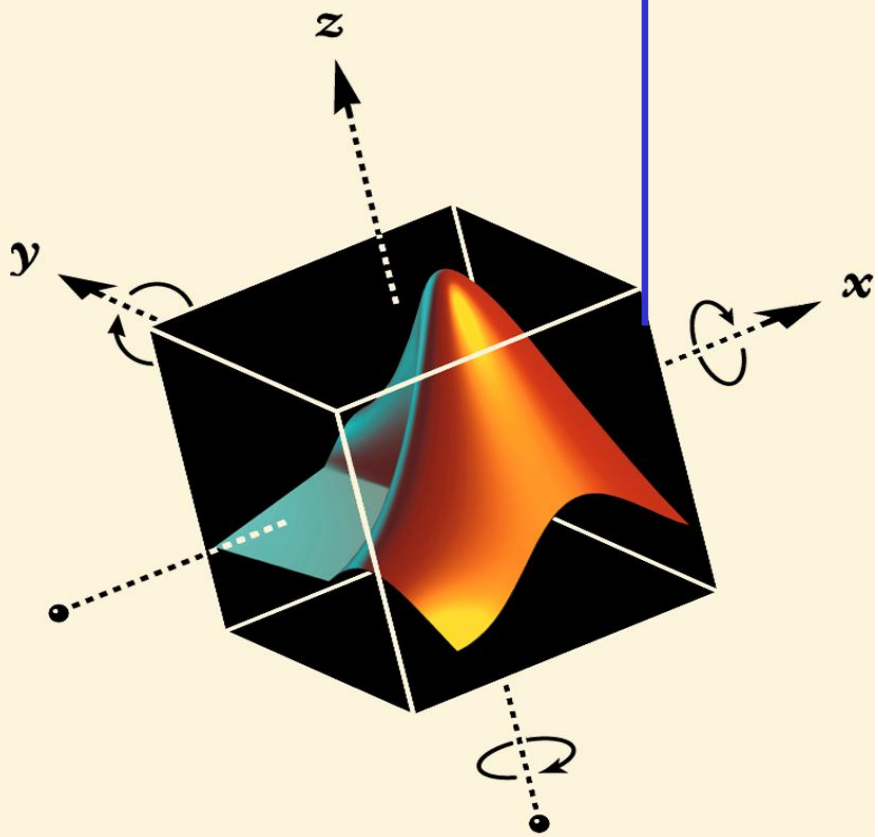
ANJUMAN-I-ISLAM'S
KALSEKAR TECHNICAL CAMPUS, NEW PANVEL
KNOWLEDGE RESOURCES & RELAY CENTRE (KRRC)

Introduction To Matlab

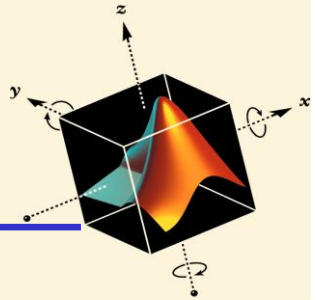
Mr. Zeeshan Ali, Asst. Professor

Department: B.E. Electronic & Telecommunication

Introduction to Matlab

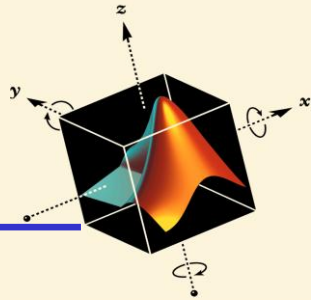


Why MATLAB?



- Industry standard software application
- Wealth of built-in functions and libraries
- Toolboxes (add-on software modules) – image and signal processing, control systems design, fuzzy logic, etc.
- Has own structured programming language
- Ease of application and testing (pre- and post processing without lots of programming and formatting)
- Platform independent

