INTRODUCTION

TO

MATLAB

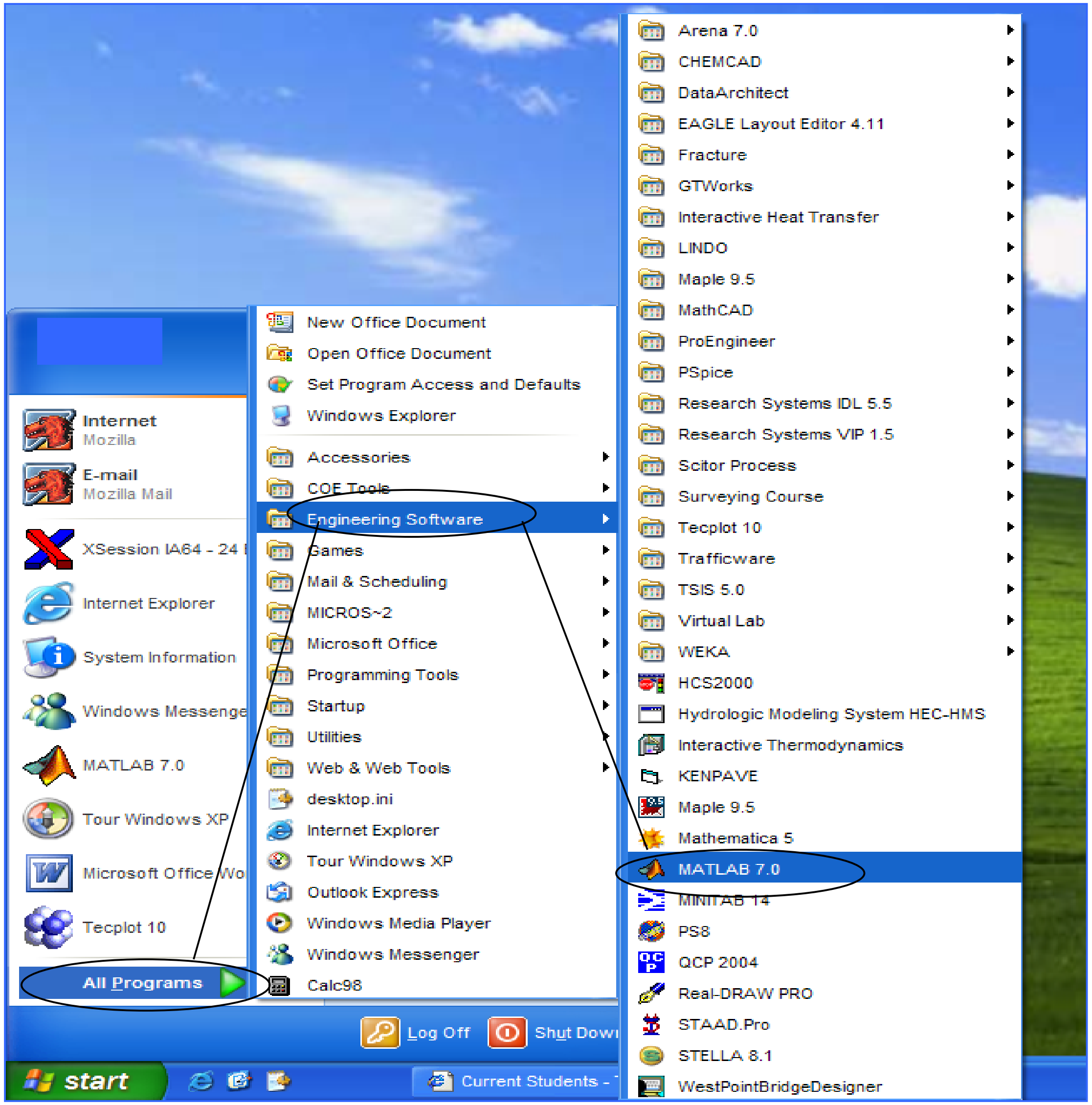
- Introduction to MATLAB
- Running MATLAB and MATLAB Environment
- Getting help
- Variables, Arithmetic and Logical Operators
- Matrices and Vectors
- Mathematical Functions
- Plotting
- Programming
- M-files
- User Defined Functions
- Miscellaneous
- Tips

Introduction to MATLAB

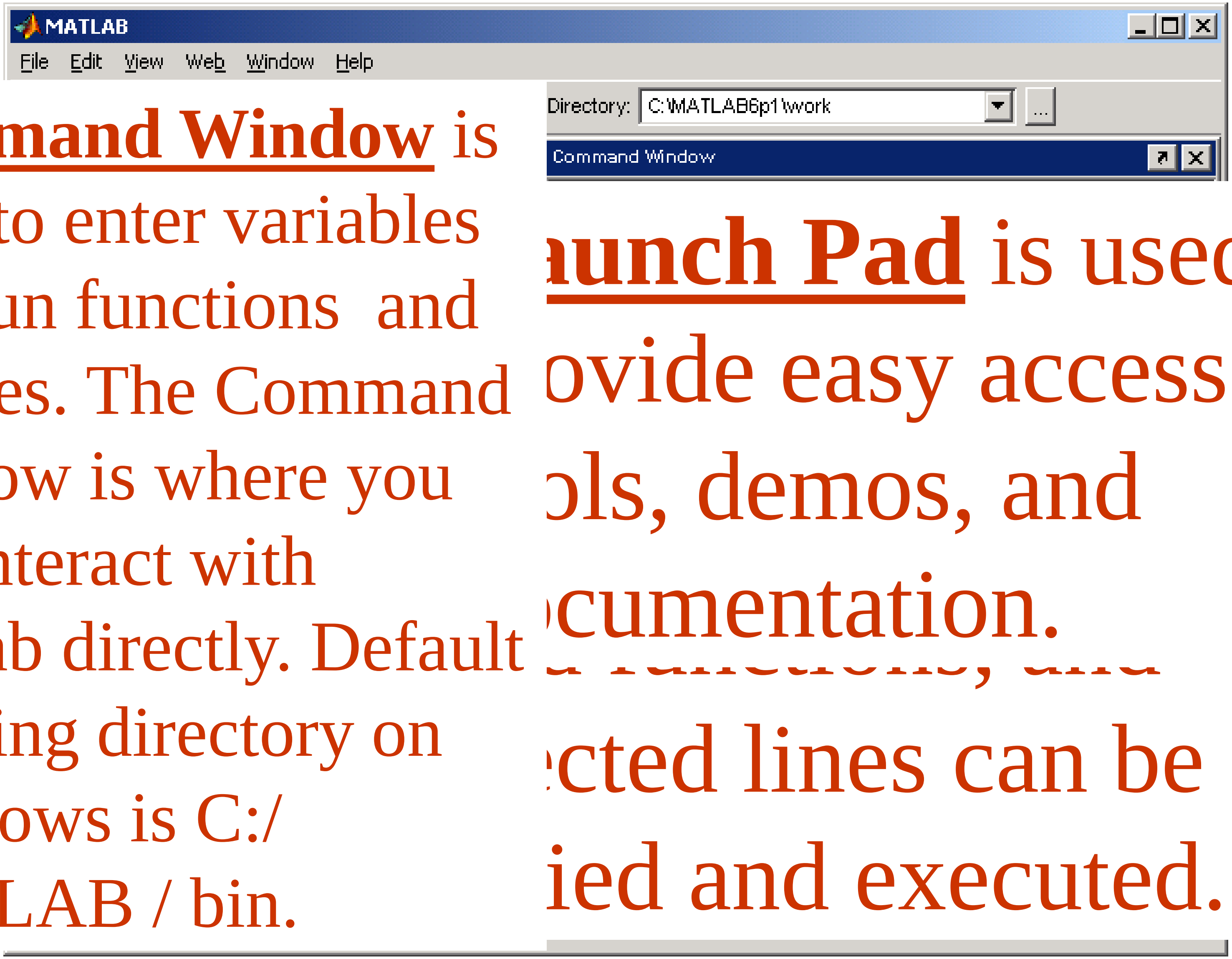
Dr NOOR BADSHAH

Running MATLAB & the MATLAB Environment

You can enter MATLAB with system command “matlab” , C:> matlab. Or, it can be started by clicking on the start-up menu or a short-cut icon



MATLAB Desktop



Directory: C:\MATLAB6p1\work

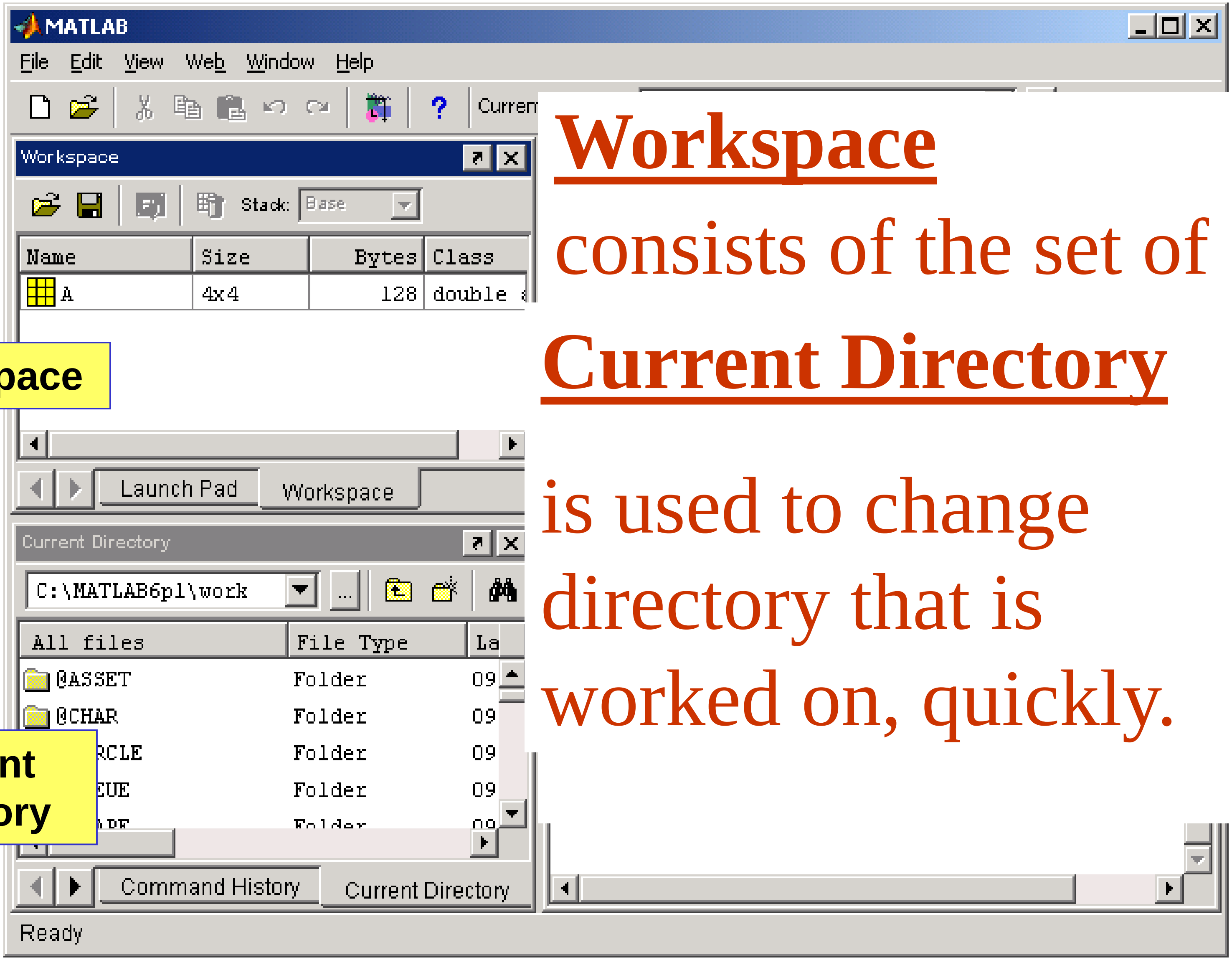
<u>Command Window</u> is used to enter variables and run functions and M-files. The Command window is where you can interact with Matlab directly. Default working directory on Windows is C:/ MATLAB / bin.

