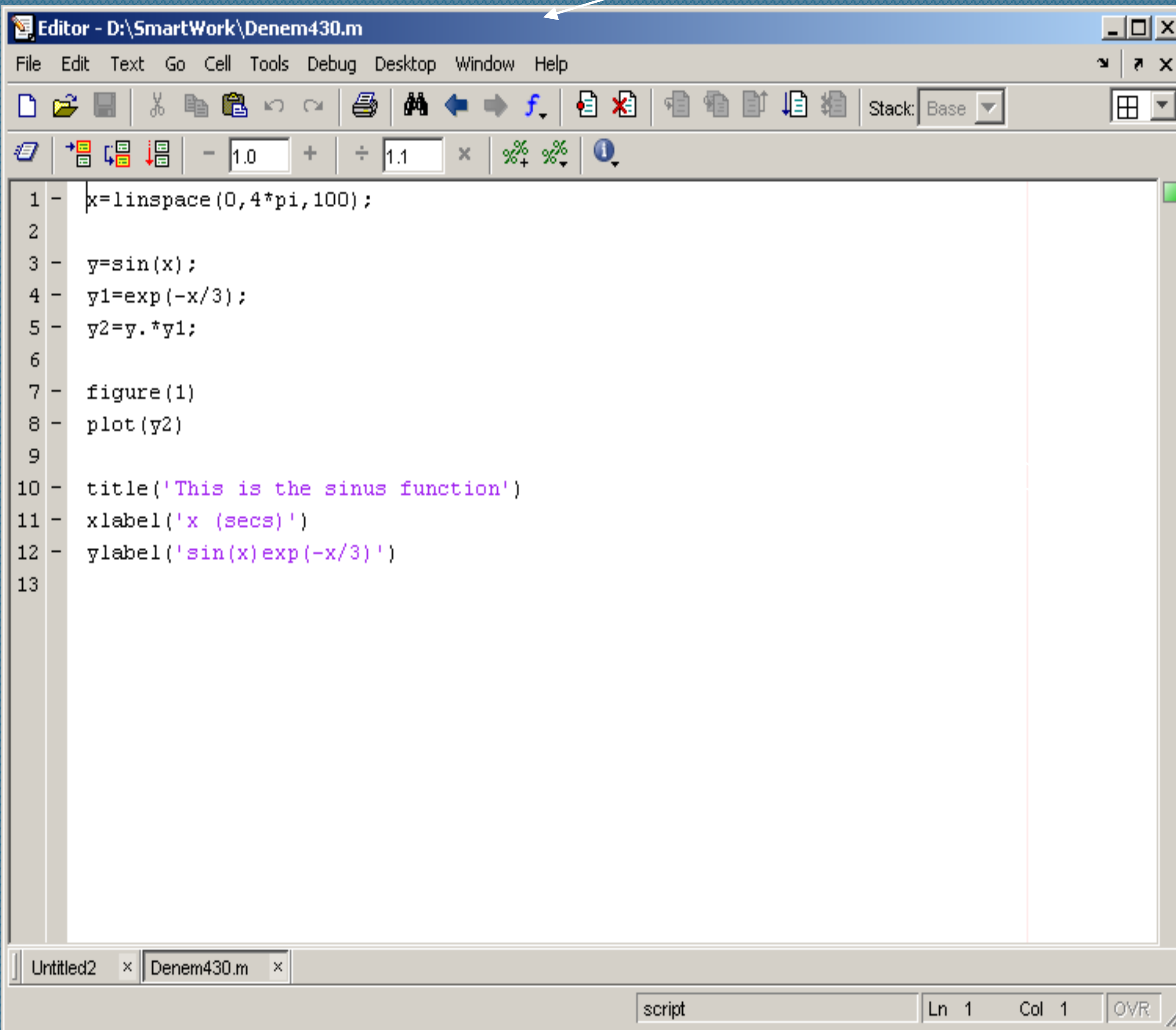




- Extension ".m"
- A text file containing script or function or program to run

Use of M-File

Save file as *Denem430*.m



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

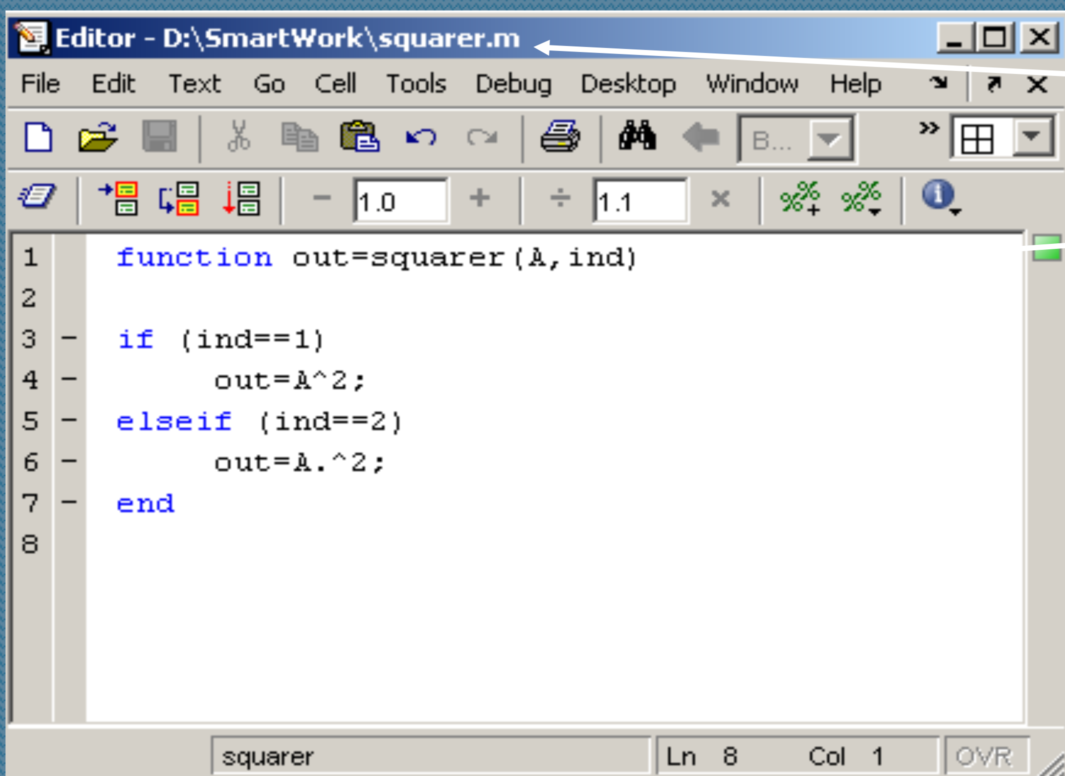
Writing User Defined Functions

- 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**

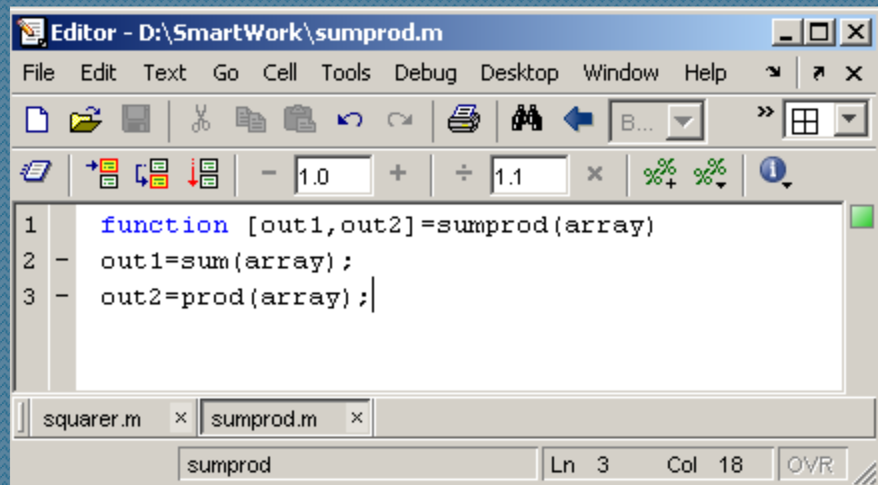
- Examples
  - Write a function : out=squarer (A, ind)
    - Which takes the square of the input matrix if the input indicator is equal to 1
