INTRODUCTION TO SCIENTIFIC & ENG. COMPUTING
BIL 108E, CRN 44448

Week.3

Dr. Feyzi HAZNEDAROĞLU

İstanbul Teknik Üniversitesi - İnşaat Fakültesi
Room : 250A, e-mail :haznedar@itu.edu.tr

04-03-2011

ARRAYS

ARRAYS → (diziler)

- Matlab stores all types of variables as in the form of an array.
- A single element array is called a scalar.
- An array with one column or row is called a vector.
- An array with m rows and n columns, where *m, n* ≠ 1 is called a matrix.

- 3 way to determine ARRAYS in MatLAB:

  x = start : increment : end

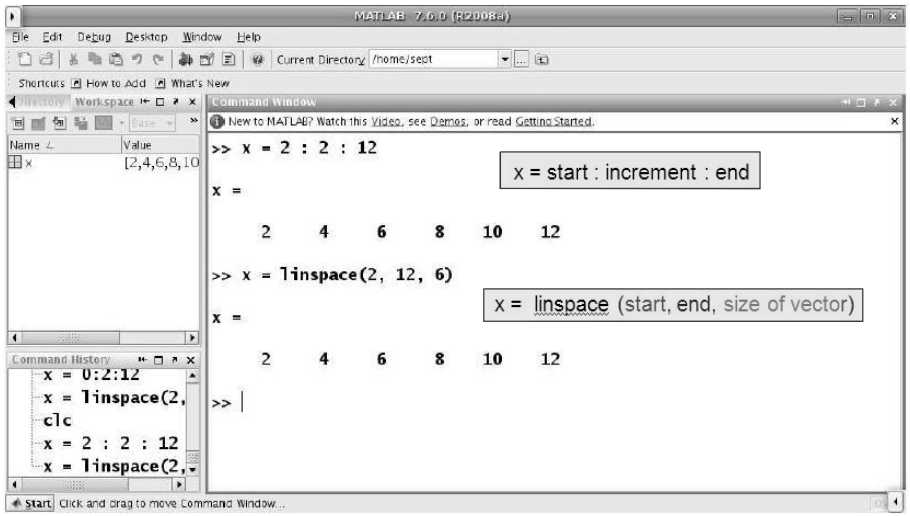
  x = linspace (start, end, size of vector)

  logspace

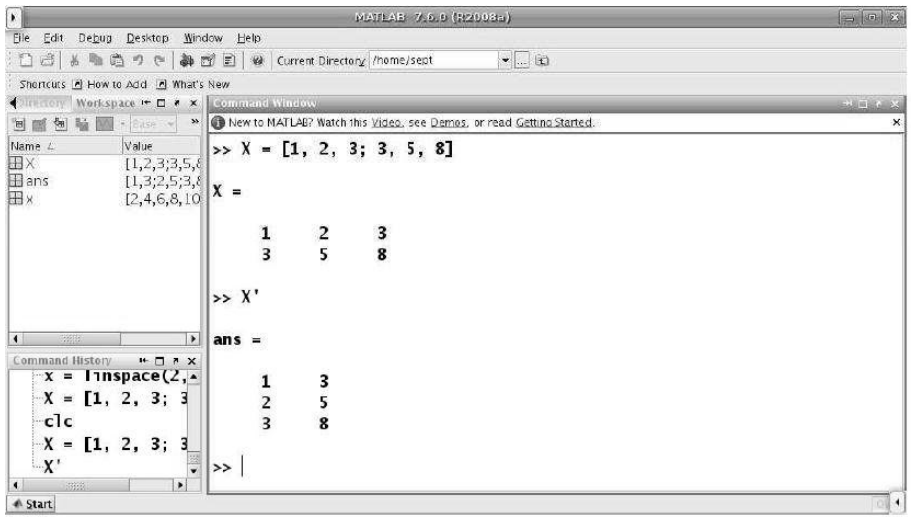
  $X = [a_{11}, a_{12}, a_{13}; a_{21}, a_{22}, a_{23}]$

  x = x' (transpose of x)

ARRAYS



ARRAYS



VECTORS

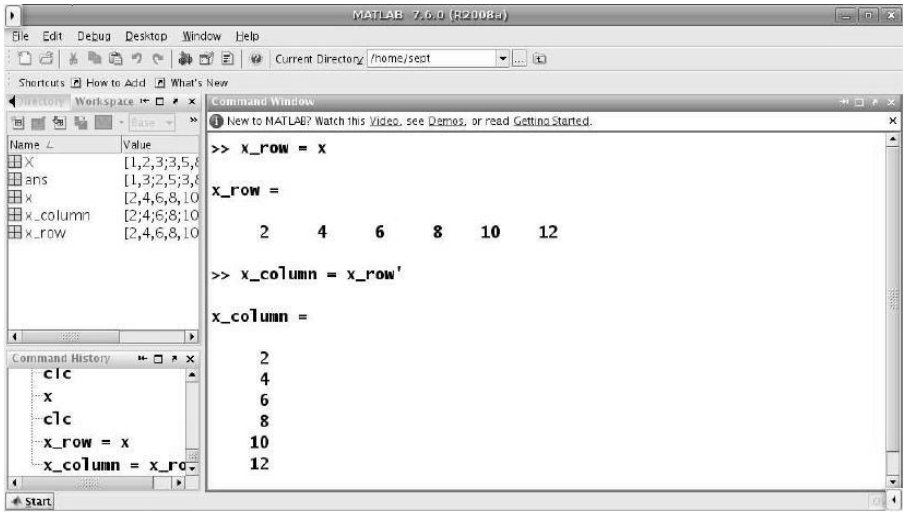
ROW VECTOR

$$\mathbf{a} = \begin{pmatrix} a_1 & a_2 & \dots & a_n \end{pmatrix}$$

COLUMN VECTOR

$$\mathbf{a} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}$$

VECTORS



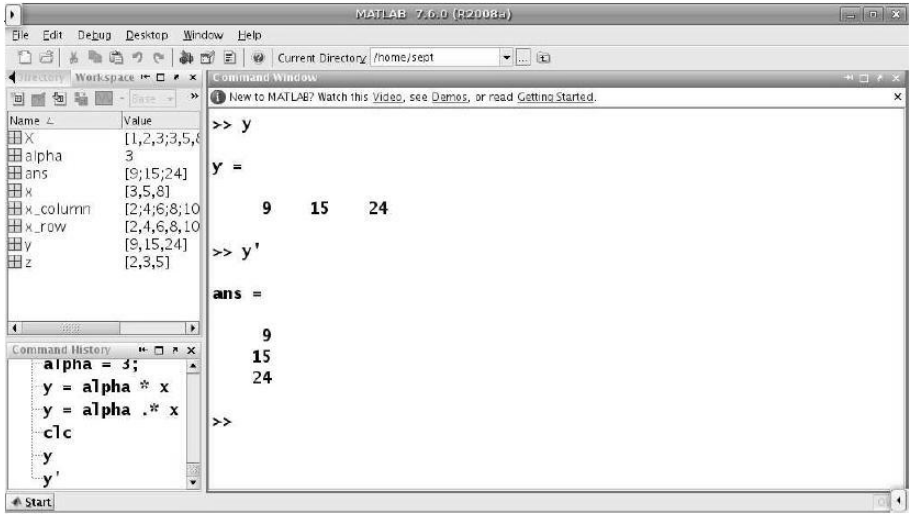
TRANSPOSE OF A VECTOR

TRANSPOSE OF A VECTOR

$$\mathbf{a} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}$$

$$\mathbf{a}^{\mathsf{T}} = \begin{pmatrix} a_1 & a_2 & \dots & a_n \end{pmatrix}$$