<u>aunch Pad</u> is used to ovide easy access to ols, demos, and cumentation.

d functions, and

cted lines can be ied and executed.

MATLAB Desktop – cont’d



Name	Size	Bytes	Class
A	4x4	128	double

C:\MATLAB6p1\work

All files	File Type	La
@ASSET	Folder	09
@CHAR	Folder	09
RCLE	Folder	09
EUE	Folder	09
APE	Folder	09

Workspace

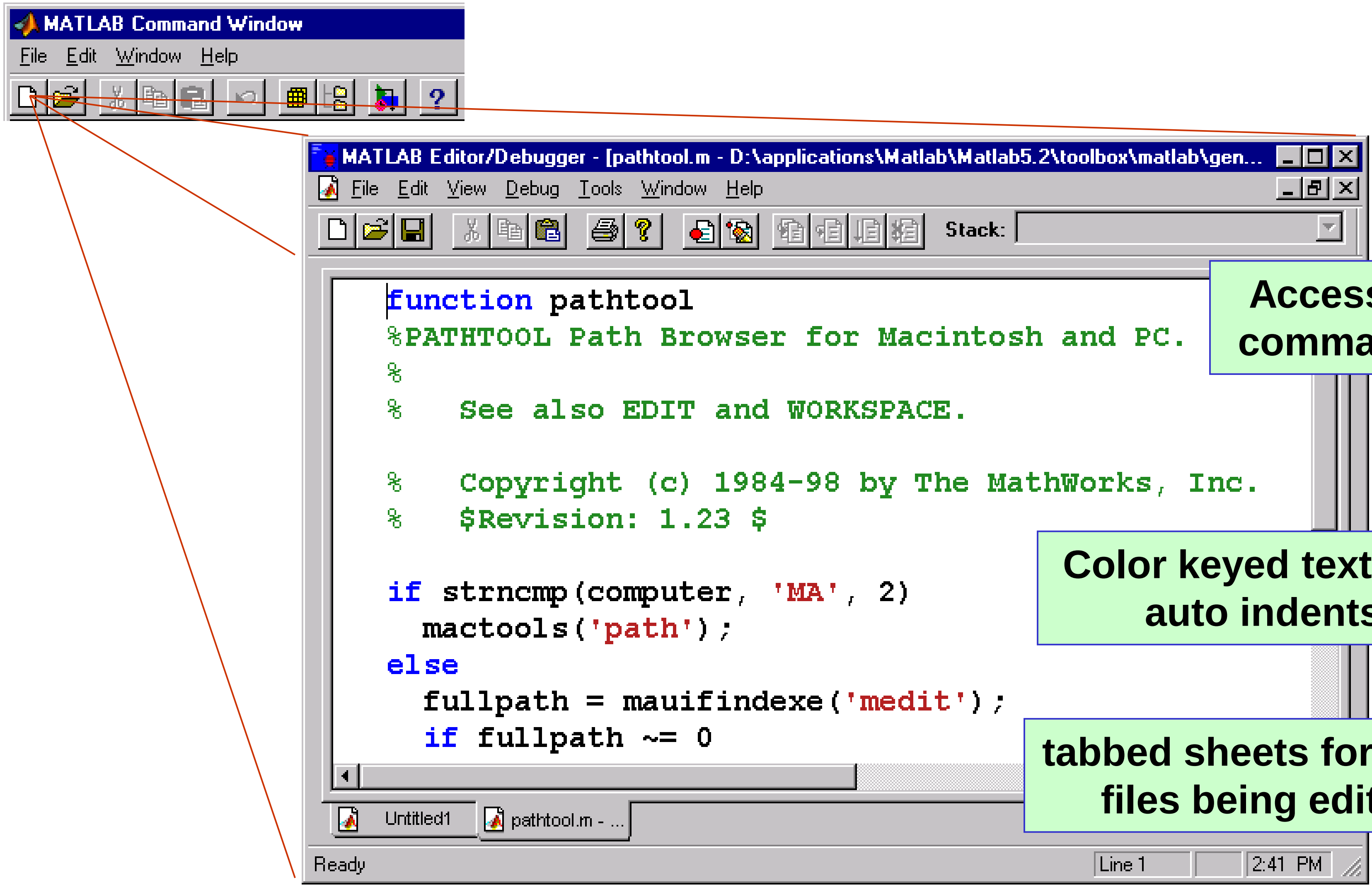
Current Directory

Workspace consists of the set of

;)

Current Directory is used to change directory that is worked on, quickly.

MATLAB Editor


MATLAB Command Window
File Edit Window Help
MATLAB Editor/Debugger - [pathtool.m - D:\applications\Matlab\Matlab5.2\toolbox\matlab\gen...
File Edit View Debug Tools Window Help
Stack:
Untitled1
pathtool.m - ...
Ready
Line 1
2:41 PM

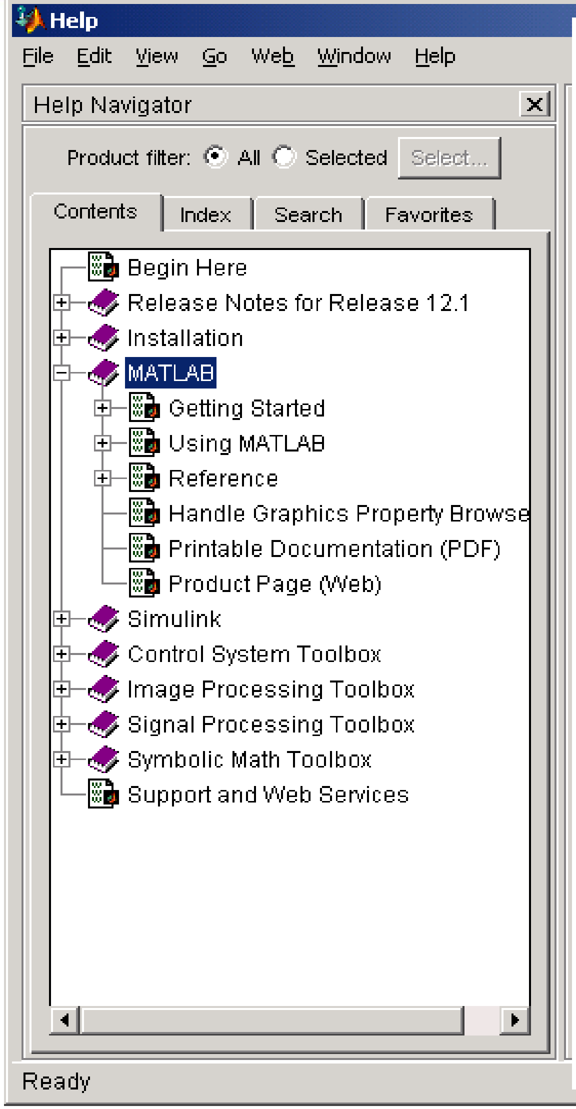
```
function pathtool
%PATHTOOL Path Browser for Macintosh and PC.
%
%   See also EDIT and WORKSPACE.

%   Copyright (c) 1984-98 by The MathWorks, Inc.
%   $Revision: 1.23 $

if strncmp(computer, 'MA', 2)
  mactools('path');
else
  fullpath = mauifindexe('medit');
  if fullpath ~= 0
```

Access to commands

Color keyed text with auto indents

tabbed sheets for other files being edited

Getting MATLAB Help


Help
File Edit View Go Web Window Help
Help Navigator
Product filter: ● All ○ Selected Select...
Contents Index Search Favorites
Begin Here
Release Notes for Release 12.1
Installation
MATLAB
Getting Started
Using MATLAB
Reference
Handle Graphics Property Browse
Printable Documentation (PDF)
Product Page (Web)
Simulink
Control System Toolbox
Image Processing Toolbox
Signal Processing Toolbox
Symbolic Math Toolbox
Support and Web Services
Ready

- Type one of the following commands in the command window:
  - >>**help** – lists all the help topics
  - >>**help** *topic* – provides help for the specified topic
  - >>**help** *command* – provides help for the specified command
  - >>**helpwin** – opens a separate help window for navigation
  - >>**Lookfor** *keyword* – search all M-files for *keyword*
- Online resource

MATLAB Variables

- The MATLAB environment is command oriented somewhat like UNIX. A prompt appears on the screen and a MATLAB statement can be entered. When the <ENTER> key is pressed, the statement is executed, and another prompt appears.
- If a statement is terminated with a semicolon (;), no results will be displayed. Otherwise results will appear before the next prompt.
- Variable names ARE case sensitive.
- Variable names can contain up to 63 characters (as of MATLAB 6.5 and newer).
- Variable names must start with a letter followed by letters, digits, and underscores.
- Variable names and their types do not have to be declared in MATLAB.
- Any variable can take real, complex, and integer values.
- The name of variable is not accepted if it is reserved word.

MATLAB Variables – cont’d

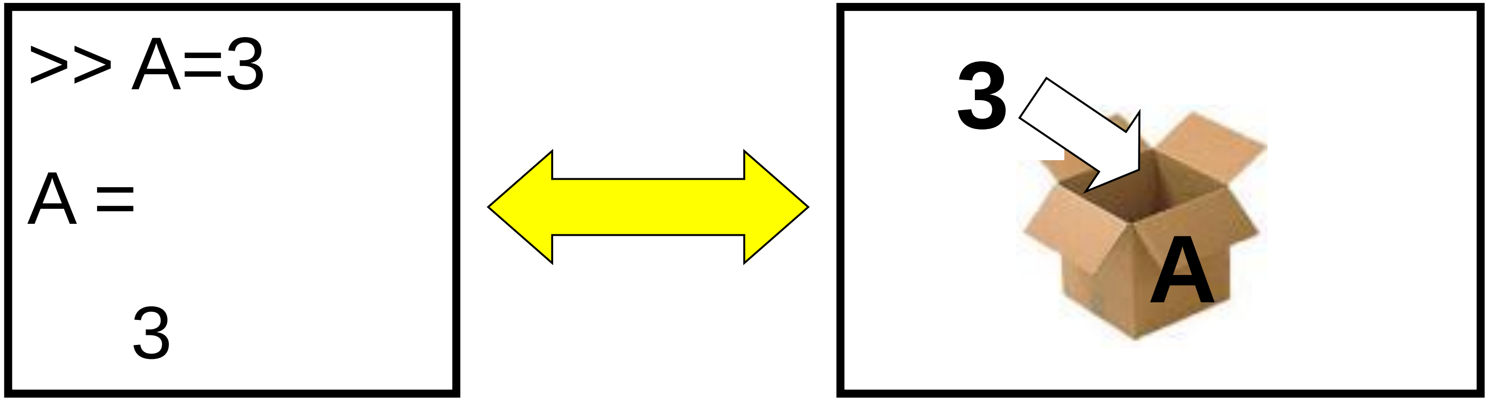
- Special variables:
  - **ans**: default variable name for the result.
  - **pi**: π = 3.1415926 ……
  - **eps**: ε= 2.2204e-016, smallest value by which two numbers can differ
  - **inf**: ∞, infinity
  - **NAN** or **nan**: not-a-number

- Commands involving variables:
  - **who**: lists the names of the defined variables
  - **whos**: lists the names and sizes of defined variables
  - **clear**: clears all variables
  - **clear** *name*: clears the variable *name*
  - **clc**: clears the command window
  - **clf**: clears the current figure and the graph window
  - **Ctrl+C**: Aborts calculation

MATLAB Variables – cont’d

- You can think of computer memory as a large set of “boxes” in which numbers can be stored. The values can be inspected and changed.