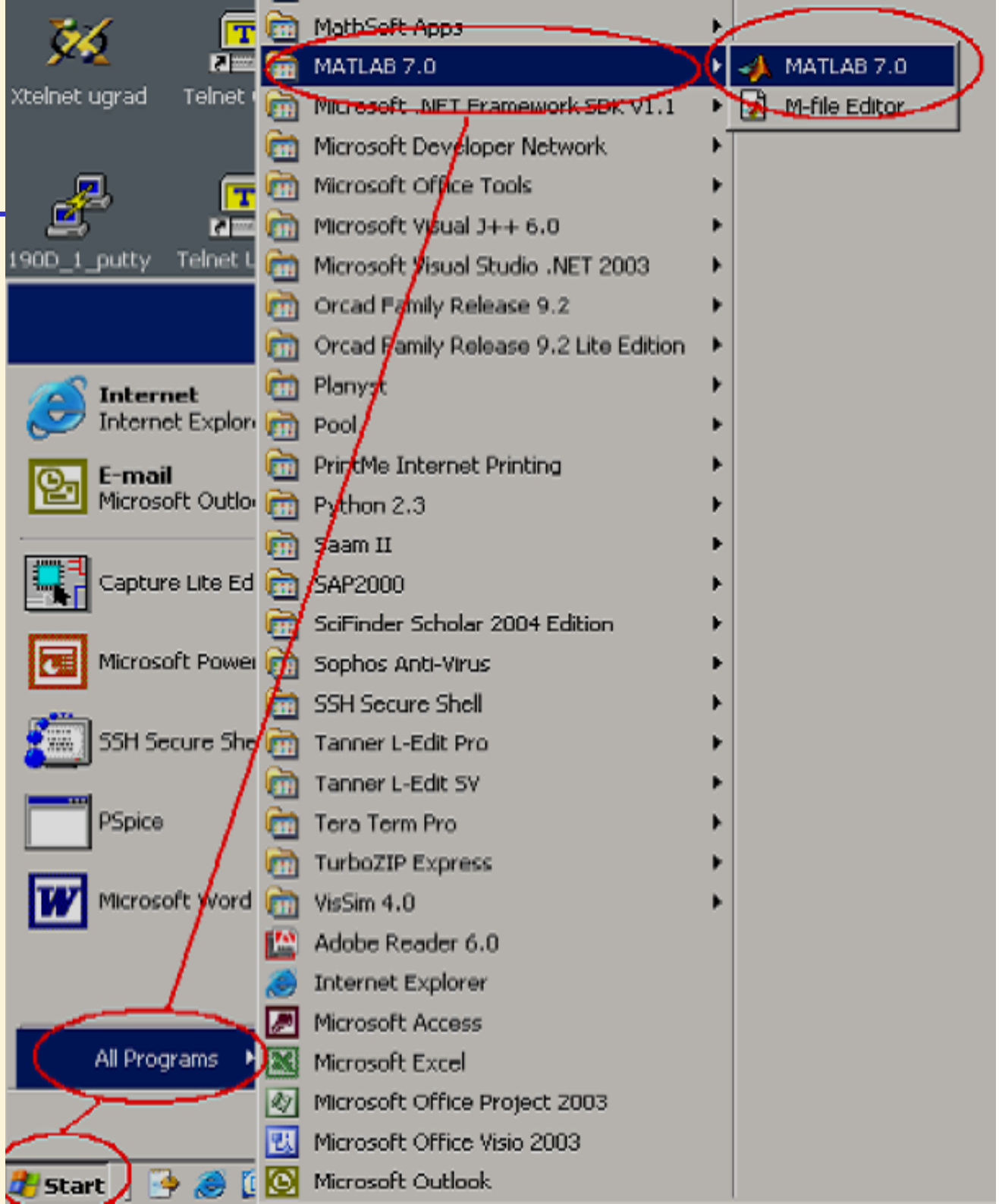
What is MATLAB?