    - And takes the element by element square of the input matrix if the input indicator is equal to 2



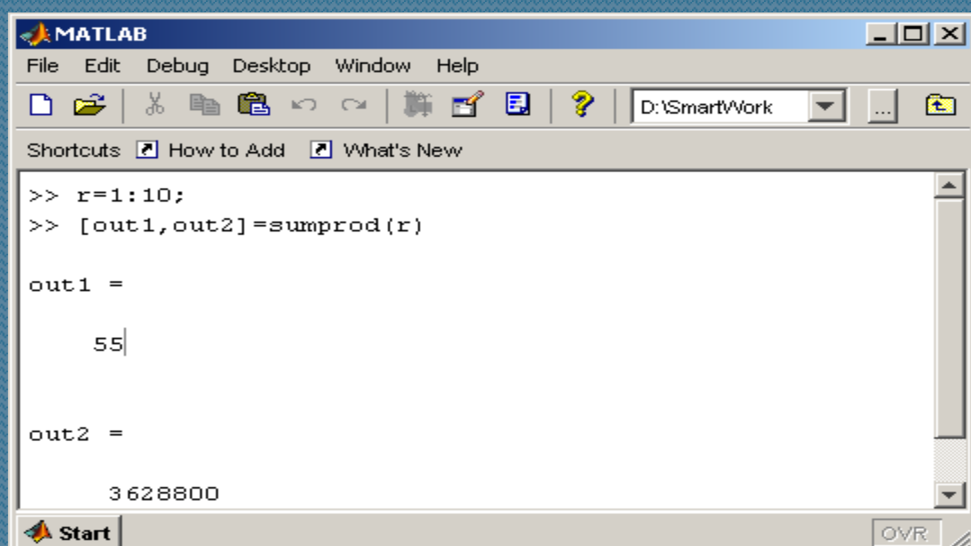
Same Name

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



- “%” is the neglect sign for Matlab (equaivalent 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 two commands used most by Matlab users are

```
>>help functionname
```

```
>>lookfor keyword
```

Help in MATLAB

- Help->Demo-> Put the topic in Search For bar

This is the in-built help provided in MATLAB.