- Boxes can be labeled with a variable name.

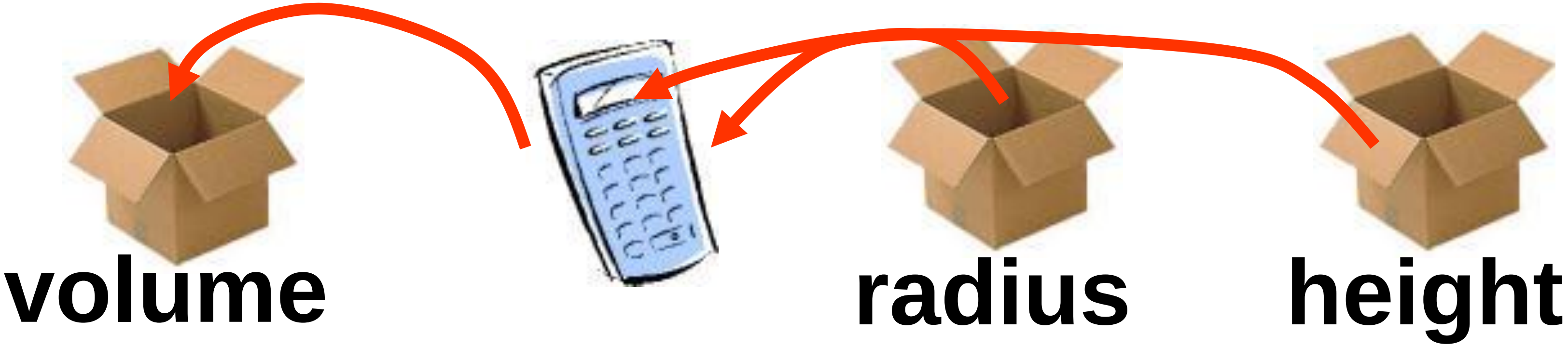


```
>> A=3

A =

     3
```

MATLAB Variables – cont’d

- Suppose we want to calculate the volume of a cylinder.
- It’s radius and height are stored as variables in memory.

```
>> volume = pi*radius^2*height
```



MATLAB Variables – cont’d

- *Variable is a name given to a reserved location in memory.*

```
>>x = 111;
>>number_of_students = 75;
>>name = ‘UET Peshawar’;
>>radius = 5;
>>area = pi * radius^2;
>>x_value=23
x_value=23
```

MATLAB Arithmetic Operators

Operator	Description
+	Addition
-	Subtraction
.*	Multiplication (element wise)
./	Right division (element wise)
.\	Left division (element wise)
=	Assignment operator,e.g. a = b,(assign b to a)
:	Colon operator (Specify Range)
.^	Power (element wise)
'	Transpose
*	Matrix multiplication
/	Matrix right division
\	Matrix left division
;	Row separator in a Matrix
^	Matrix power

Logical Operators in MATLAB

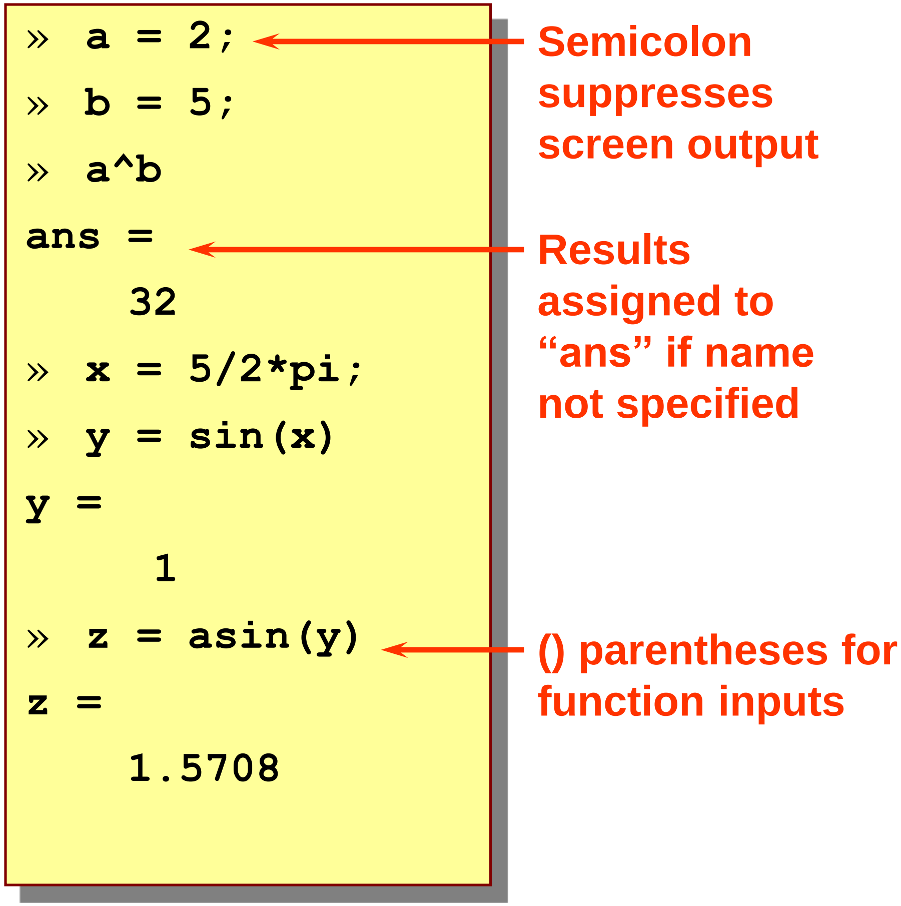
Operator	**Description**
&	Returns 1 for every element location that is true (nonzero) in both arrays, and 0 for all other elements.
|	Returns 1 for every element location that is true (nonzero) in either one or the other, or both, arrays and 0 for all other elements.
~	Complements each element of input array, A.
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Equal to
~=	Not equal to

Calculations at the Command Line / Workspace

MATLAB as a calculator

```
»  -5/(4.8+5.32)^2
ans =
   -0.0488
» (3+4i)*(3-4i)
ans =
    25
» cos(pi/2)
ans =
  6.1230e-017
» exp(acos(0.3))
ans =
    3.5470
```

Assigning Variables



```
»  a = 2;
»  b = 5;
»  a^b
ans =
    32
»  x = 5/2*pi;
»  y = sin(x)
y =
     1
»  z = asin(y)
z =
    1.5708
```

Semicolon suppresses screen output

Results assigned to “ans” if name not specified

() parentheses for function inputs

Vector & Matrix in MATLAB

Columns

How do you specify this 5 x 5 matrix 'A' in MATLAB ?

A =

Rows (m)					
1	4[1]	10[6]	1[11]	6[16]	2[21]
2	8[2]	1.2[7]	9[12]	4[17]	25[22]
3	7.2[3]	5[8]	7[13]	1[18]	11[23]
4	0[4]	0.5[9]	4[14]	5[19]	56[24]
5	23[5]	83[10]	13[15]	0[20]	10[25]

```
>>A=[4  10  1  6  2
        8  1.2  9 4  25
        7.2  5  7  1  11
        0  0.5 4  5  56
        23  83 13 0  10 ];
```

```
>>A=[4  10  1  6  2; 8  1.2  9  4  25; 7.2  5  7   1  11; 0  0.5 4   5  56; 23  83  13 0  10 ];
```

Examples (Vectors)

```
X=[2 7 4];
```

2	7	4

Row Vector

```
X=[2; 7; 4];

X=[2 7 4]';
```

2
7
4

Column Vector

```
X=[2 7 4;3 8 9];
```

2	7	4
3	8	9

Matrix or a 2D array

```
Y=[X X];
```

2	7	4	2	7	4
3	8	9	3	8	9

Matrix of matrices

More on Vectors

x = start:end	Creates row vector x starting with start, counting by 1 , ending at end
x = initial value : increment : final value	Creates row vector x starting with start, counting by increment, ending at or before end
x = linspace(start,end,number)	Creates linearly spaced row vector x starting with start, ending at end, having number elements
x = logspace(start,end,number)	Creates logarithmically spaced row vector x starting with start, ending with end, having number elements
length(x)	Returns the length of vector x
y = x’	Transpose of vector x
dot(x,y),cross(x,y)	Returns the scalar dot and vector cross product of the vector x and y

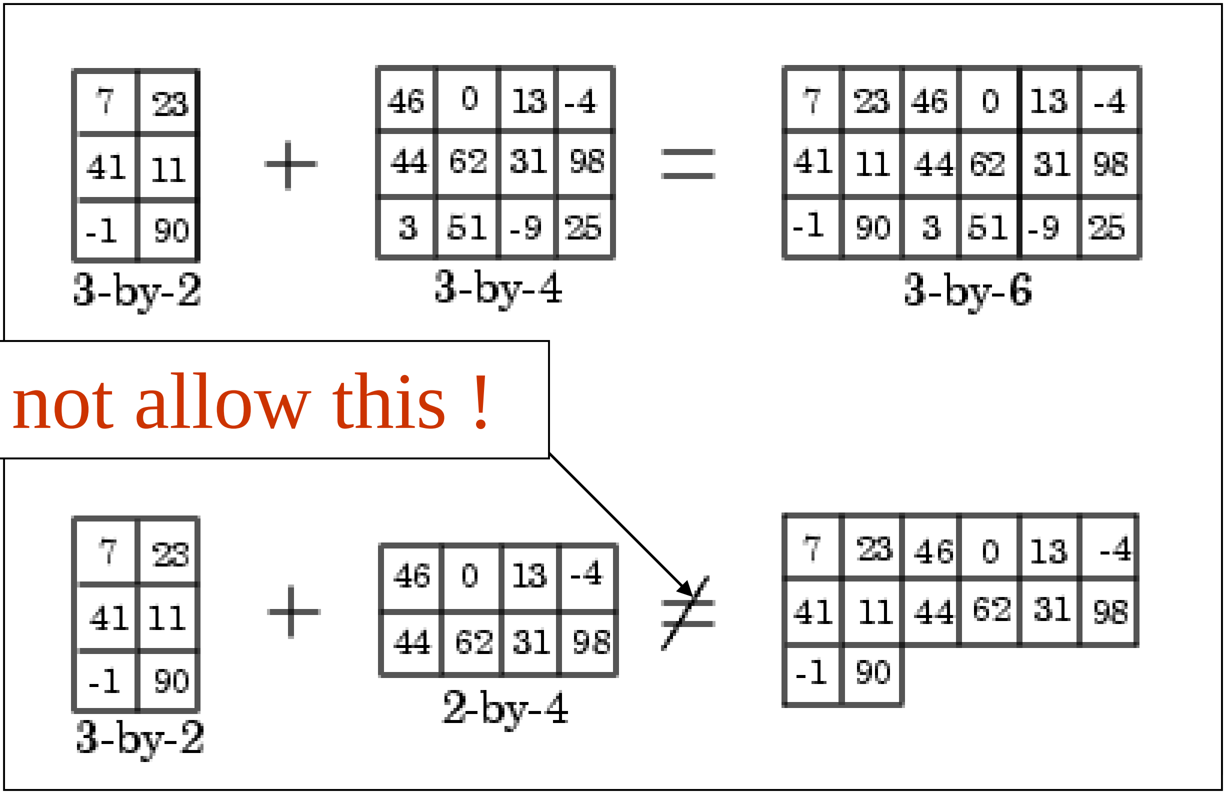
More on Matrices

zeros(n)	**Returns a n x n matrix of zeros**
zeros(m,n)	**Returns a m x n matrix of zeros**
rand(m,n)	**Returns a m x n matrix of random numbers**
eye(m,n)	**Returns a m x n Identity matrix**
ones(n)	**Returns a n x n matrix of ones**
ones(m,n)	**Returns a m x n matrix of ones**
size(A)	**For a m x n matrix A, returns the row vector [m,n] containing the number of rows and columns in matrix**
length(A)	**Returns the larger of the number of rows or columns in A**