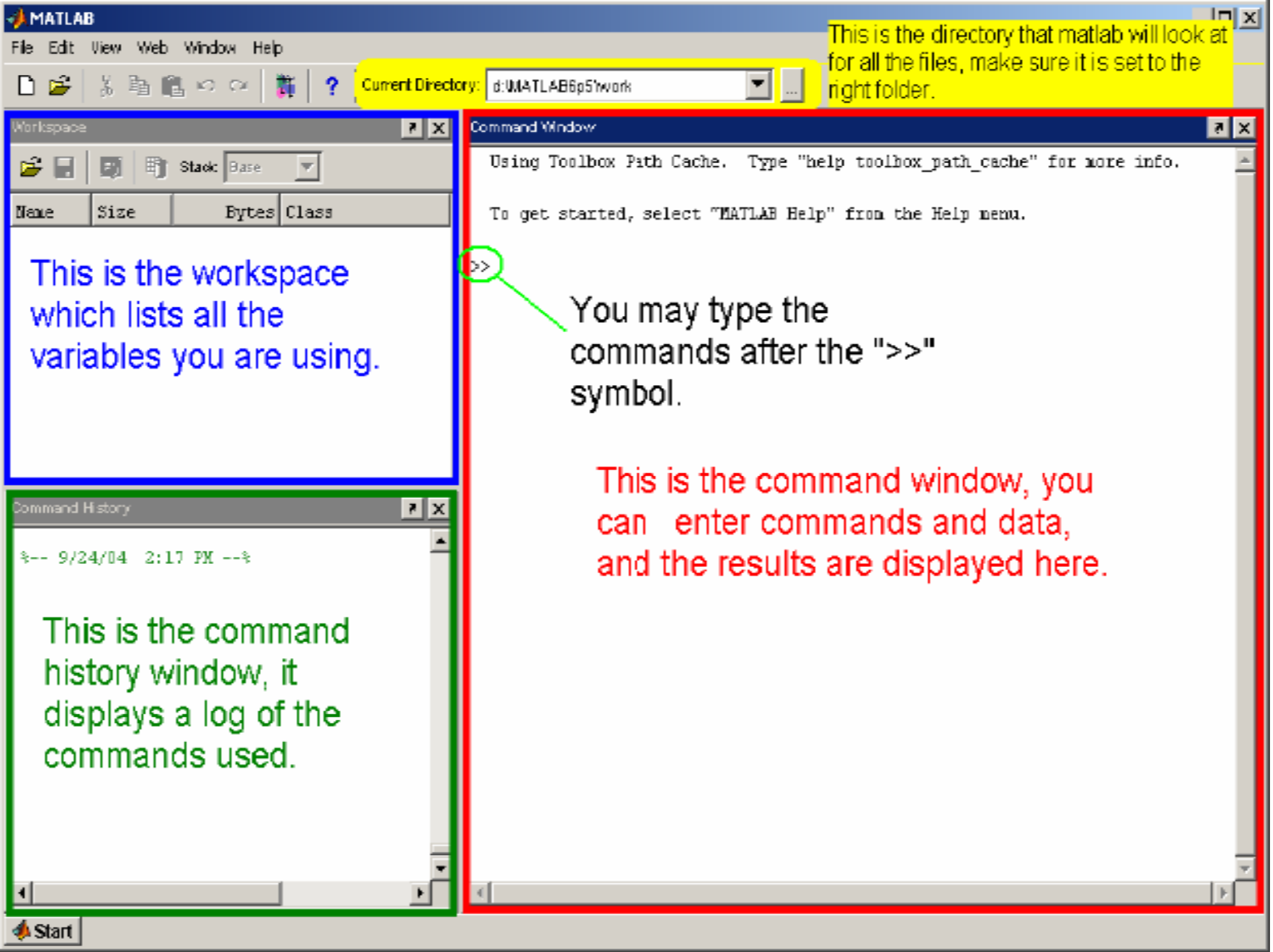


❖ MATLAB is a tool for doing numerical computations with matrices and vectors. It is very powerful and easy to use. In fact, it integrates computation, visualization and programming all together in an easy-to-use environment and can be used on almost all the platforms: windows, Unix, and HP-UX, Mac OS X and Solaris ,etc.

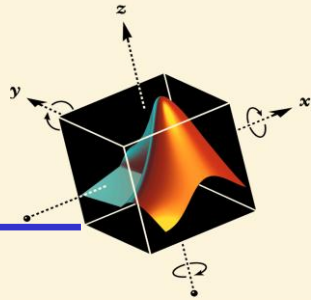
MATLAB Environment

- To start MATLAB:
  - START
  - PROGRAMS
  - MATLAB 7.0
  - MATLAB 7.0



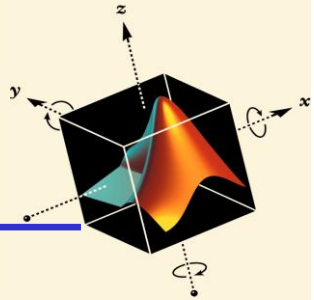

MATLAB
File Edit View Web Window Help
Current Directory: d:\MATLAB6p5\work
This is the directory that matlab will look at for all the files, make sure it is set to the right folder.
Workspace
Stack: Base
Name Size Bytes Class
This is the workspace which lists all the variables you are using.
Command Window
Using Toolbox Path Cache. Type "help toolbox_path_cache" for more info.
To get started, select "MATLAB Help" from the Help menu.
>>
You may type the commands after the ">>" symbol.
This is the command window, you can enter commands and data, and the results are displayed here.
Command History
%-- 9/24/04 2:17 PM --%
This is the command history window, it displays a log of the commands used.
Start

Desktop Tools



- Command Window
  - type commands
- Workspace
  - view program variables
  - `clear` to clear
  - double click on a variable to see it in the Array Editor
- Command History
  - view past commands
  - save a whole session using `diary`

General Commands

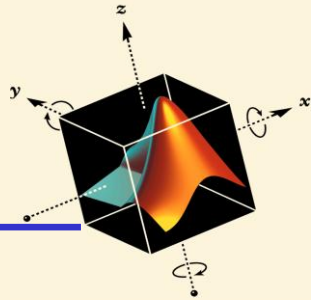


- help – gives description for a command

  syntax : help <command>
- who, whos – gives the variables present in the workspace
- clear – delete variables from workspace

  syntax : clear < var-name>; clear

Commands …

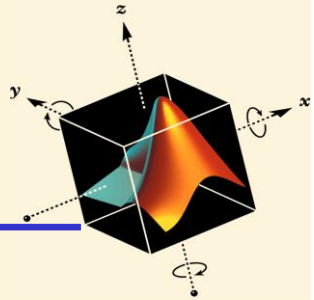


- edit – edit a matlab m-file

  syntax : edit <file>
- open – open a file by extension

  syntax : open <file>
- **clear --** remove all variables from memory
- **clc --** clear the command window
- **clf --** clear the graphics window