TRANSPOSE OF A VECTOR



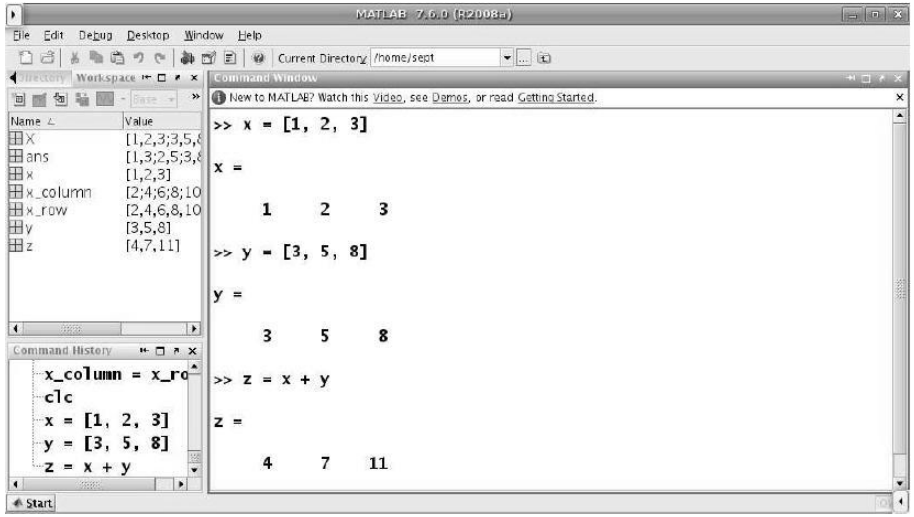
VECTOR ADDITION AND SUBTRACTION

Addition and subtraction are element-by element operations

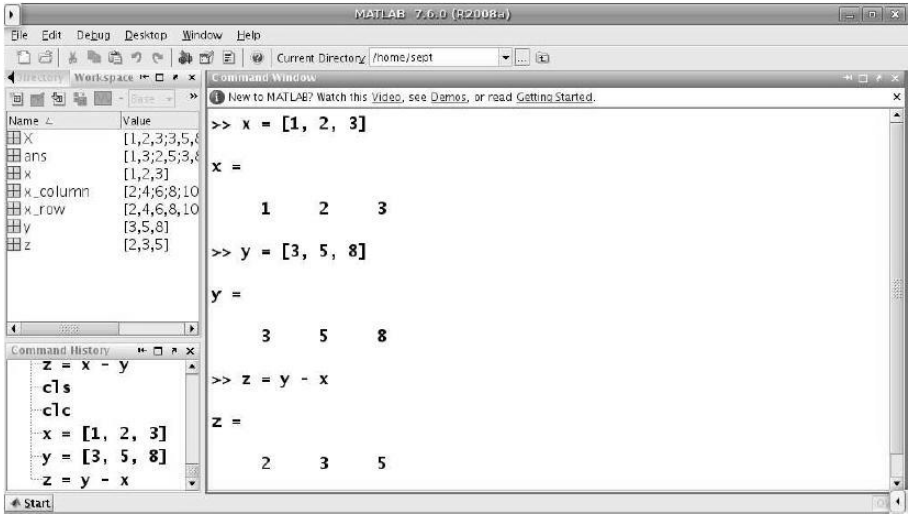
- $c = a + b,\ c_i = a_i + b_i,\ i = 1, 2, ..., n$
- $d = a - b,\ d_i = a_i - b_i,\ i = 1, 2, ..., n$

$$\mathbf{a} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{pmatrix} \pm \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix} = \begin{pmatrix} a_1 \pm b_1 \\ a_2 \pm b_2 \\ \vdots \\ a_m \pm b_m \end{pmatrix}$$

VECTOR ADDITION AND SUBTRACTION



VECTOR ADDITION AND SUBTRACTION



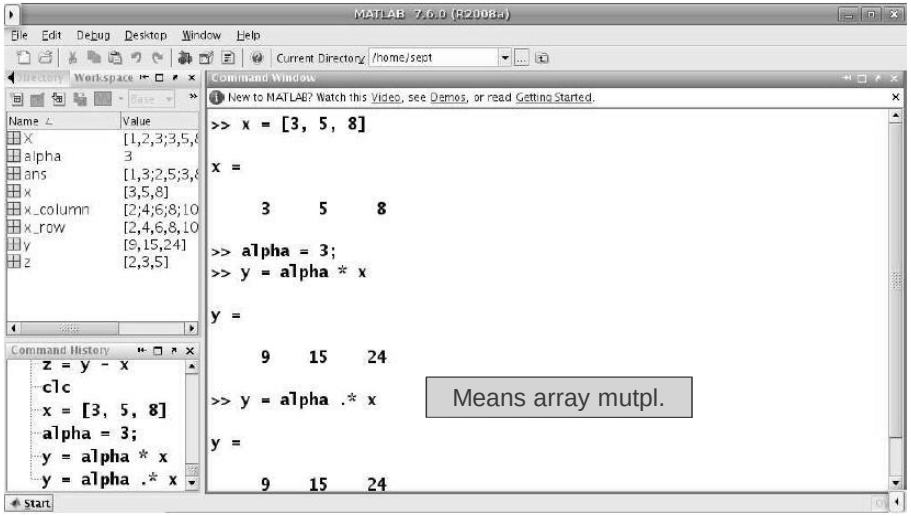
MULTIPLICATION BY SCALAR

Multiplication by a scalar involves multiplying each element in the vector by the scalar:

- $b = \alpha \times (a_i) = (\alpha \times a_i)$

$$\mathbf{b} = \alpha \times \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} \alpha \times a_1 \\ \alpha \times a_2 \\ \vdots \\ \alpha \times a_n \end{pmatrix}$$

MULTIPLICATION BY SCALAR



OPERATIONS ON ARRAYS

(./) and **(.*)** called DOT PRODUCT

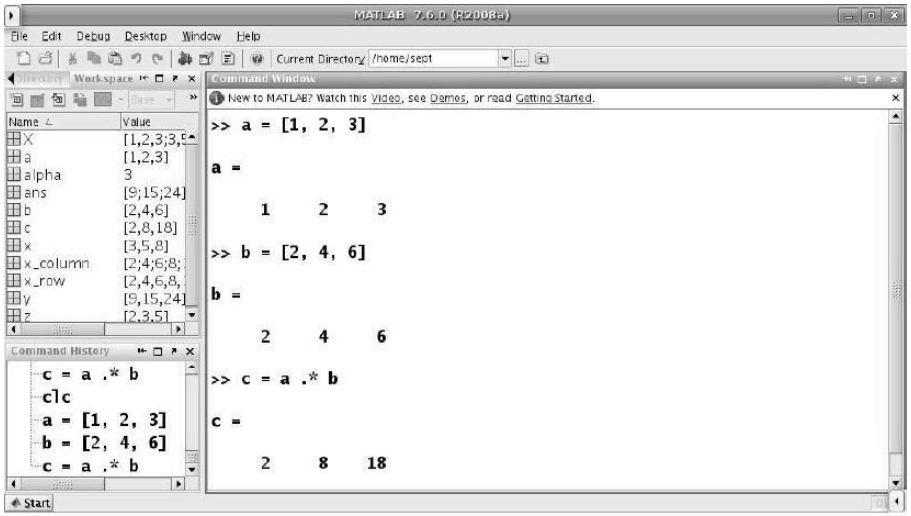
Multiplication and Division

DOT PRODUCT

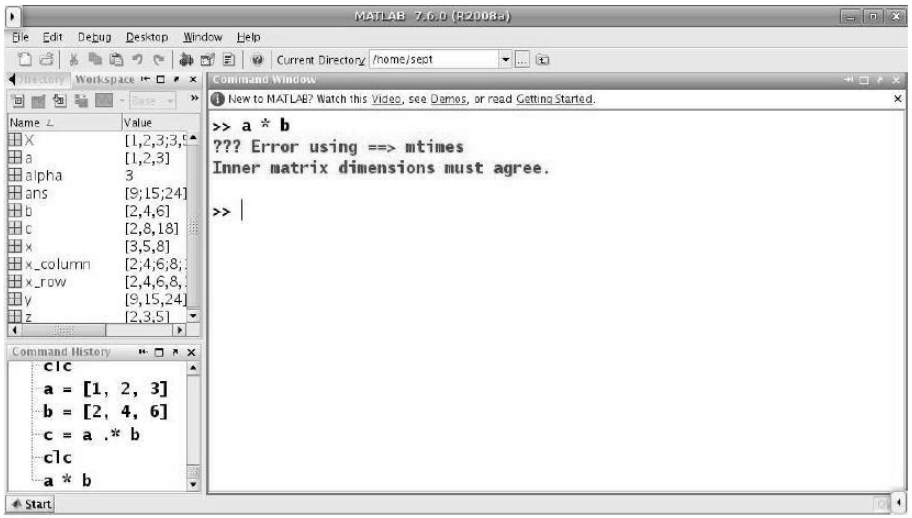
$C = A .* B = (a_1 \times b_1, a_2 \times b_2, \ldots, a_n \times b_n)$