Entering Numeric Arrays

Row separator:
semicolon (;)

Column separator:
space / comma (,)



$$\underset{\text{3-by-2}}{\begin{bmatrix}7&23\\41&11\\-1&90\end{bmatrix}} + \underset{\text{3-by-4}}{\begin{bmatrix}46&0&13&-4\\44&62&31&98\\3&51&-9&25\end{bmatrix}} = \underset{\text{3-by-6}}{\begin{bmatrix}7&23&46&0&13&-4\\41&11&44&62&31&98\\-1&90&3&51&-9&25\end{bmatrix}}$$

MATLAB does not allow this !

$$\underset{\text{3-by-2}}{\begin{bmatrix}7&23\\41&11\\-1&90\end{bmatrix}} + \underset{\text{2-by-4}}{\begin{bmatrix}46&0&13&-4\\44&62&31&98\end{bmatrix}} \neq \begin{array}{|c|c|c|c|c|c|}\hline 7&23&46&0&13&-4\\ \hline 41&11&44&62&31&98\\ \hline -1&90&&&&\\ \hline\end{array}$$

Matrices must be rectangular/same height. **(Set undefined elements to zero)**

Any MATLAB expression can be entered as a matrix element

Entering Numeric Arrays - cont.

Scalar expansion

Creating sequences colon operator (:)

Utility functions for creating matrices.

```
» w=[1 2;3 4] + 5
w =
     6     7
     8     9
» x = 1:5
x =
     1     2     3     4     5
» y = 2:-0.5:0
y =
  2.0000   1.5000   1.0000   0.5000   0
» z = rand(2,4)
z =
  0.9501   0.6068   0.8913   0.4565
  0.2311   0.4860   0.7621   0.0185
```

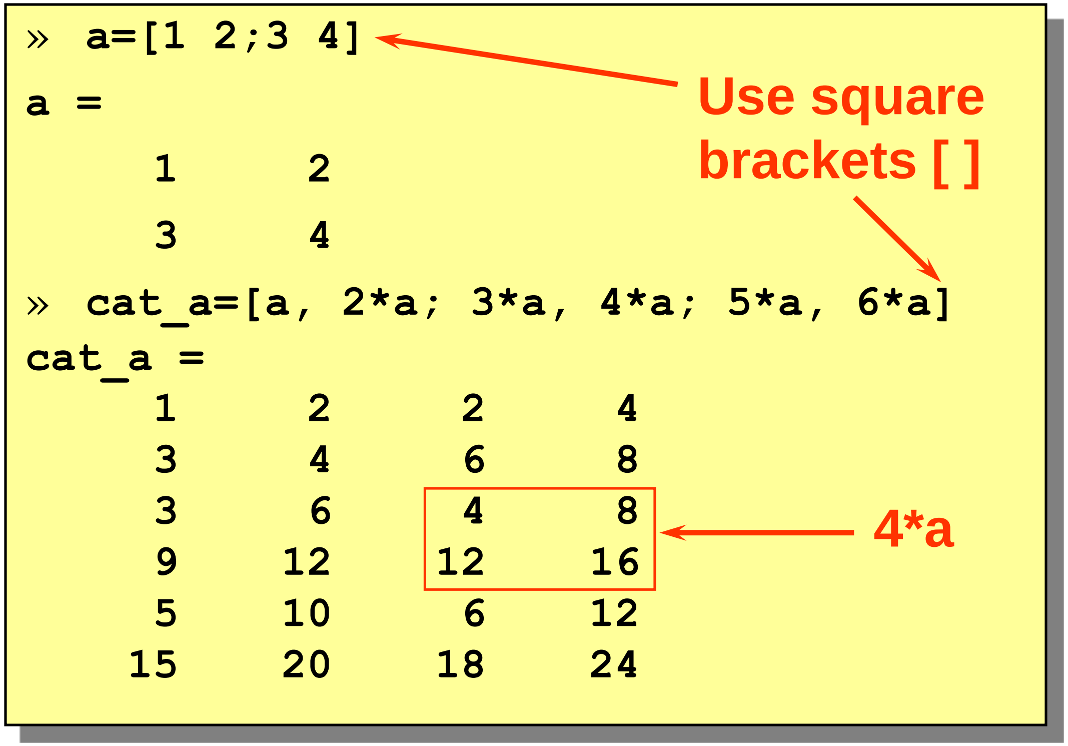
Numerical Array Concatenation - [ ]

Use [ ] to combine existing arrays as matrix “elements”

<u>Row separator:</u>
semicolon (;)

<u>Column separator:</u>
space / comma (,)

The resulting matrix must be rectangular.

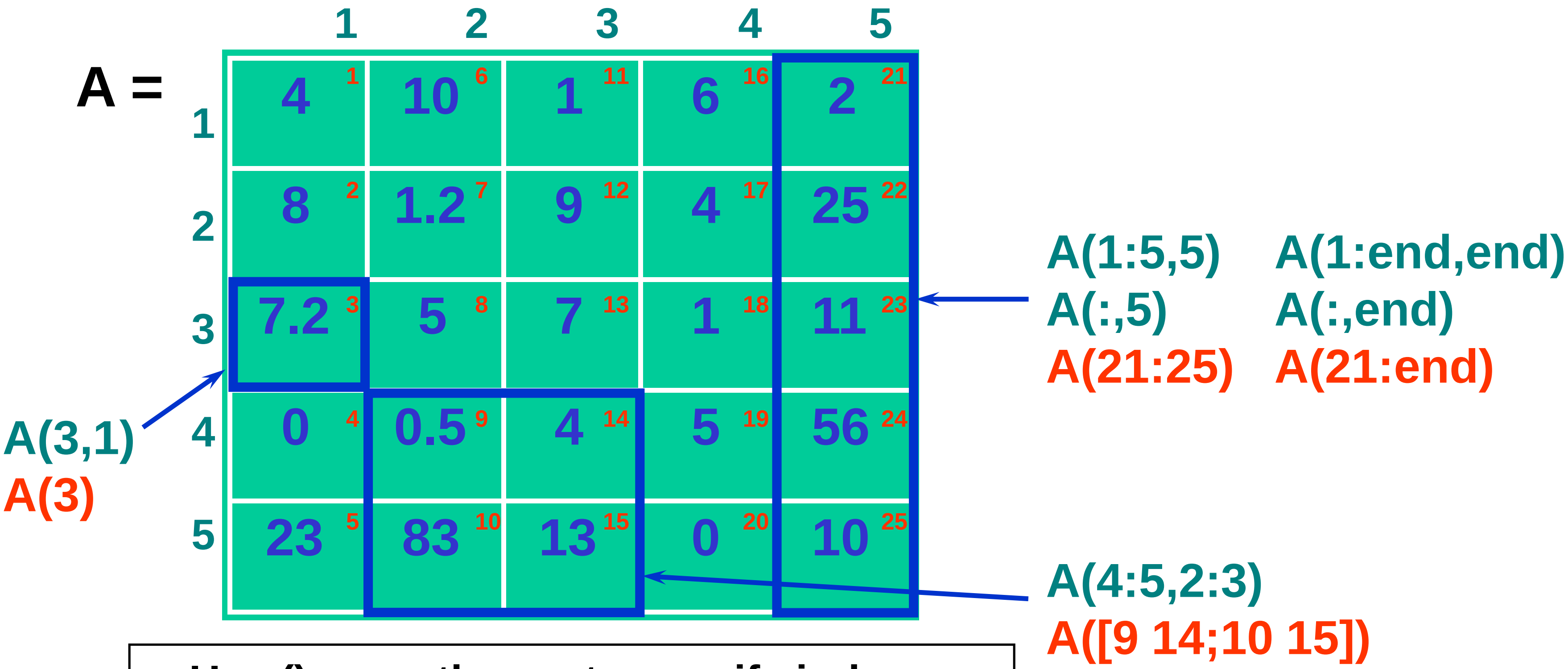


```
» a=[1 2;3 4]
a =
     1     2
     3     4
» cat_a=[a, 2*a; 3*a, 4*a; 5*a, 6*a]
cat_a =
     1     2     2     4
     3     4     6     8
     3     6     4     8
     9    12    12    16
     5    10     6    12
    15    20    18    24
```

Use square brackets [ ]

4*a

Array Subscripting / Indexing



$$A = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 4^{1} & 10^{6} & 1^{11} & 6^{16} & 2^{21} \\ 2 & 8^{2} & 1.2^{7} & 9^{12} & 4^{17} & 25^{22} \\ 3 & 7.2^{3} & 5^{8} & 7^{13} & 1^{18} & 11^{23} \\ 4 & 0^{4} & 0.5^{9} & 4^{14} & 5^{19} & 56^{24} \\ 5 & 23^{5} & 83^{10} & 13^{15} & 0^{20} & 10^{25} \end{array}$$

A(3,1)
A(3)

A(1:5,5)   A(1:end,end)
A(:,5)   A(:,end)
A(21:25)   A(21:end)

A(4:5,2:3)
A([9 14;10 15])

- Use () parentheses to specify index
- colon operator (:) specifies range / ALL
- [ ] to create matrix of index subscripts
- ‘end’ specifies maximum index value

Some operations should be handled with care

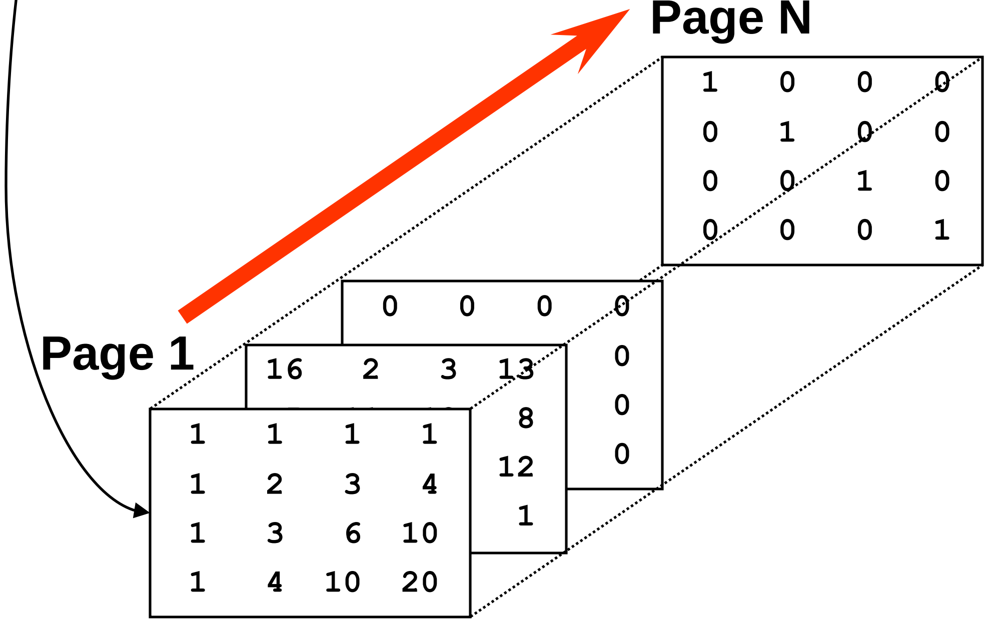
```
>>A=[1 2;4 5];

>>B=A*A      % prints

9    12

24   33             % Proper matrix multiplication

>>

>>B=A.*A      % prints

1     4

16   25             % Element by element multiplication
```