Matrices



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

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

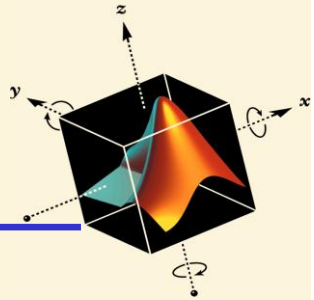
- a matrix `x = [1 2 3; 5 1 4; 3 2 -1]`

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

- transpose `y = x.'`

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

Matrices



- x(i,j) subscription

```
y=x(2,3)
y =
     4
```

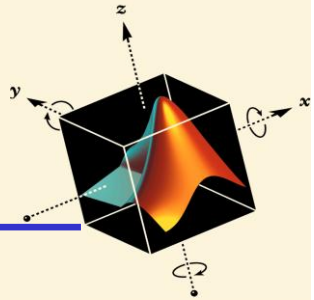
- whole row

```
y=x(3,:)
y =
     3     2    -1
```

- whole column

```
y=x(:,2)
y =
     2
     1
     2
```

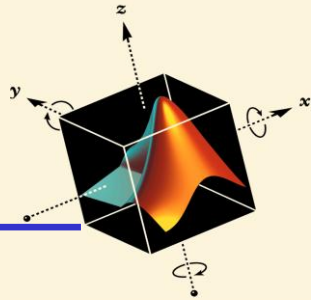
Concatenation of matrices



```
>>a =[ 1 2 ; 3 4]
      a =
           1     2
           3     4

>>  b=[a a+3;a+10 a+100]
      b =
           1     2     4     5
           3     4     6     7
          11    12   101   102
          13    14   103   104
```

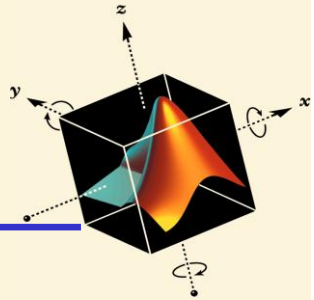
Sub - matricies



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

```
>> size(x)
     ans =
          3    3
```

Sub - matricies

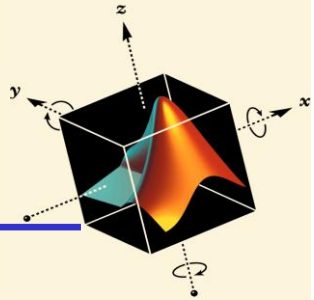


- **>> y = x(2,3)**
- **y =**
- **4**

- **>> y = x(3,:)**
- **y =**
- **3 2 -1**

- **>> y = x(:,2)**
- **y =**
- **2**
- **1**
- **2**

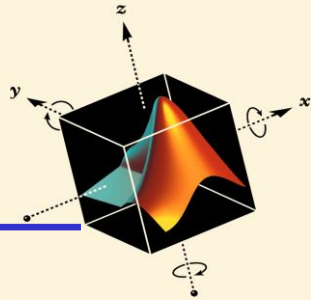
Matrices…



- E = [ ] % an empty matrix of 0-by-0 elements!

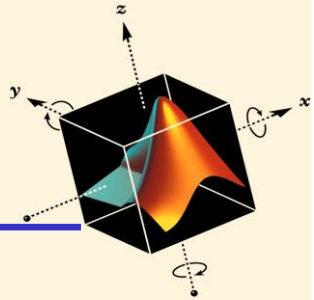
```
>> size(E)
    ans = 0 0
>> I = eye(3);   % the 3-by-3 identity matrix
    I =
          1 0 0
          0 1 0
          0 0 1
```

Matrices…



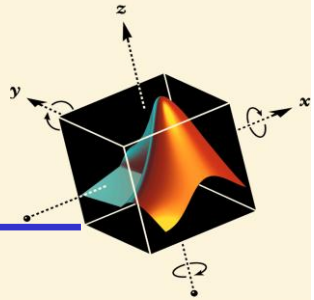
```
>> r = [1 3 -2]; R = diag(r)        % create a diagonal matrix with r on the
                                       diagonal
R =
     1 0 0
     0 3 0
     0 0 -2
>> A = [1 2 3; 4 5 6; 7 8 9];
>> diag(A)                          % extracts the diagonal entries of A
ans =
          1
          5
          9
```

Matrices…



```
D = rand(2,3)          % a matrix of random numbers; you will
                         get a different one!
D =
    0.0227 0.9101 0.9222
    0.0299 0.0640 0.3309
```

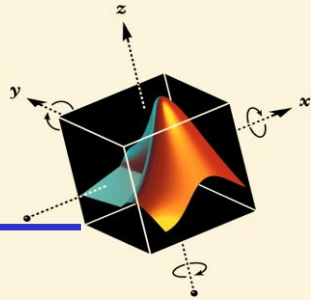
Operators (arithmetic)