$C = A ./ B = (a_1/b_1, a_2/b_2, \ldots, a_n/b_n)$

OPERATIONS ON ARRAYS



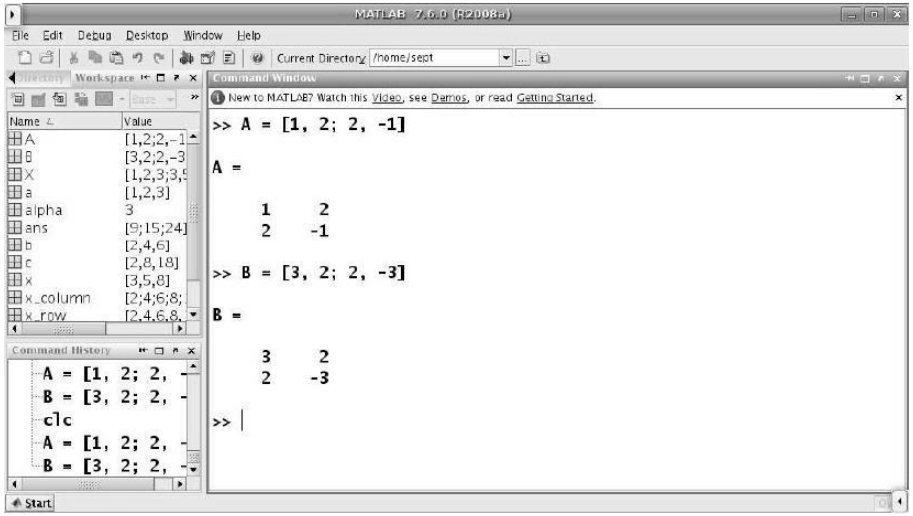
OPERATIONS ON ARRAYS

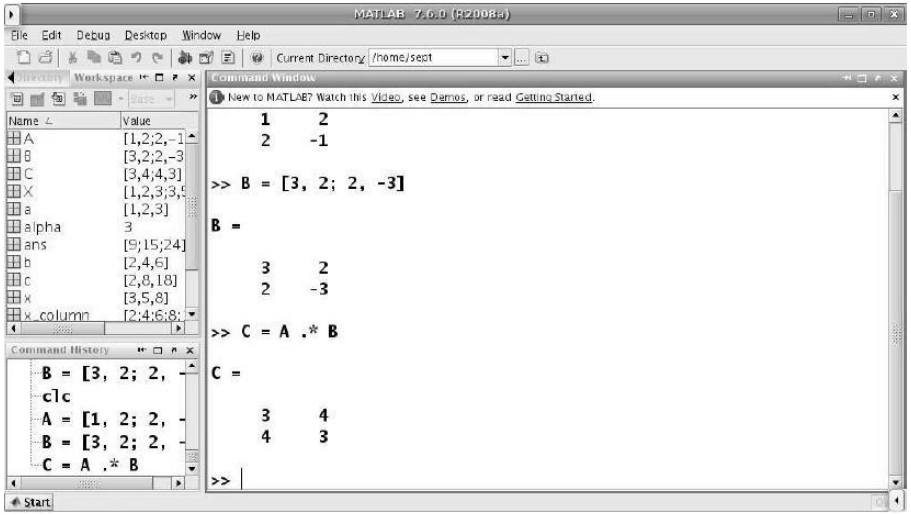


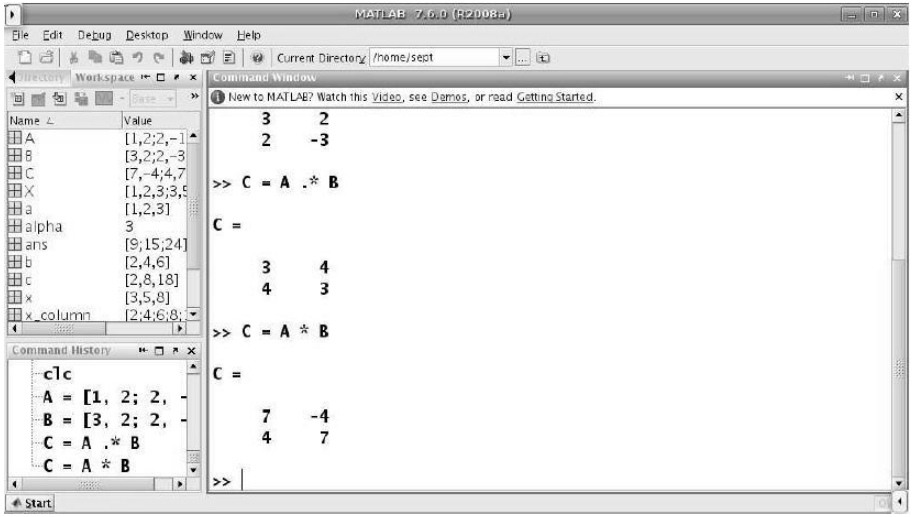
MATRIX MULTIPLICATION

$$c_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj}$$

$$i = 1, 2, \ldots, m, \; j = 1, 2, \ldots, n$$







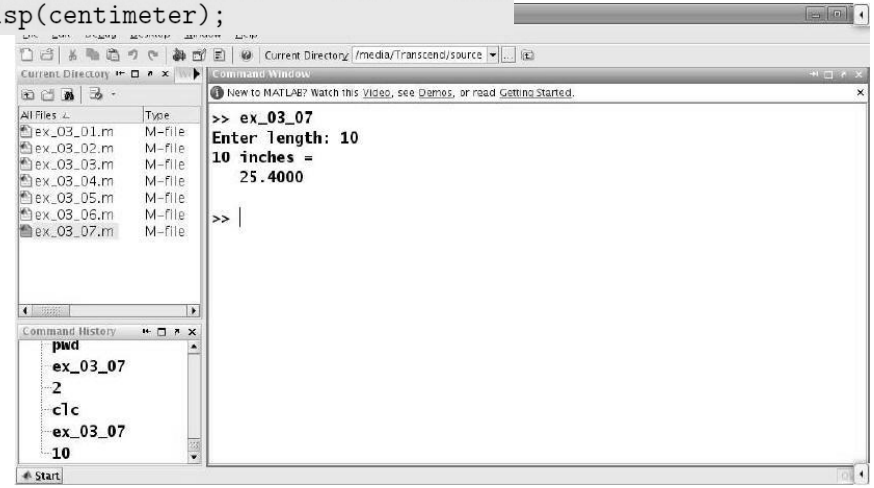
The input statement

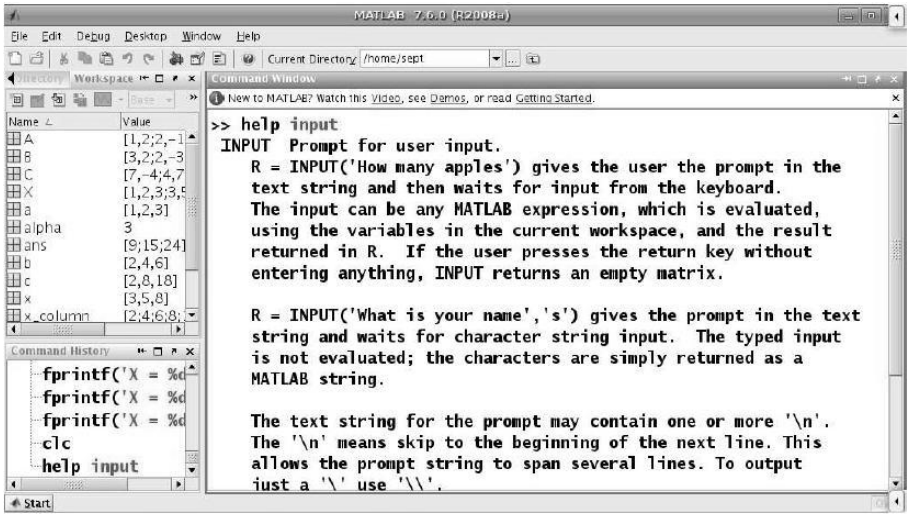
variable = input(' *prompt* ')

```
inch = input('Enter length: ');
centimeter = inch * 2.54;
disp([num2str(inch), ' inches = ']);
disp(centimeter);
```



```
inch = input('Enter length: ');
centimeter = inch * 2.54;
disp([num2str(inch), ' inches = ']);
disp(centimeter);
```





input

- The prompt message prompts for the value(s) to be entered. It must be enclosed in apostrophes (single quotes).
- A semicolon at the end of the input statement will prevent the value entered from being immediately echoed on the screen.
- You normally do not use input from the command line, since you shouldn't need to prompt yourself in command – line mode.
- Vectors and matrices may also be entered with input, as long as you remember to enclose the elements in square brackets.
- You can enter an expression in response to the prompt – for example, $a + b$ (as long as a and b have been defined) or rand(5). When entering an expression in this way, don't include a semicolon (it is not part of the expression).

Yansıtmak

USING SYSTEM COMMANDS

Executing operating system commands

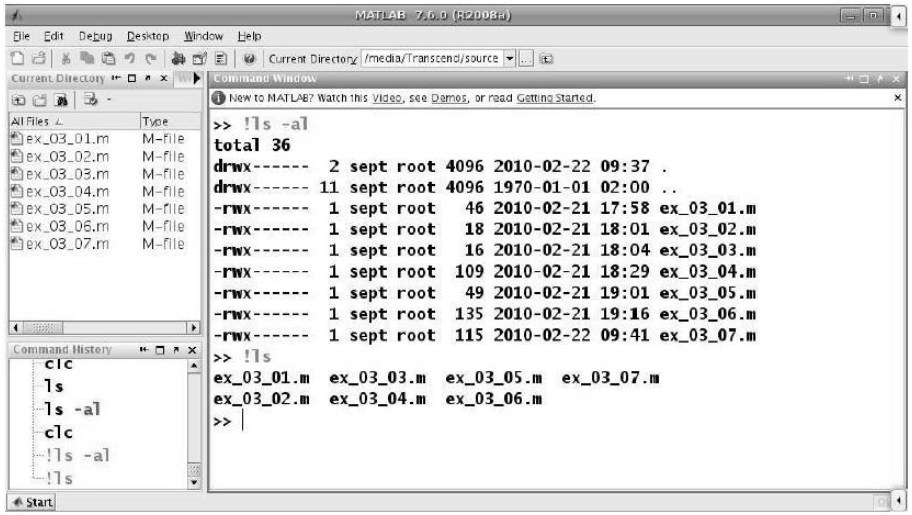
Example;

```
!time
```

Changing computer time

USING SYSTEM COMMANDS

- file listing (UNIX).