Operations on Matrices

Transpose	B=A’
Identity Matrix	eye(n) -> returns an n X n identity matrix eye(m,n) -> returns an m X n matrix with ones on the main diagonal and zeros elsewhere
Addition and Subtraction	C =A +B C =A - B
Scalar Multiplication	B = α A, where α is a scalar
Matrix Multiplication	C = A * B
Matrix Inverse	B = inv(A), A must be a square matrix in this case
Matrix powers	B = A * A , A must be a square matrix
Determinant	det(A), A must be a square matrix

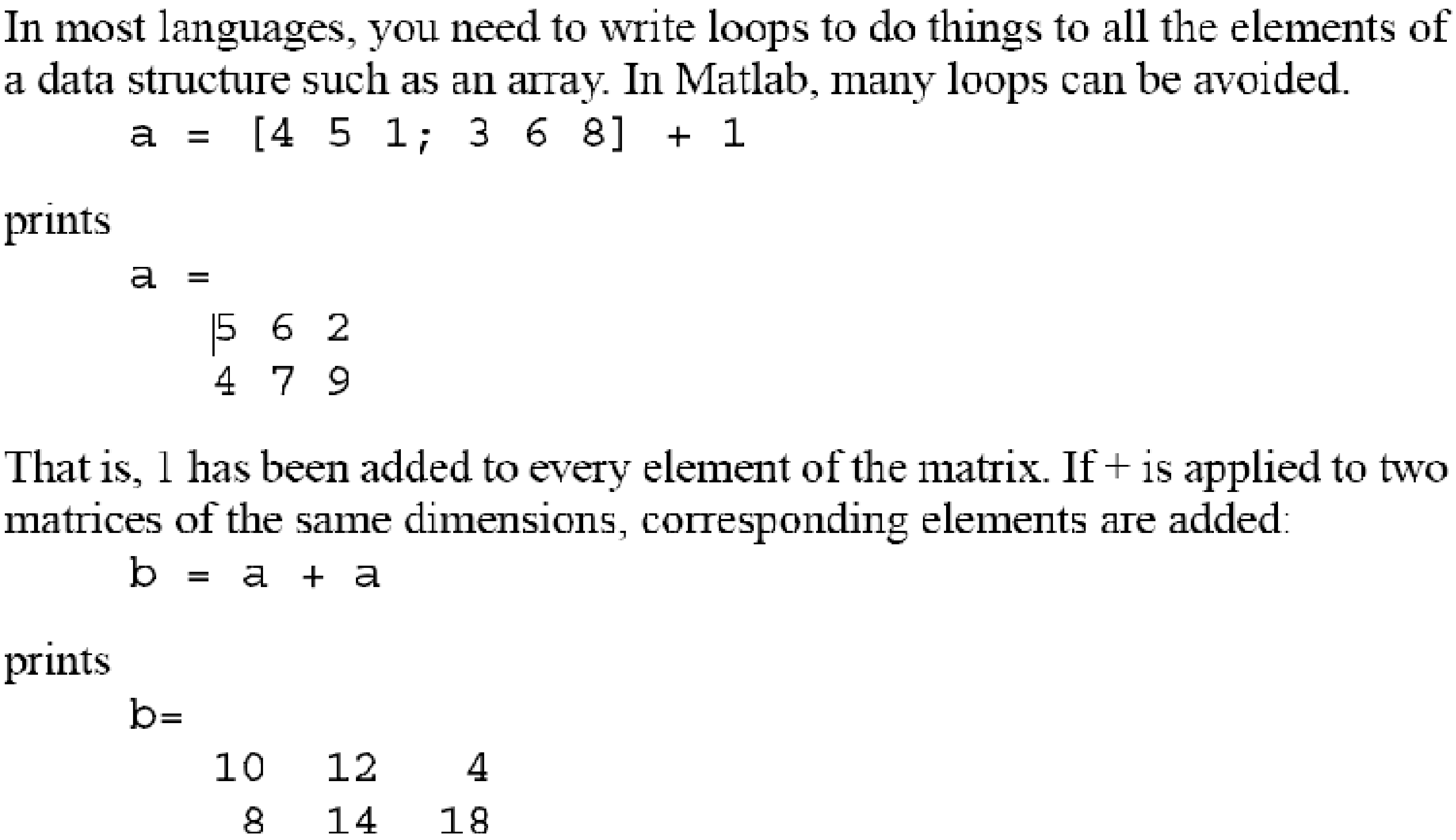
Multidimensional Arrays

```
>>A = 1;
while length(A) < 4
A = [0 A] + [A 0];
end
```



```
» A = Pascal(4);
» A(:,:,2) = magic(4)
A(:,:,1)
      1      1      1      1
      1      2      3      4
      1      3      6     10
      1      4     10     20
A(:,:,2)
     16      2      3     13
      5     11     10      8
      9      7      6     12
      4     14     15      1
» A(:,:,9) = diag(ones(1,4));
```

Operating on Matrices



In most languages, you need to write loops to do things to all the elements of a data structure such as an array. In Matlab, many loops can be avoided.

```
        a = [4 5 1; 3 6 8] + 1
```

prints

```
        a =
               5  6  2
               4  7  9
```

That is, 1 has been added to every element of the matrix. If + is applied to two matrices of the same dimensions, corresponding elements are added:

```
        b = a + a
```

prints

```
        b=
               10   12    4
                8   14   18
```

Operating on Matrices - cont.

Array Multiplication

- Matrices must have the same dimensions
- Dimensions of resulting matrix = dimensions of multiplied matrices
- Resulting elements = product of corresponding elements from the original matrices

```
» a = [1 2 3 4; 5 6 7 8];
» b = [1:4; 1:4];
» c = a.*b
c =
      1      4      9     16
      5     12     21     32
```

c(2,4) = a(2,4)*b(2,4)

Same rules apply for other array operations too !

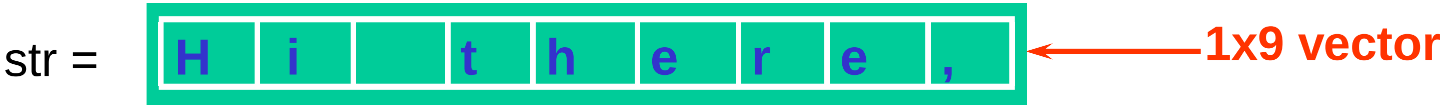
String Arrays

- Created using single quote delimiter (')

```
» str  = 'Hi there,'
str =
Hi there,
» str2 = 'Isn't MATLAB great?'
str2 =
Isn't MATLAB great?
```

- Each character is a separate matrix element

(16 bits of memory per character)



str =

H	i		t	h	e	r	e	,

1x9 vector

- Indexing same as for numeric arrays

Mathematical Functions of MATLAB-1

Elemantary Mathematical (Trigonometric) Functions	
Trigonometric functions	Remarks

sin(x)
cos(x)
tan(x)
asin(x)
acos(x)
atan(x)
atan2(y,x)
sinh(x)
cosh(x)
tanh(x)
asinh(x)
acosh(x)
atanh(x)
	- pi/2 ≤ atan(x) ≥ pi/2, Same as atan(y/x) but –pi ≥ atan(y,x) ≥ pi

Mathematical Functions of MATLAB-2

Other elamantary functions	Remarks
abs(x)	Absolute value of x
angle(x)	Phase angle of complex value: If x = real, angle = 0. If x = √-1, angle = pi/2
sqrt(x)	Square root of x
real(x)	Real part of complex value x
imag(x)	Imaginary part of complex value x
conj(x)	Complex conjugate x
round(x)	Round to do nearest integer
fix(x)	Round a real value toward zero
floor(x)	Round x toward - ∞
ceil(x)	Round x toward + ∞
sign(x)	+1 if x > 0; -1 if x < 0
exp(x)	Exponential base e
log(x)	Log base e
log10(x)	Log base 10
factor(x)	1 if x is a prime number, 0 if not

And there are many many more !

Plotting in MATLAB

- Specify x-data and/or y-data
- Specify color, line style and marker symbol

- Syntax: 2-D Plotting
  - Plotting single line:

```
plot(xdata, ydata, 'color_linestyle_marker')
```

  - Plotting multiple lines:

```
plot(x1, y1, 'clm1', x2, y2, 'clm2', ...)
```

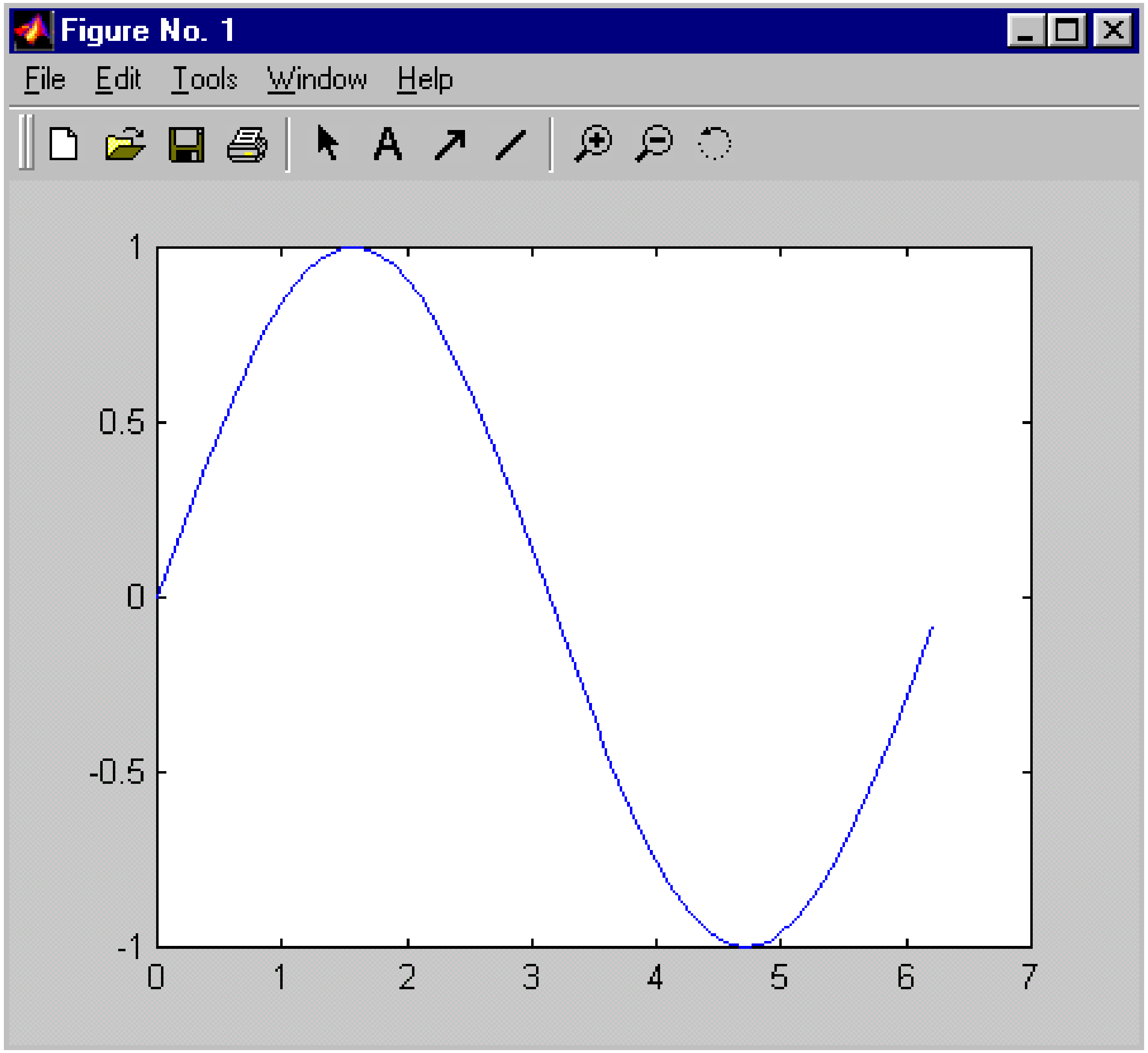
2-D Plotting - example

Create a Blue

(default color)