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

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

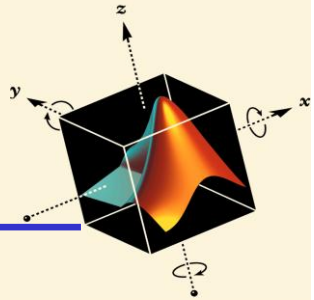
Operators (relational, logical)



==	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

Generating Vectors from functions



- zeros(M,N) MxN matrix of zeros

```
x = zeros(1,3)
x =
   0     0     0
```

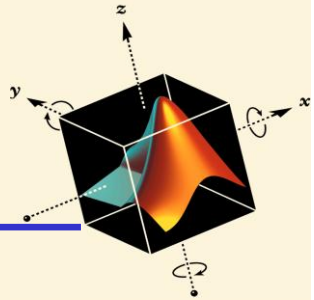
- ones(M,N) MxN matrix of ones

```
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
```

Operators



[ ] concatenation

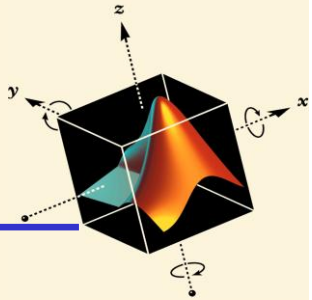
```
x = [ zeros(1,3) ones(1,2) ]
x =
     0   0   0   1   1
```

( ) subscription

```
x = [ 1 3 5 7 9]
x =
     1   3   5   7   9

y = x(2)
y =
     3
y = x(2:4)
y =
     3   5   7
```

Arrays Operations



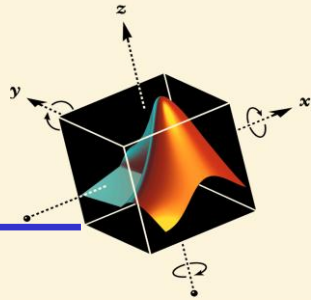
Element by Element Operation,

dot (.) operator:

- **C=A+B;**
- **C=A-B;**
- **C=A.*B;**
- **C=A./B;**
- **C=A.^B**

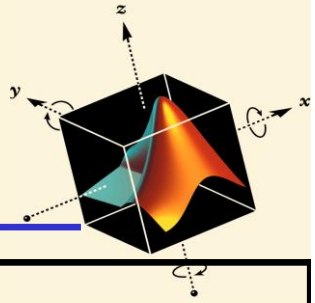
- **Note: A and B Need to be same size or B be scalar!**

Matrices and Operators



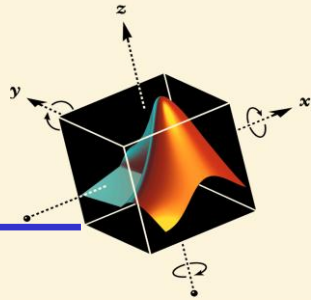
- **A=B+C; Same Size or one of them be scalar**
- **A=B-C; Same Size or one of them be scalar**
- **A=B*C; Matched Size or one of them be scalar**
- **A=k*B; k is scalar**
- **B=inv(A);**
- **d=det(A);**
- **B=A^2;**

Elementary MATLAB Commands



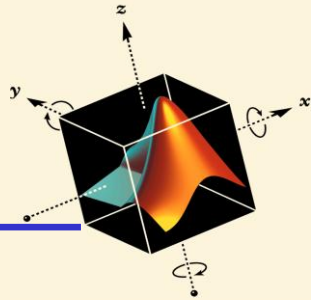
Function	Syntax (What to type in)	Example
Display	disp('Hello') / disp Hello disp (x)	Hello 5
Trig Functions	sin(13*pi/5)	0.9511
Square Root	sqrt(17)	4.1231
Remainder	rem(15,6)	3
Size	d=[1 2 3;4 5 6] size(d)	[2 3]
Clear	clear	(Clears all variables)
Clear Screen	clc	(Clears Screen)

Numerical Display



MATLAB Command	Display	Example
format short	4 decimal points	3.1416
format long	14 decimal points	3.14159265358979
format short e	4 decimal points	3.1416e+000
format long e	14 decimal digits	3.14159164358979 3e+000
format bank	2 decimal digits	3.14

Evenly spaced rows



- **V_name = start : step : end**

- >> a = 1:2:21

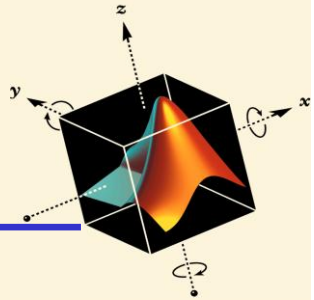
```
a =
     1     3     5     7     9    11
    13    15    17    19    21
```