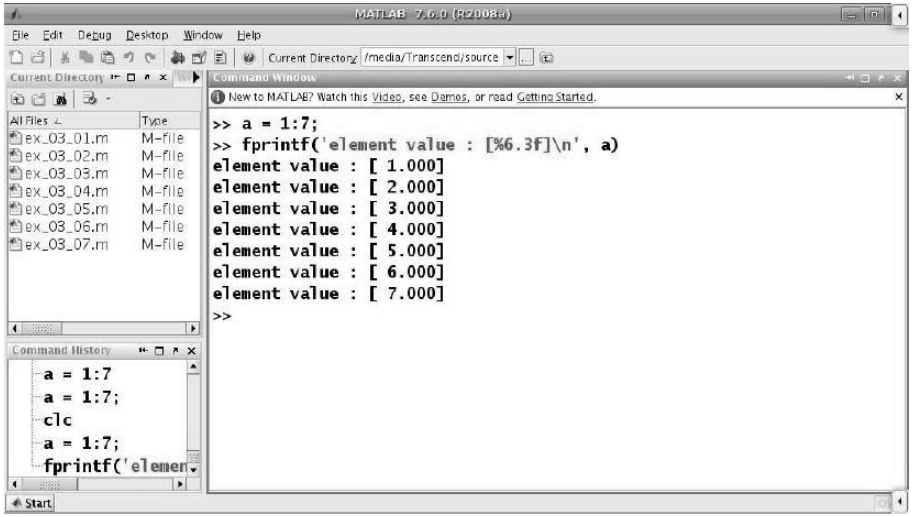
fprintf

- `fprintf`
- `fprintf('formatstring', listofvariables)`
- `fprintf('filename', 'formatstring', listofvariables)`

Example;

```
fprintf( 'myfile', '%f', x )
```


fprintf



GENERAL FILE I/O

- `fopen`
- `fclose`
- `fread`
- `fwrite`
- `fseek`

GOOD PROGRAMMING STYLE

Here are some hints on how to improve your programming style:

- You should make liberal use of comments, both at the beginning of a script to describe briefly what it does and any special methods that may have been used, and throughout the coding to introduce different logical sections.
- The meaning of each variable should be described briefly in a comment when it is initialized. You should describe variables systematically, for example, in alphabetical order.
- Blank lines should be freely used to separate sections of coding (e.g., before and after loop structures).

GOOD PROGRAMMING STYLE

cont'd,

- Coding (i.e., statements) inside structures (fors, ifs, whiles) should be indented (tabulated) a few columns to make them stand out.
- Blank spaces should be used in expressions to make them more readable – for example, on either side of operators and equal signs. However, blanks may be omitted in places in complicated expressions where this may make the logic clearer.

INTRODUCTION TO GRAPHICS

PRESENTING AND VISUALIZING GRAPHICAL DATA

A picture is worth a thousand words.

- 2D plotting
- 3D plotting

SIMPLE 2D GRAPHICS

`plot` statement

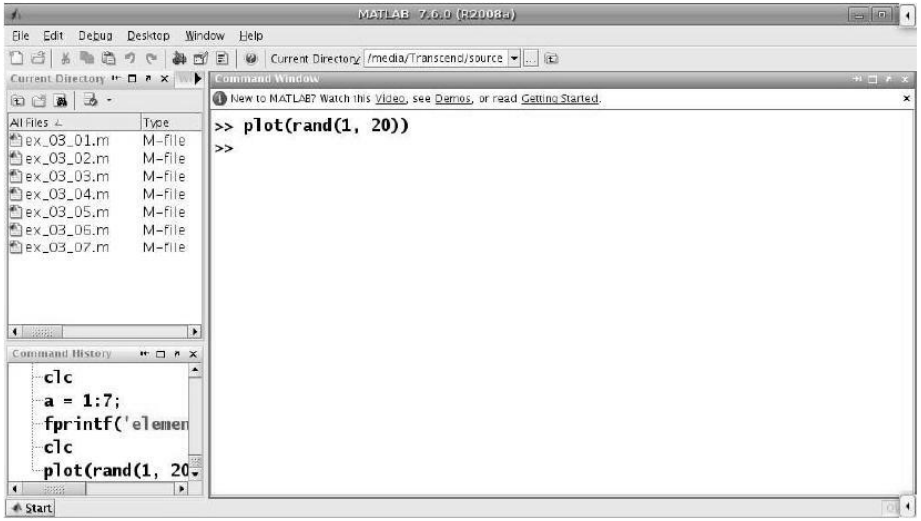
In its sipmplest form `plot` takes a single vector argument.

Example;

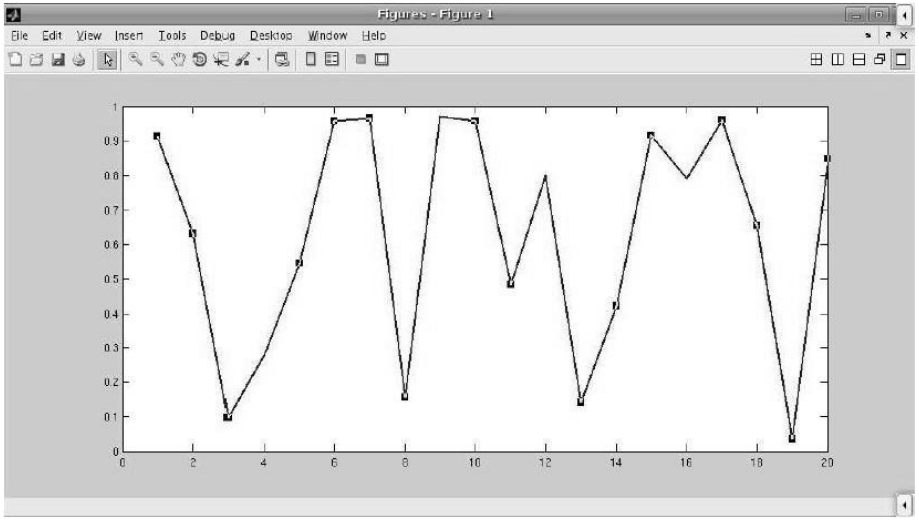
```
plot(rand(1, 20))
```

Plots 20 random numbers against the integers 1-20.

SIMPLE 2D GRAPHICS



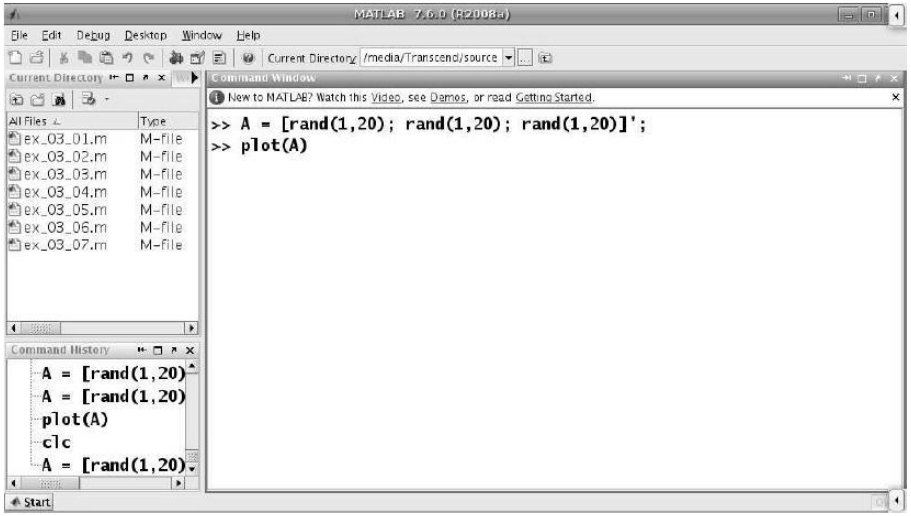
SIMPLE 2D GRAPHICS



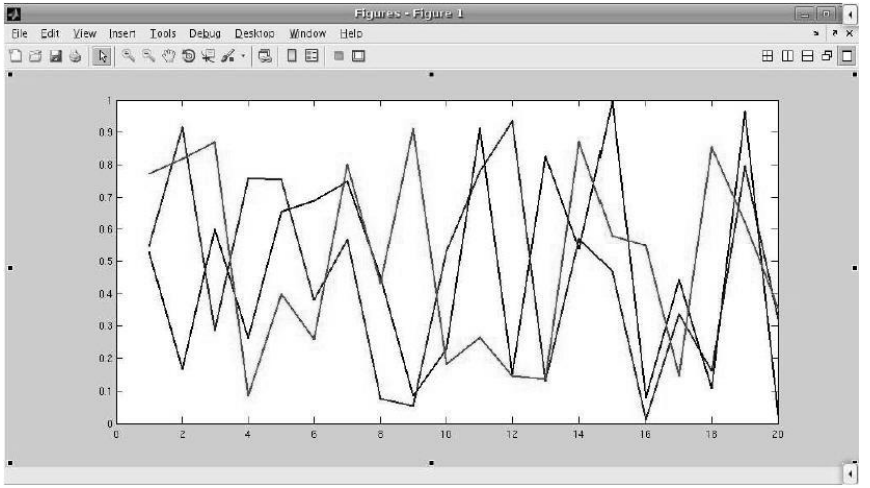
SIMPLE 2D GRAPHICS

- If the argument is a matrix , its columns are plotted against element indexes.
- Axes are automatically scaled and drawn to include the minimum and maximum data points.