Sine Wave

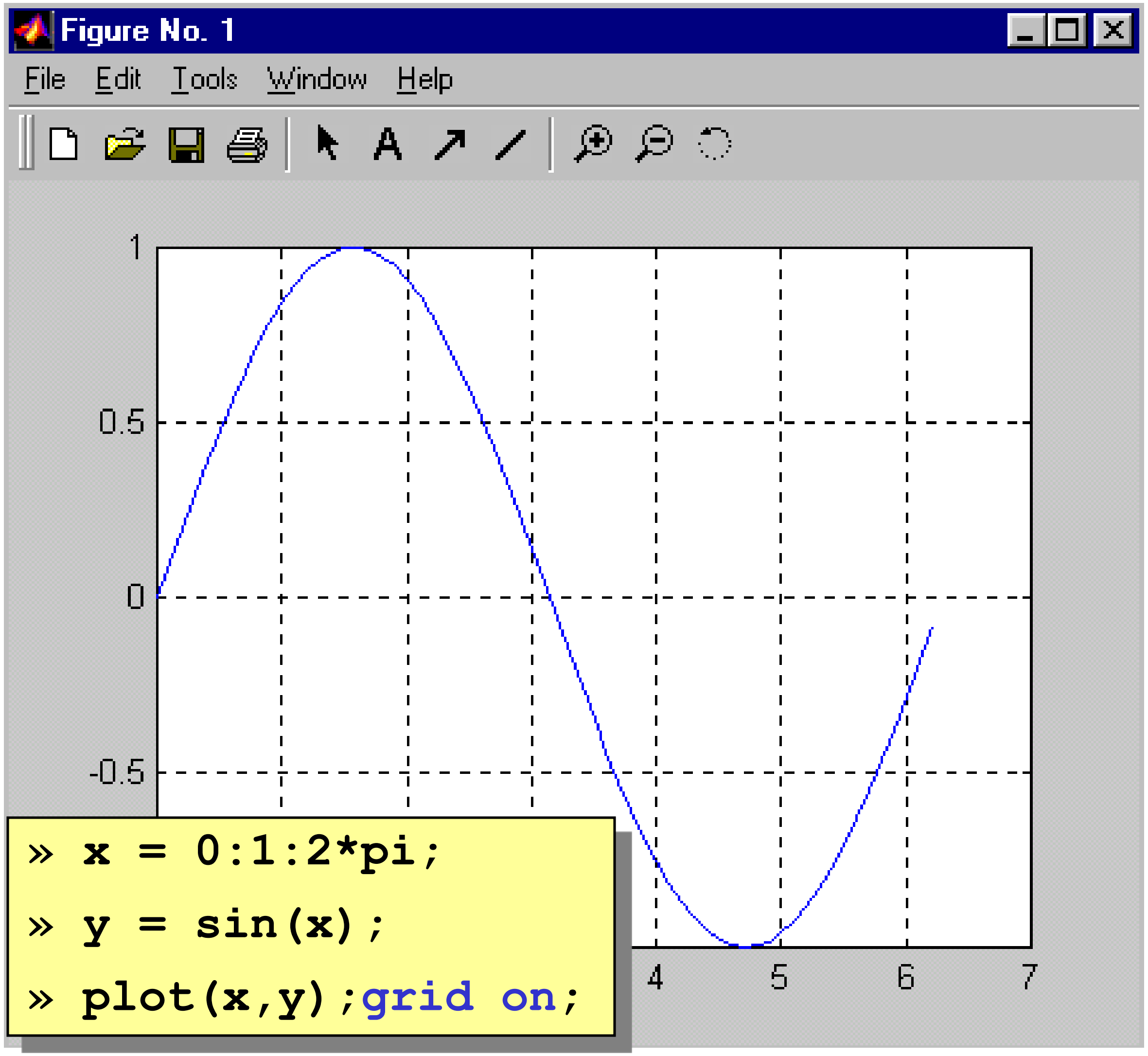
```
» x = 0:1:2*pi;
» y = sin(x);
» plot(x,y);
```



2-D Plotting : example-cont.

Adding a Grid

- GRID ON creates a grid on the current figure
- GRID OFF turns off the grid from the current figure
- GRID toggles the grid state

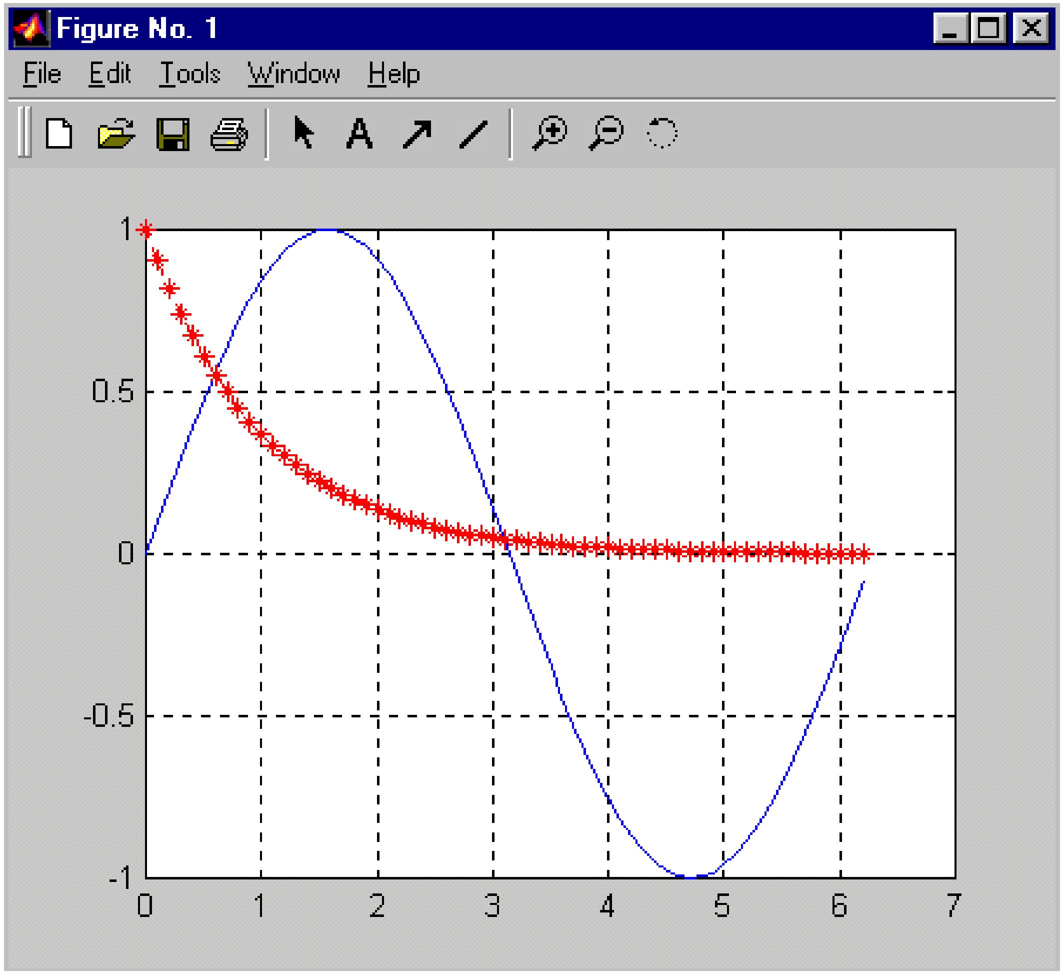


```
» x = 0:1:2*pi;
» y = sin(x);
» plot(x,y);grid on;
```

Adding additional plots to a figure

- HOLD ON holds the current plot
- HOLD OFF releases hold on current plot
- HOLD toggles the hold state

```
» x = 0:.1:2*pi;
» y = sin(x);
» plot(x,y,'b')
» grid on;
» hold on;
» plot(x,exp(-x),'r:*');
```



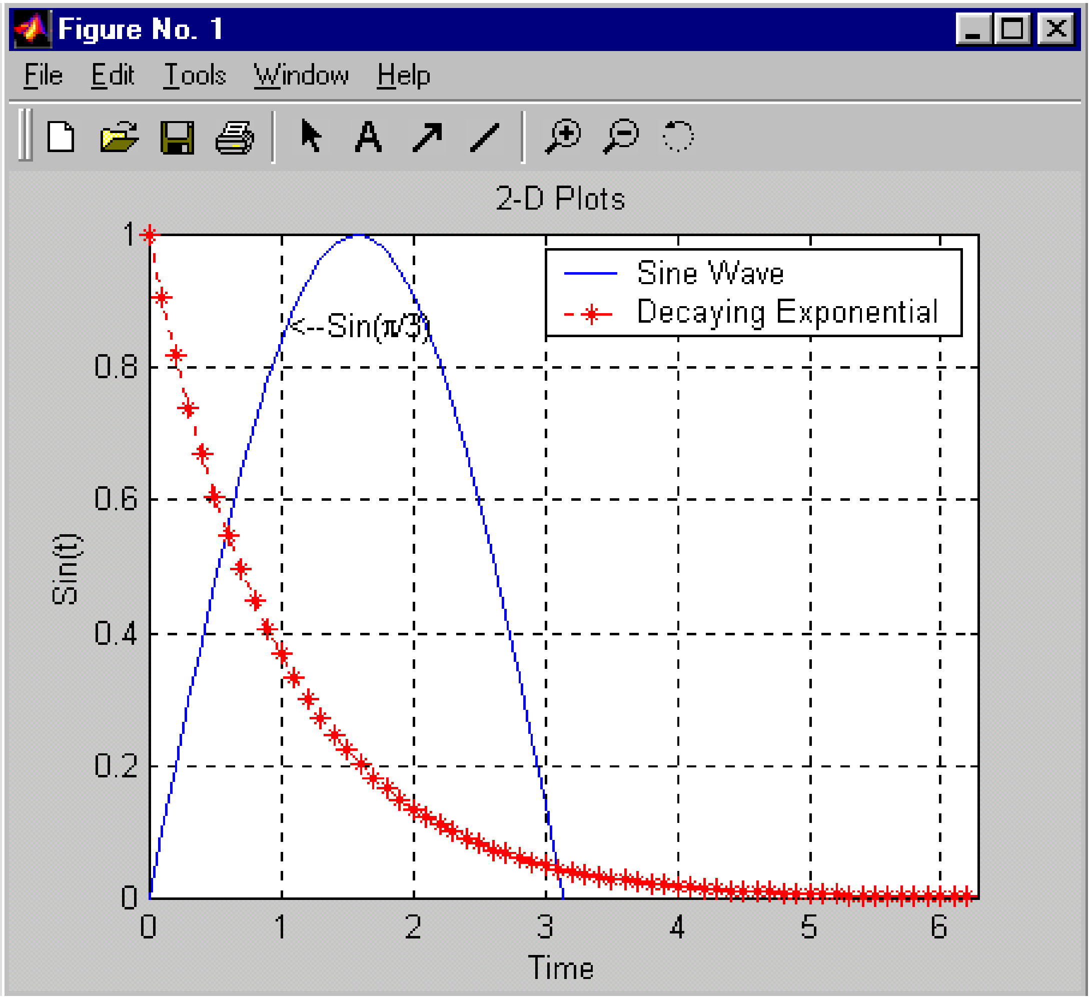
Controlling Viewing Area

- **ZOOM ON allows user to select viewing area**
- **ZOOM OFF prevents zooming operations**
- **ZOOM toggles the zoom state**