- >> b = 10 : -1.5 : -2

```
b =

   10.0000    8.5000    7.0000    5.5000    4.0000    2.5000    1.0000   -
    0.5000   -2.0000
```

Linearly Spaced Vector



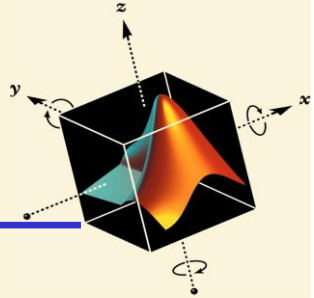
- linspace(x1,x2,n)
  - X1 – start
  - X2 – end
  - N – number of points between x1 and x2

- >> linspace(1,20,3)

  ans =

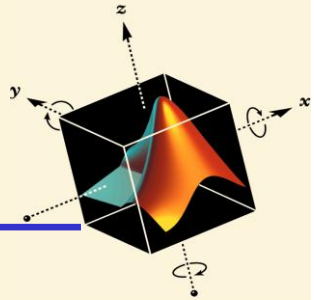
  1.0000 10.5000 20.0000

Examples



```
>> -2:3
ans = -2 -1 0 1 2 3
>> 0.2:0.5:2.4
ans = 0.2000 0.7000 1.2000 1.7000 2.2000
>>  A = [1 2 3; 4 5 6; 7 8 9];
>> diag(A)
ans = 1
        5
        9
>> 1.5:-0.5:-0.5
ans =1.5000 1.0000 0.5000 0 -0.5000\
>> linspace (1,100,10)
1    12    23    34    45    56    67    78    89   100
```

Important

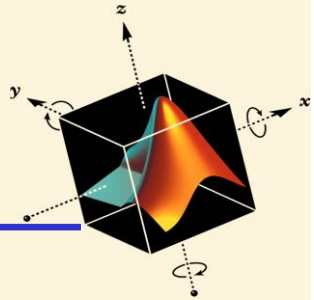


- X = [1 4 11 100]
- Y = [14; 200;-100]
- Z = [1.4 10.7 -1.1 20.9]

  - sum(x)
  - length(x)
  - min(y)
  - sign(y)
  - sort(y)
  - ceil (z)

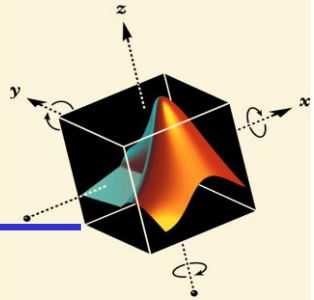
  mean(x)
  max(y)
  prod(x)
  round(z)
  size(x)
  floor (z)

Colon notation



Parts of vectors can be extracted by using a colon notation

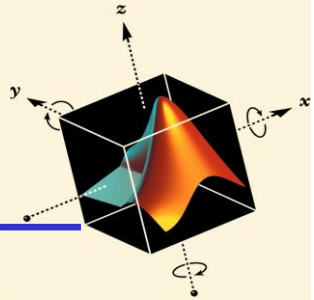
```
>> r = [-1:2:6, 2, 3, -2]      % -1:2:6 = -1 1 3 5
     r =-1 1 3 5 2 3 -2
```



- **v = max(A) v is a vector with the maximum value of the elements in each column of A**

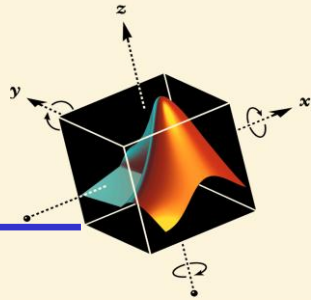
  or v is the maximum of all elements if A is a vector
- **v = min(A) ditto - with minimum**
- **v = sum(A) ditto - with sum**

Matlab Files (.m)



- Use predefined functions or write your own functions
- Reside on the `current directory` or the `search path`
  - add with File/Set Path
- Use the Editor/Debugger to edit, run

Program Documentation



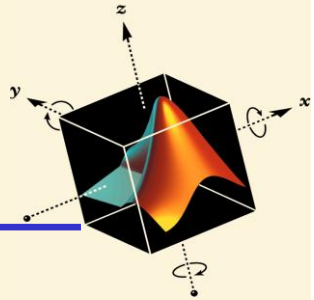
Comments!!!