SIMPLE 2D GRAPHICS



SIMPLE 2D GRAPHICS



SIMPLE 2D GRAPHICS

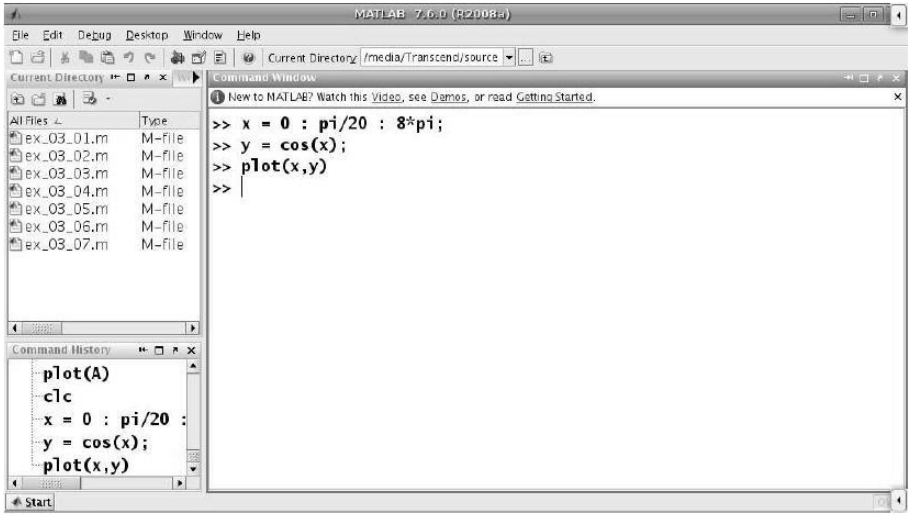
`plot(x, y)`

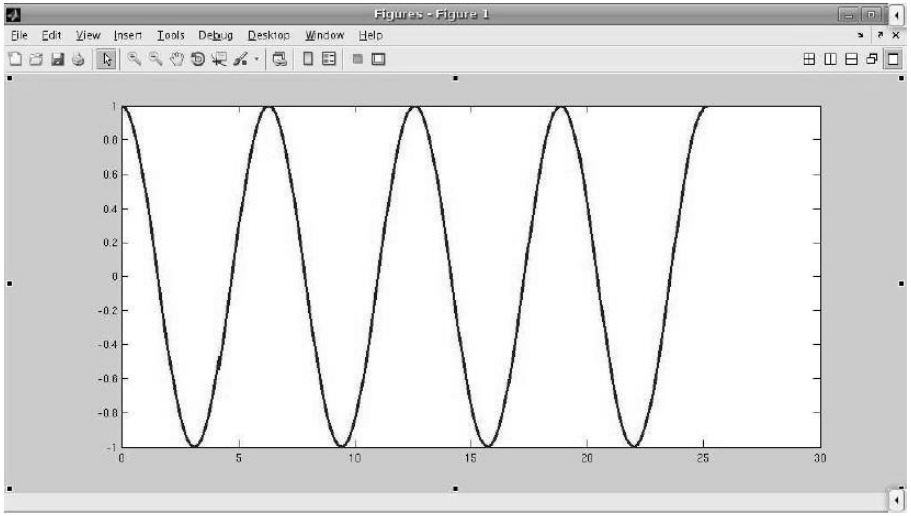
x and y are vectors of the same length. Example;

```
x = 0 : pi/20 : 8*pi;
y = cos(x);
plot(x, y)
```

i th point coordinates are x_i, y_i

SIMPLE 2D GRAPHICS





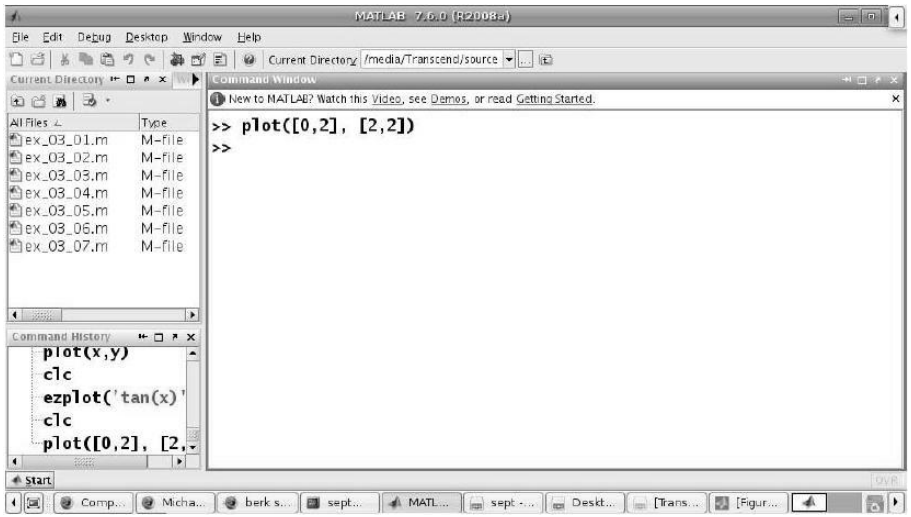
DRAWING STRAIGHT LINES

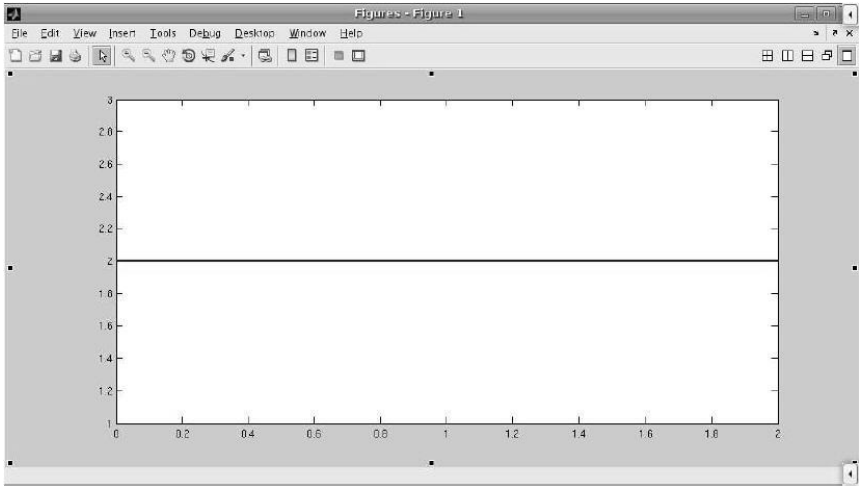
Straight-line graphs are drawn by giving the x and y coordinates of the end points by two vectors.

```
plot([0 2], [2 2])
```

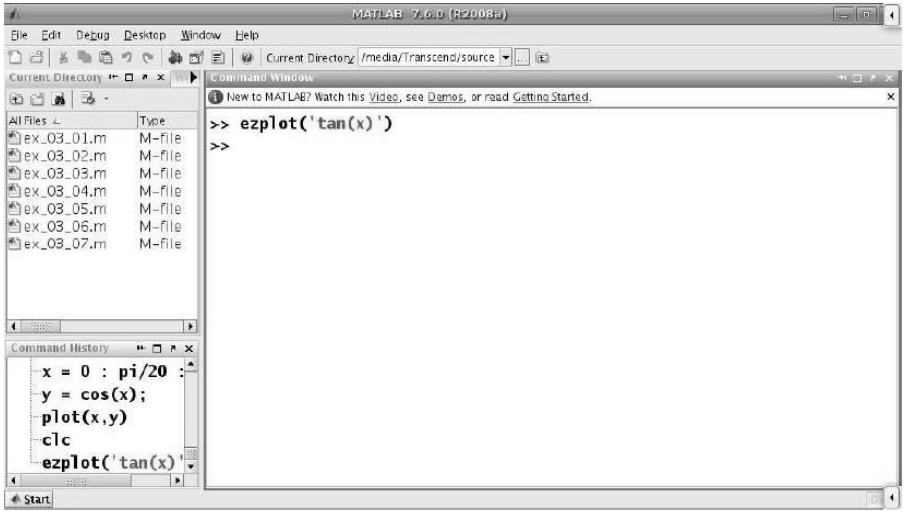
Matlab has a set of easy-to-use plotting commands.

```
ezplot('tan(x)')
```

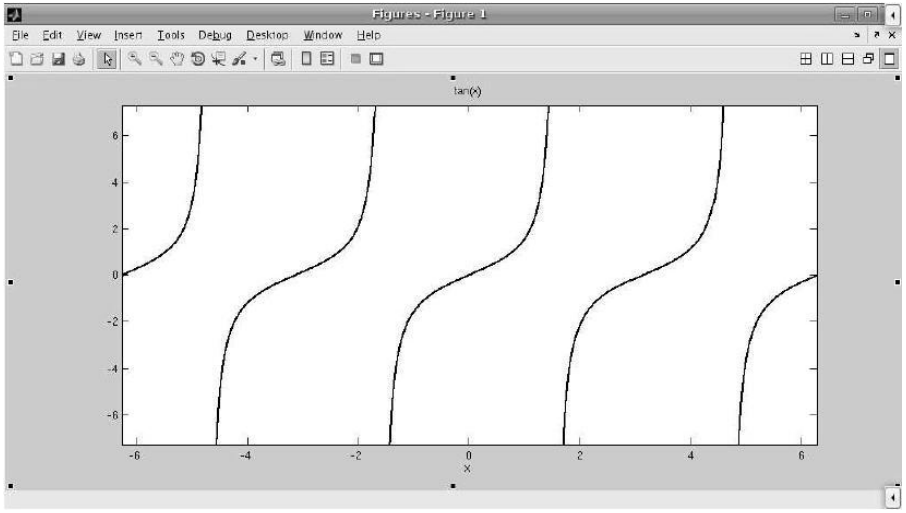





SIMPLE 2D GRAPHICS



SIMPLE 2D GRAPHICS



LABEL SETTINGS

Graphs may be labeled with the following statements:

- `gtext('text')` writes a string in the graph window. Text may be placed also with **Tools–Edit Plot** from the figure window.
- `grid` add/removes grid lines.
- `text(x, y, 'text')` writes text at the point specified by x and y.
- `title('text')` writes the text as a title at the top of the graph.
- `xlabel('horizontal')` labels the x-axis.
- `ylabel('vertical')` labels the y-axis.