- **AXIS sets axis range**

```
[xmin xmax ymin ymax]
```

```
» axis([0 2*pi 0 1]);
```



Graph Annotation

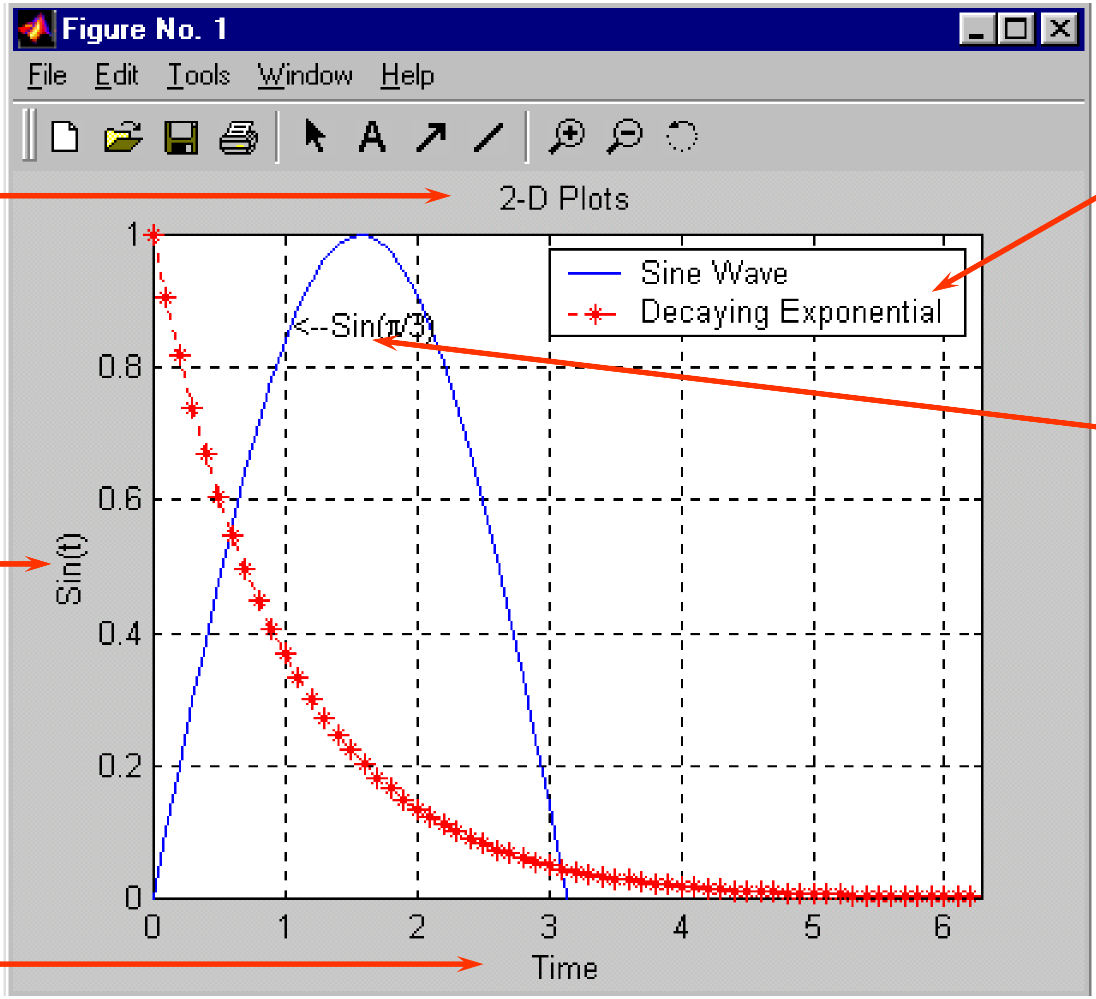
TITLE

LEGEND

TEXT
or
GTEXT

YLABEL

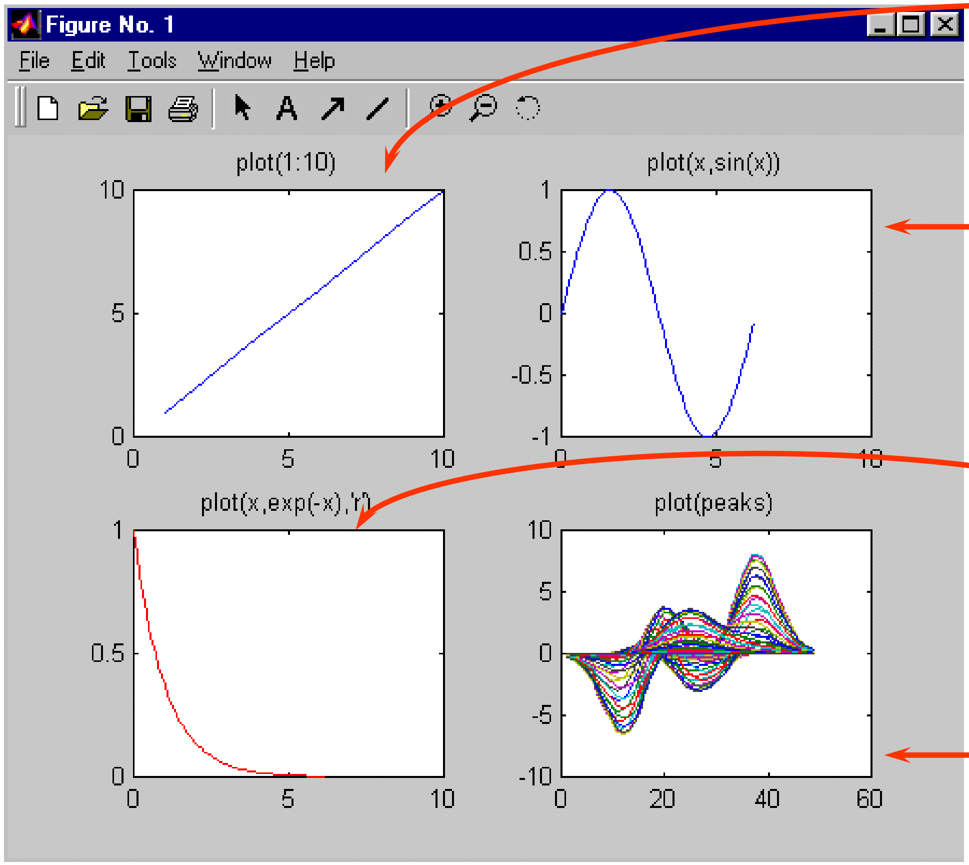
XLABEL



Subplots

SUBPLOT- display multiple axes in the same figure window

```
subplot(#rows, #cols, index);
```

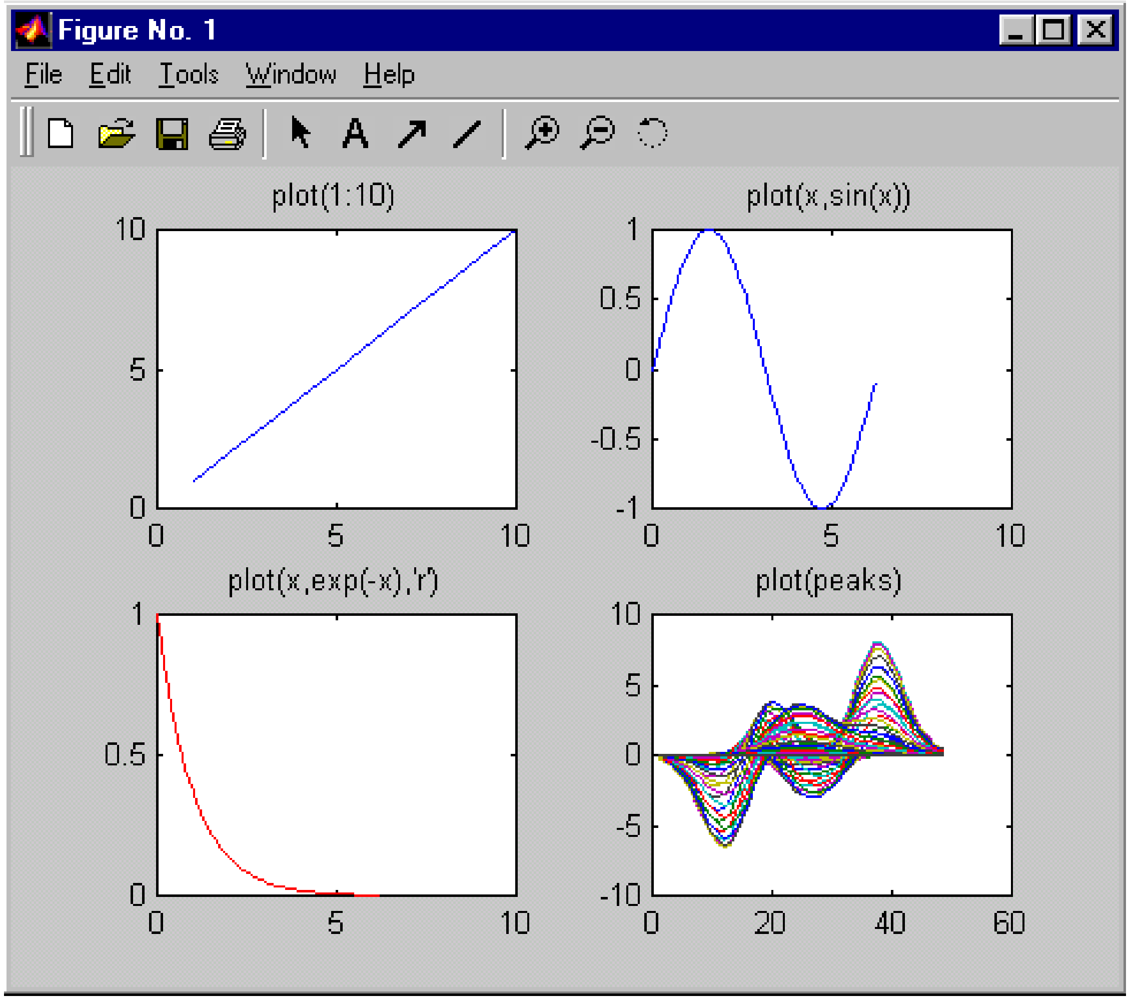


```
»subplot(2,2,1);
»plot(1:10);

»subplot(2,2,2);
»x = 0:.1:2*pi;
»plot(x,sin(x));

»subplot(2,2,3);
»x = 0:.1:2*pi;
»plot(x,exp(-x),’r’);

»subplot(2,2,4);
»plot(peaks);
```