- ***Program Documentation***

  You must include comments in the computer programs you turn in -- otherwise we will have great difficulty knowing what you are doing and to include comment we use ‘%’

- **Perhaps the most important thing to remember is semicolons (;) at the end of a line to suppress output**

Matlab Functions



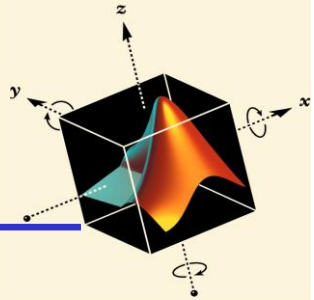
Input output functions

- R = input(‘Enter radius in meters - ’)
  - Number
  - String if written in ‘……’

- R = input(‘Enter your name - ’,’s’)
  - Takes only string

- disp()
  - DISP(X) displays the array, without printing the array name

Matlab Functions

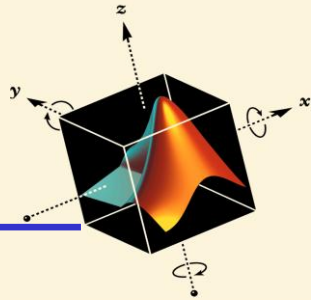


User Defined Functions

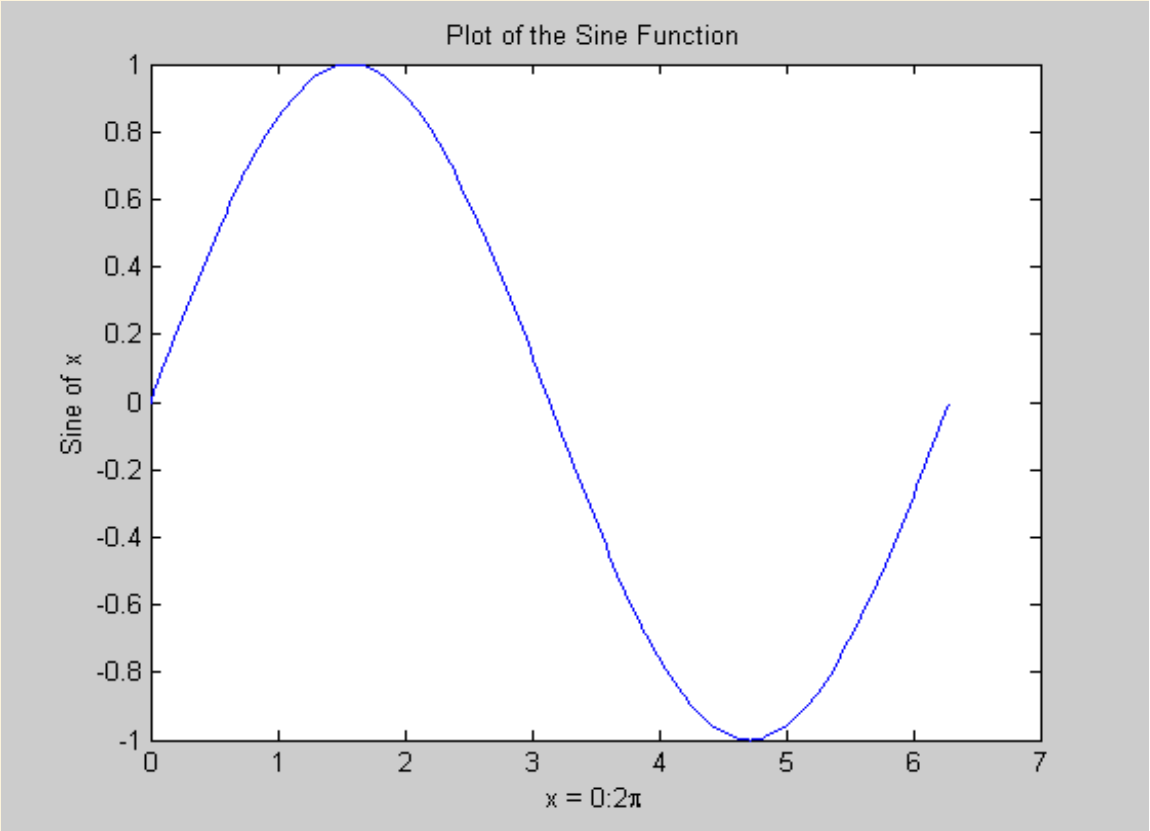
```
function f=myfunction(x,y)
   f=x+y;
```