MULTIPLE PLOTS

Multiple plots on the same axis

- Use `hold` to keep the current plot on the axes. Released with either `hold off` or `hold`.
- Use `plot` with multiple arguments.
  `plot(`x_1`, `y_1`, `x_2`, `y_2`)`
- Use `plotyy` to have independent y-axis on the left and on the right
  `plotyy(`x`, `y_1`, `x`, `y_2`)`
- Use the form `plot(x, y)`, where x and y may be both matrices or one may be a vector and one a matrix.

LINE STYLES, MARKERS AND COLOR

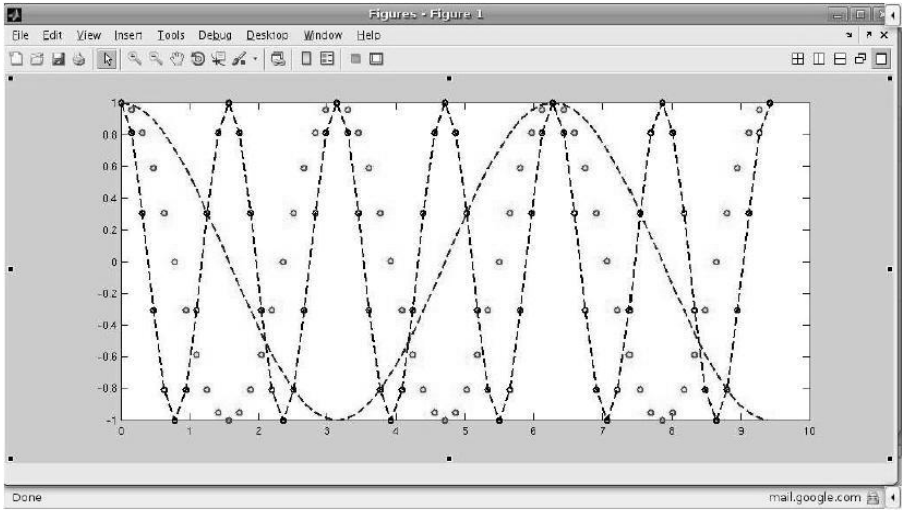
Line style, markers and color for a graph may be selected with a string argument to plot.

Example;

```
x = 0 : pi/20 : 3*pi;
y = cos(x);
plot(x, y, '--')
hold
plot(x, cos(2*x), 'o')
plot(x, cos(4*x), 'om--')
```

Available color symbols **c, m, y, k, r, g, b, w**

MULTIPLE PLOTS



AXIS SETTINGS

Axis limits can be overriden with

`axis([xmin, xmax, ymin, ymax])`

- Sets the scaling on the current plot.
- Use `Inf` or `-Inf` for the autoscaled limit.
- Use `axis auto` to return to the automatic axis scaling.
- `v = axis` returns the current axis scaling in the vector v.
- Use `axis manual` to freeze current scaling, so subsequent plots use the same limits.
- Use `axis equal` to make equal unit length on both axis. The effect is undone with `axis normal`.
- Turn axis labeling and tick marks with `axis off` and `axis on`.

MULTIPLE PLOTS - subplot

To show a number of plots in the same figure window use `subplot` function.

`subplot(m, n, p)`

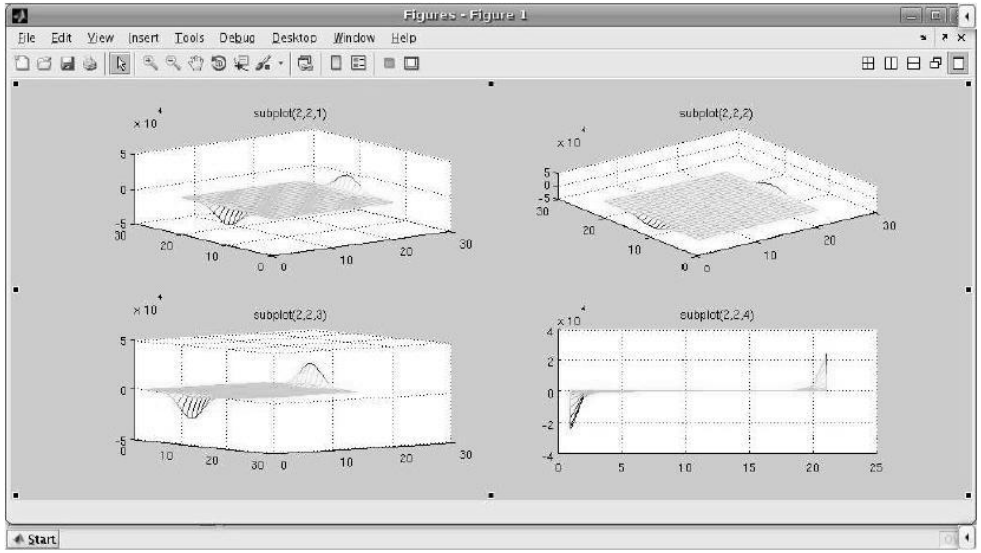
divides the figure window into *mxn* small sets and selects the *p* th set for the current plot.

The command `subplot(1,1,1)` returns to a single set of axes.

MULTIPLE PLOTS - subplot

```
[x, y] = meshgrid(-3:0.3:3);
z = x .* exp(x .^ 2 - y .^ 2);
subplot(2,2,1)
mesh(z),title('subplot(2,2,1)')
subplot(2,2,2)
mesh(z)
view(-37.5,70),title('subplot(2,2,2)')
subplot(2,2,3)
mesh(z)
view(37.5,-10),title('subplot(2,2,3)')
subplot(2,2,4)
mesh(z)
view(0,0),title('subplot(2,2,4)')
```


MULTIPLE PLOTS - subplot



figure

`figure(h)`, creates a new figure window or make the *h* th figure window current.

h is called figure handle

`clf` clears current figure.

`cla` deletes all plots and text from the current axes.

LOGARITHMIC PLOT

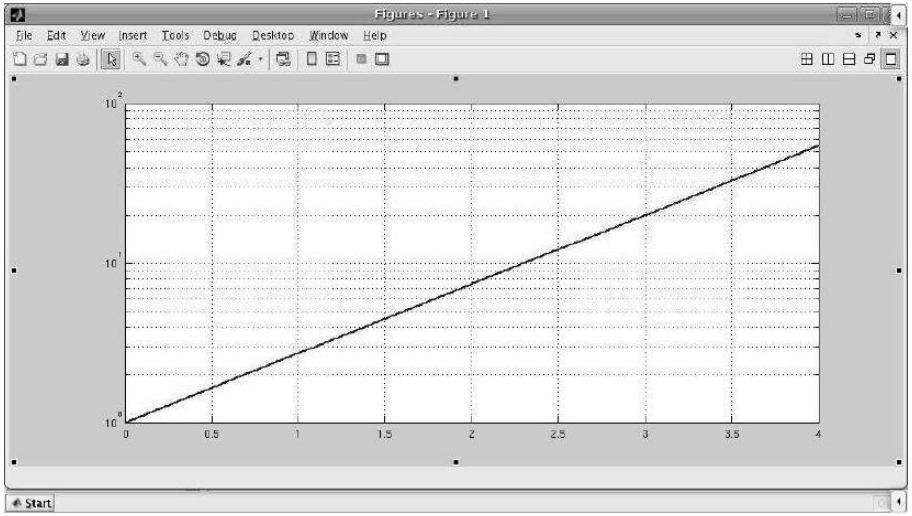
- `loglog` plots both axis in logarithmic scale
- `semilogx` plots only x axis in logarithmic scale
- `semilogy` plots only y axis in logarithmic scale

LOGARITHMIC PLOT

- `loglog` plots both axis in logarithmic scale
- `semilogx` plots only x axis in logarithmic scale
- `semilogy` plots only y axis in logarithmic scale

```
x = 0:0.01:4;
semilogy(x, exp(x)), grid
```


LOGARITHMIC PLOT



POLAR PLOT

The point (x, y) in cartesian coordinates represented by the point (θ, r) in *polar* coordinates

$x = r\cos(\theta)$

$y = r\sin(\theta)$

`polar(theta, r)`

POLAR PLOT

The point (x, y) in cartesian coordinates represented by the point (θ, r) in *polar* coordinates