Alternative Scales for Axes

LOGLOG
Both axes logarithmic

SEMILOGX
log X
linear Y



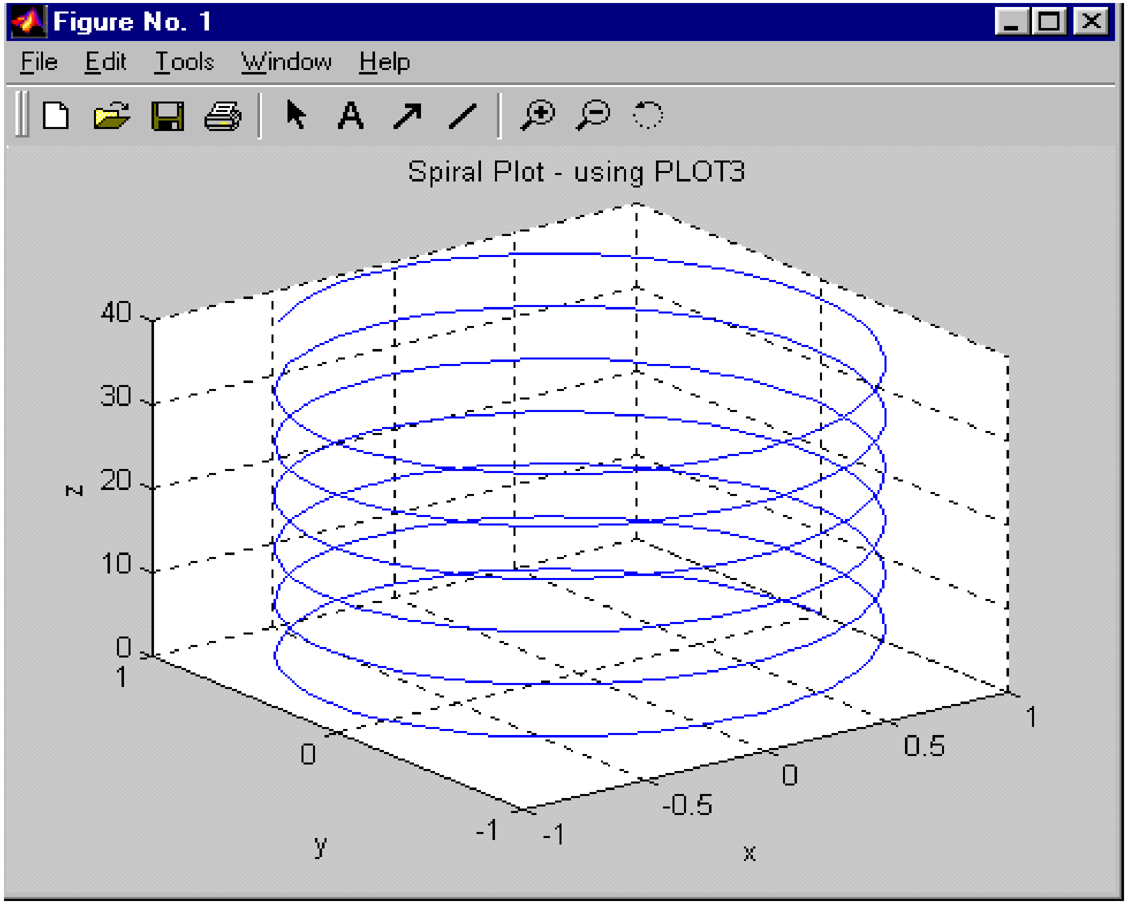
SEMILOGY
log Y
linear X

PLOTYY
2 sets of linear axes

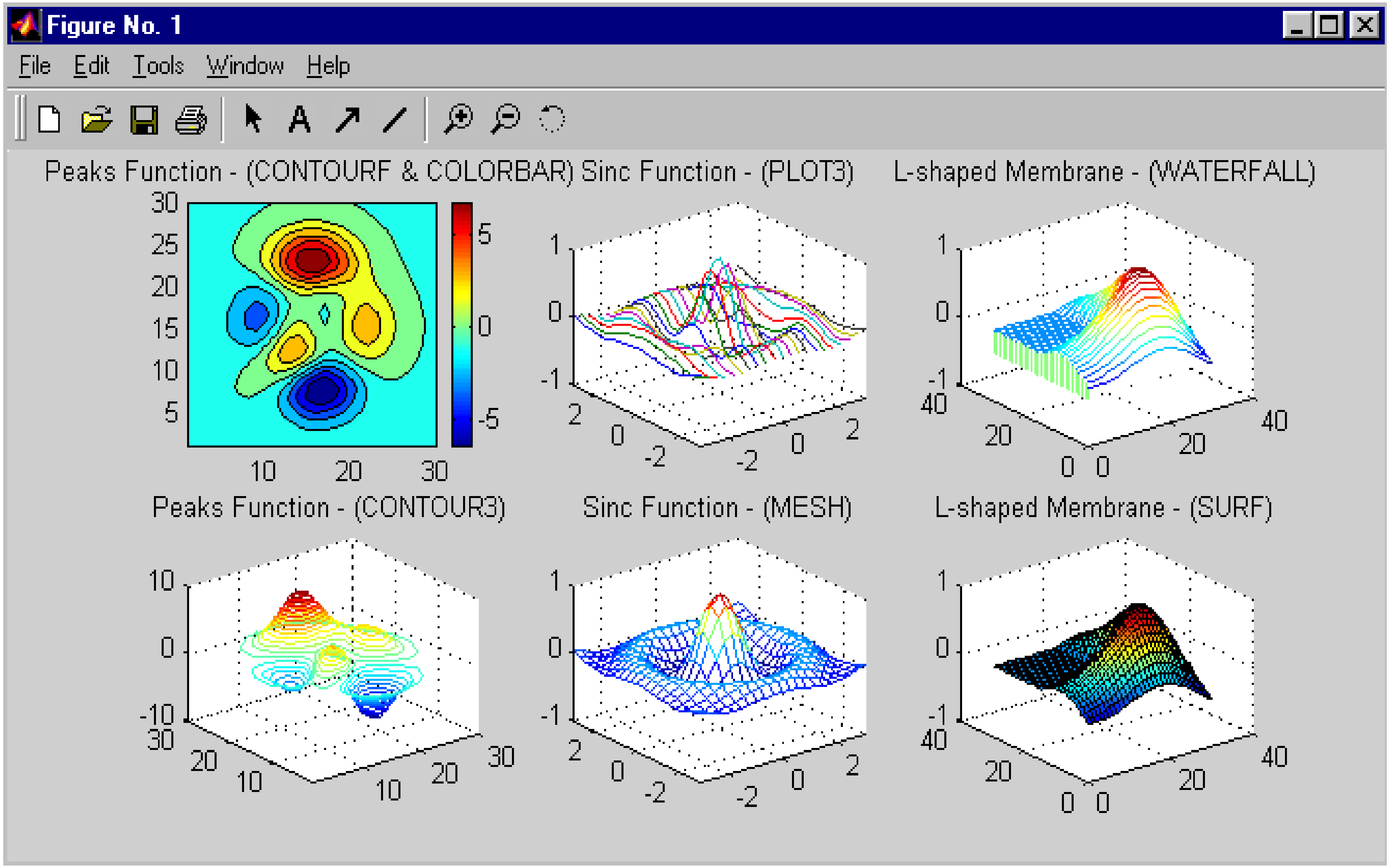
3-D Line Plotting

```
plot3(xdata, ydata, zdata, 'clm', ...)
```

```
» z = 0:0.1:40;
» x = cos(z);
» y = sin(z);
» plot3(x,y,z);grid on;
```



3-D Surface Plotting



Example:Advanced 3-D Plotting

```
» B = -0.2;
» x = 0:0.1:2*pi;
» y = -pi/2:0.1:pi/
» [x,y] = meshgrid(
» z = exp(B*x).*sin
» surf(x,y,z)
```

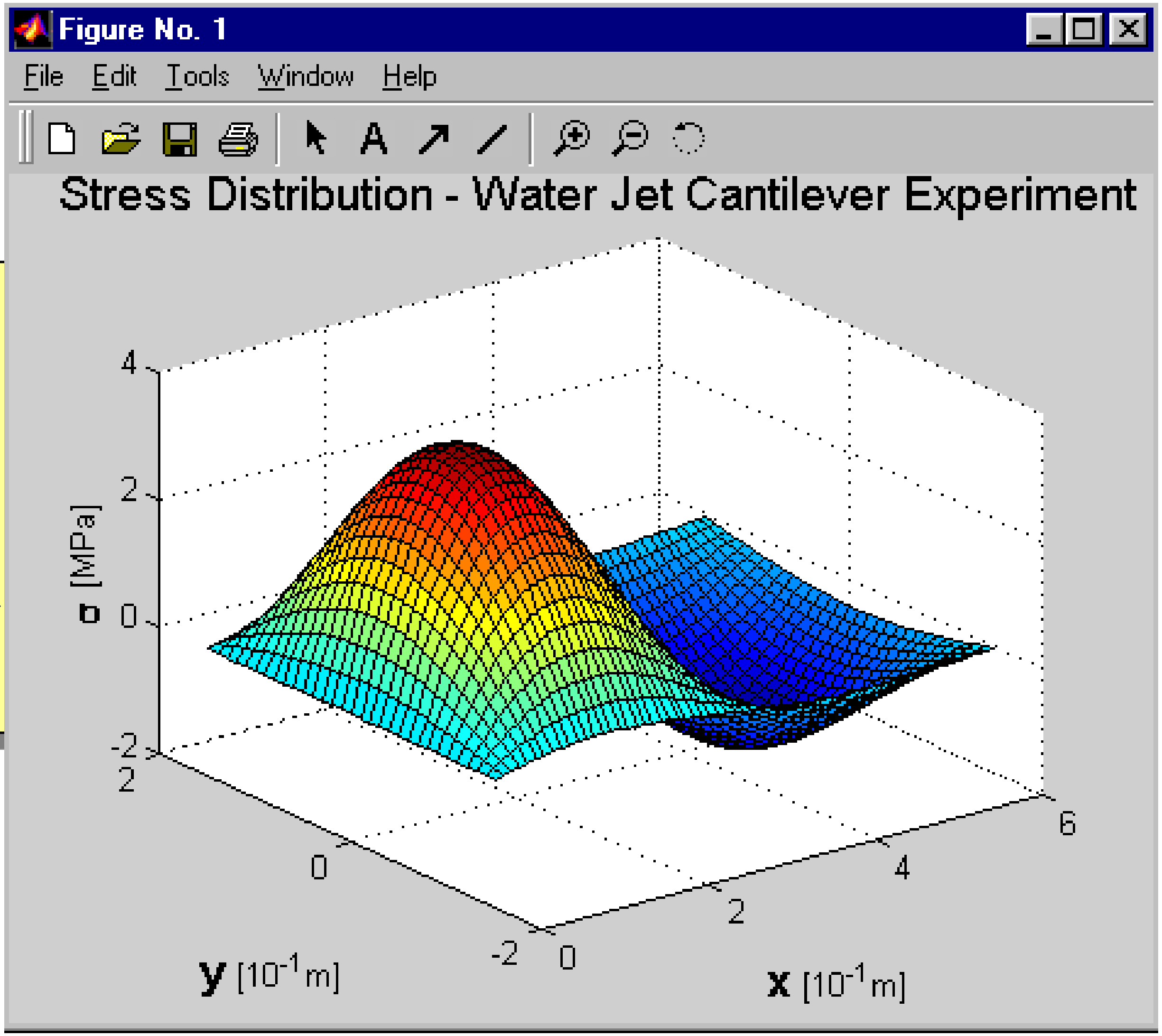


Figure No. 1
File
Edit
Tools
Window
Help
Stress Distribution - Water Jet Cantilever Experiment
σ [MPa]
4
2
0
-2
2
0
-2
0
2
4
6
y [10^{-1}m]
x [10^{-1}m]