- save it in myfunction.m
- call it with `y=myfunction(x,y)`

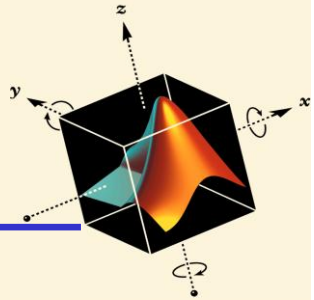
Matlab Graphics



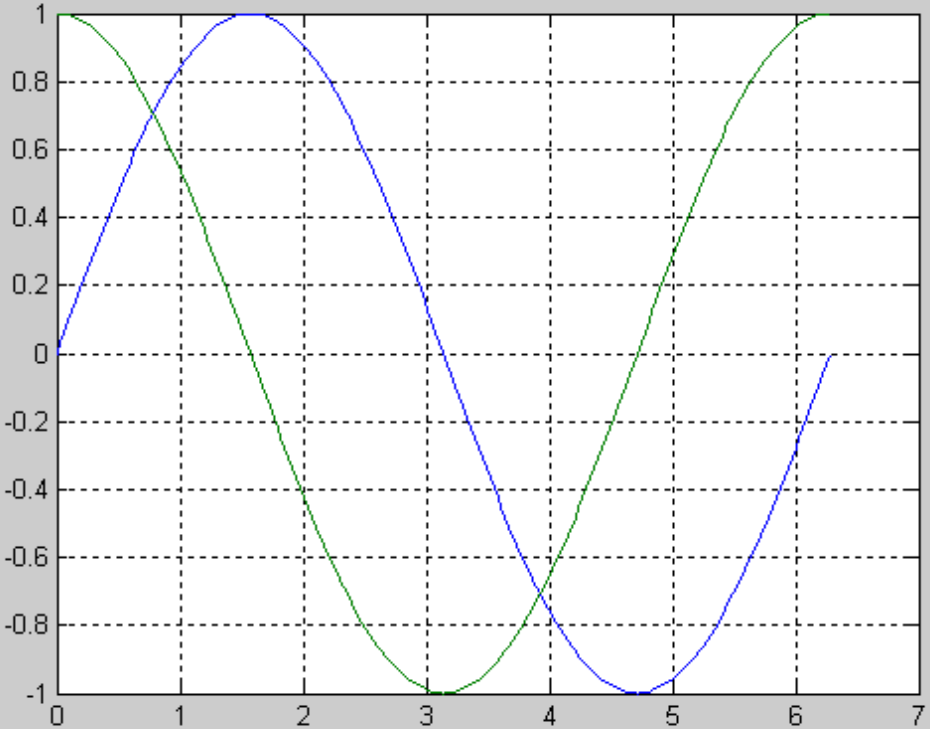

```
x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y)
xlabel('x = 0:2\pi')
ylabel('Sine of x')
title('Plot of the
   Sine Function')
```



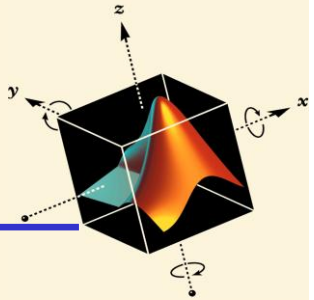
Multiple Graphs



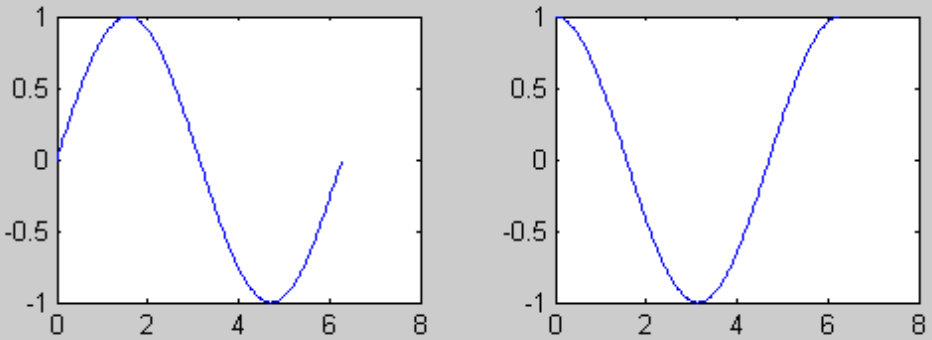

```
t = 0:pi/100:2*pi;
y1=sin(t);
y2=sin(t+pi/2);
plot(t,y1,t,y2)
grid on
```



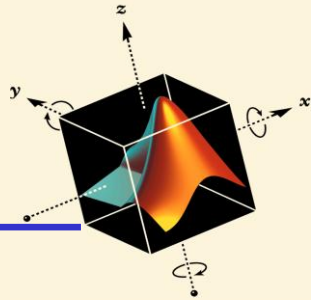
Multiple Plots



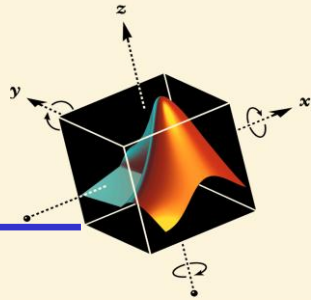

```
t = 0:pi/100:2*pi;
y1=sin(t);
y2=sin(t+pi/2);
subplot(2,2,1)
plot(t,y1)
subplot(2,2,2)
plot(t,y2)
```



Graph Functions (summary)



- plot linear plot
- stem discrete plot
- grid add grid lines
- xlabel add X-axis label
- ylabel add Y-axis label
- title add graph title
- subplot divide figure window
- figure create new figure window
- pause wait for user response



Thank You