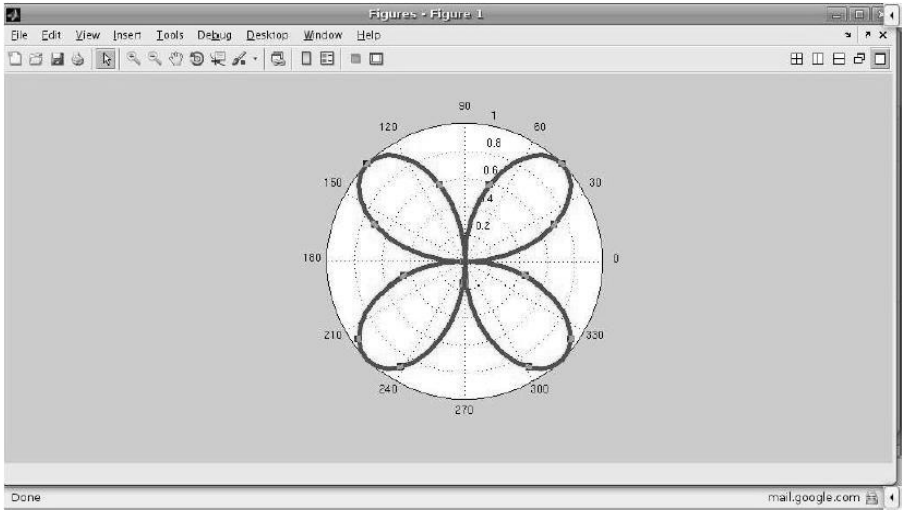
$x = r\cos(\theta)$

$y = r\sin(\theta)$

`polar(theta, r)`

```
x = 0 : pi/40 : 2 * pi;
polar(x, sin(2*x)), grid
```

POLAR PLOT



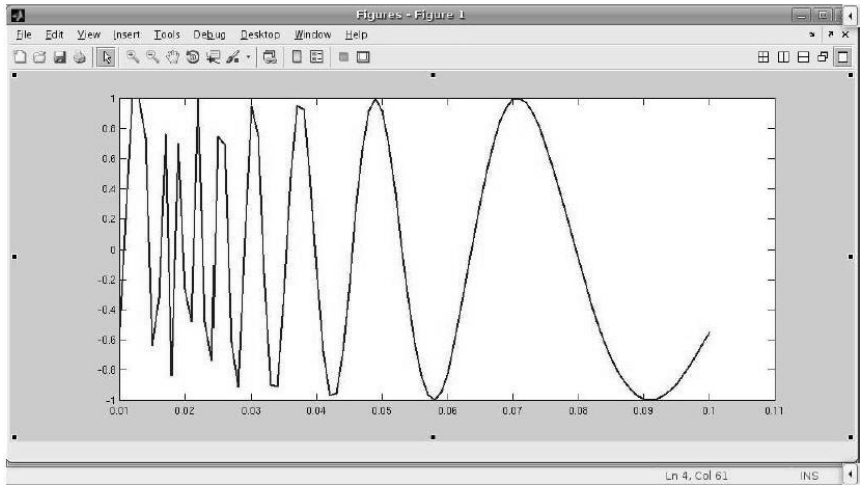
fplot

`fplot` plots the function given with the string.

- Use `fplot` if the function changes rapidly.
- Reduce the increment when using the `plot` command.

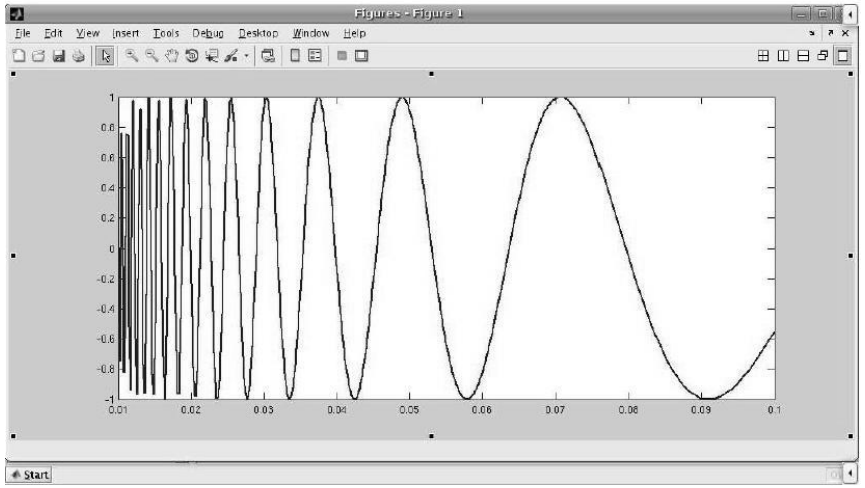
fplot

```
x = 0.01:0.001:0.1;
plot(x, sin(1./x))
```



fplot

```
fplot('sin(1/x)', [0.01 0.1])
```



SIMPLE 3D GRAPHICS

The `plot3` function

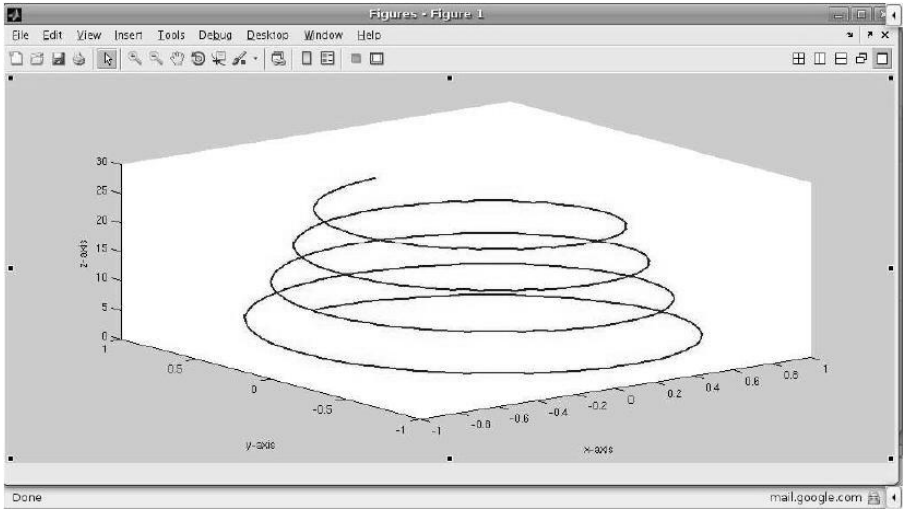
Usage;

```
plot3(x, y, z)
```

Example;

```
t = 0: pi/40 : 8*pi
plot3(exp(-0.02*t).*sin(t), exp(-0.02*t).*cos(t),t),
xlabel('x-axis'), ylabel('y-axis'), zlabel('z-axis')
```


SIMPLE 3D GRAPHICS



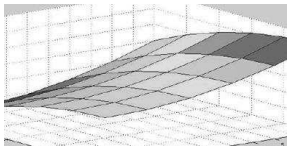
comet3

`comet3` is similar to `plot3`

except it draws an animated graphic.

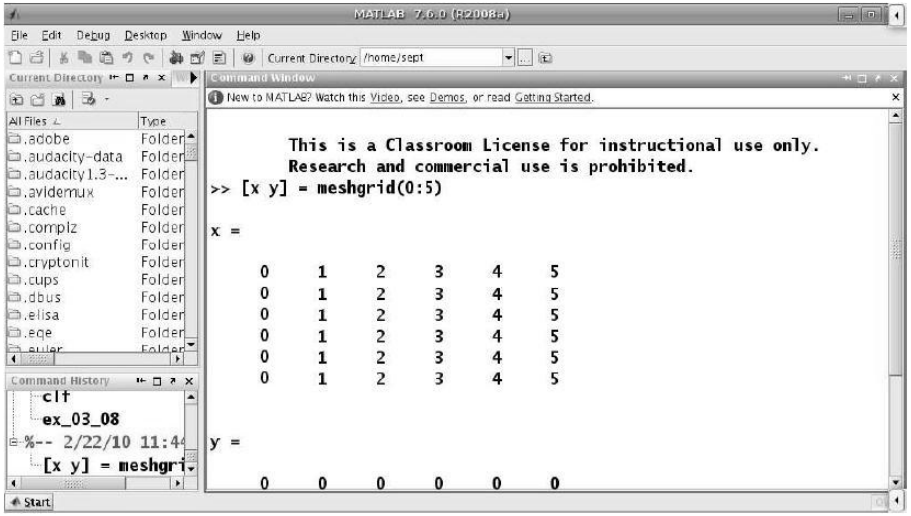
MESH SURFACES

- **MESH** : Open spaces in a net or screen; material (threads, wires, or cords, etc.) used to make an object with open spaces in it.

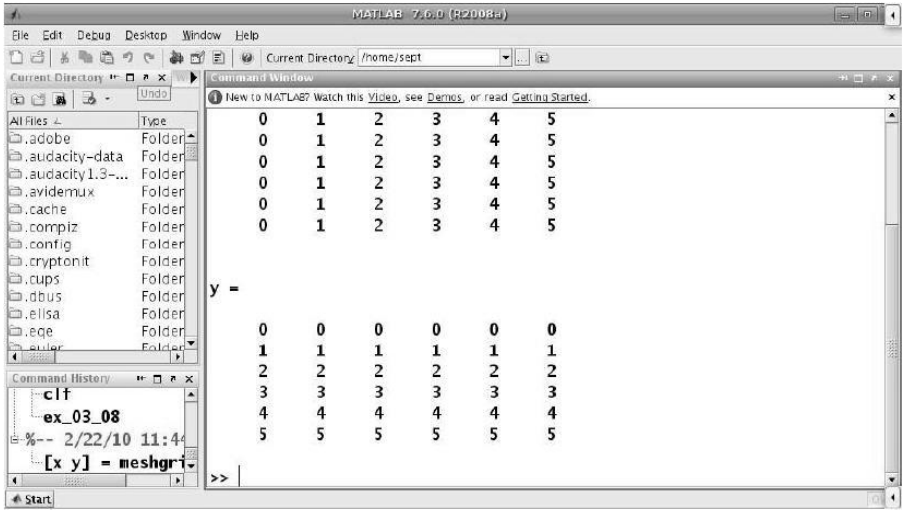


```
meshgrid
mesh(x, y, z)
surf(x, y, z)
contour
[x, y] = meshgrid(0:5)
```

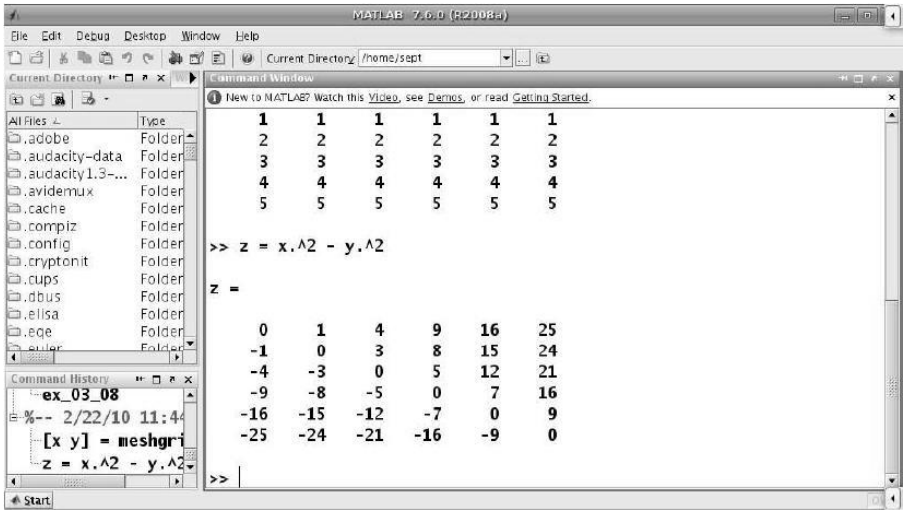
MESH SURFACES



MESH SURFACES



MESH SURFACES

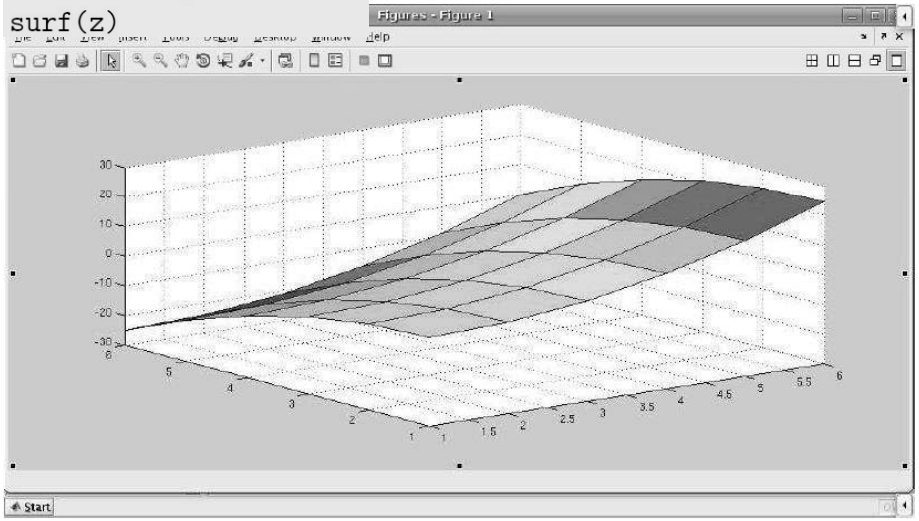


MESH SURFACES

```
[x y] = meshgrid(0:5);
z = x.^2 - y.^2;
surf(z)
```

MESH SURFACES

```
[x y] = meshgrid(0:5);
z = x.^2 - y.^2;
surf(z)
```

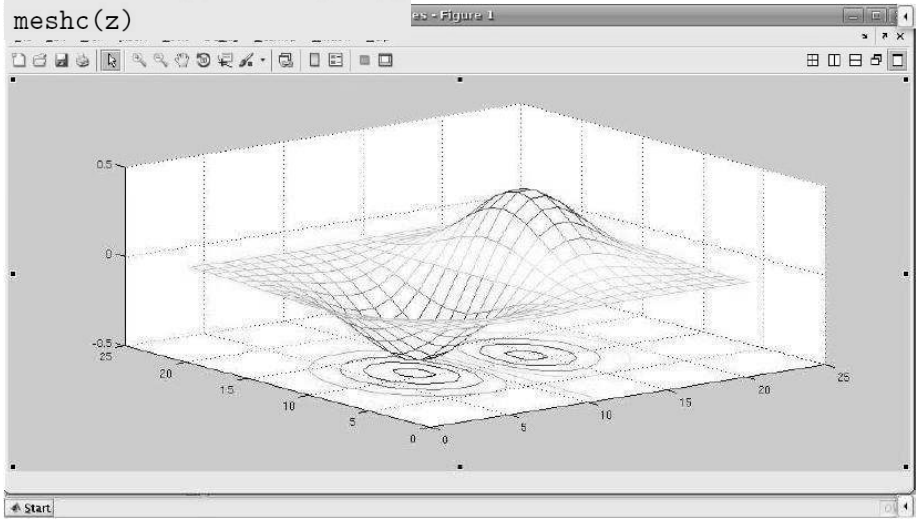


MESH SURFACES

```
[x y] = meshgrid(-2:.2:2);
z = x .* exp(-x.^2 -y.^2);
meshc(z)
```


MESH SURFACES

```
[x y] = meshgrid(-2:.2:2);
z = x .* exp(-x.^2 -y.^2);
meshc(z)
```



MATRIX VISUALIZATION

`mesh(A)`

`spy(A)`

A complete list of graphics functions

MATLAB help

```
>> help spy
 SPY Visualize sparsity pattern.
    SPY(S) plots the sparsity pattern of the matrix S.

    SPY(S,'LineSpec') uses the color and marker from the line
    specification string 'LineSpec' (See PLOT for possibilities).

    SPY(S,markersize) uses the specified marker size instead of
    a size which depends upon the figure size and the matrix order.

    SPY(S,'LineSpec',markersize) sets both.

    SPY(S,markersize,'LineSpec') also works.
```

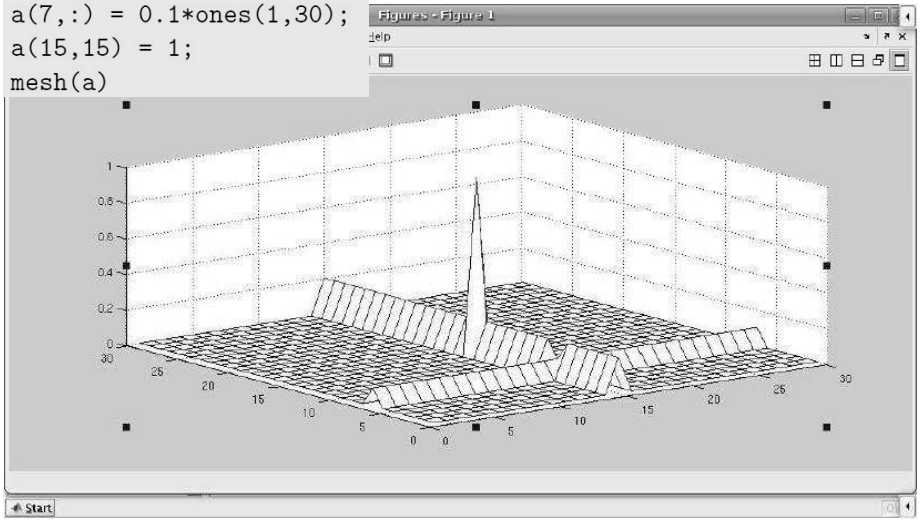
```
>> help mesh
 MESH   3-D mesh surface.
    MESH(X,Y,Z,C) plots the colored parametric mesh defined by
    four matrix arguments.  The view point is specified by VIEW.
    The axis labels are determined by the range of X, Y and Z,
    or by the current setting of AXIS.  The color scaling is determined
    by the range of C, or by the current setting of CAXIS.  The scaled
    color values are used as indices into the current COLORMAP.

    MESH(X,Y,Z) uses C = Z, so color is proportional to mesh height.

    MESH(x,y,Z,C) and MESH(x,y,Z) with two vector arguments replacing
    the first two matrix arguments, must have length(x) = n and
                                    s where [m,n] = size(Z).  In this case, the vertices
                                    lines are the triples (x(j), y(i), Z(i,j)).
                                    corresponds to the columns of Z and y corresponds to


                                    ESH(Z,C) use x = 1:n and y = 1:m.  In this case,
                                    ', is a single-valued function, defined over a
                                    rectangular grid.

                                    pertyName',PropertyValue,...) sets the value of
                                    surface property.  Multiple property values can be set
                                    statement.

    MESH(AX,...) plots into AX instead of GCA.

    MESH returns a handle to a surface plot object.

    AXIS, CAXIS, COLORMAP, HOLD, SHADING, HIDDEN and VIEW set figure,
    axes, and surface properties which affect the display of the mesh.
```

MATRIX VISUALIZATION

```
a = zeros(30,30);
a(:,15) = 0.2*ones(30,1);
a(7,:) = 0.1*ones(1,30);
a(15,15) = 1;
mesh(a)
```


MATRIX VISUALIZATION

```
a = zeros(30,30);
a(:,15) = 0.2*ones(30,1);
a(7,:) = 0.1*ones(1,30);
a(15,15) = 1;
mesh(a)
```



References for Week 3

1. Brian Hahn, Daniel T.Valentine, Essential Matlab for Engineers and Scientists, Elsevier, 2010.

2. Misza Kalechman, Practical Matlab Basics for Engineers, CRC Press, 2009.

Have a nice Week End