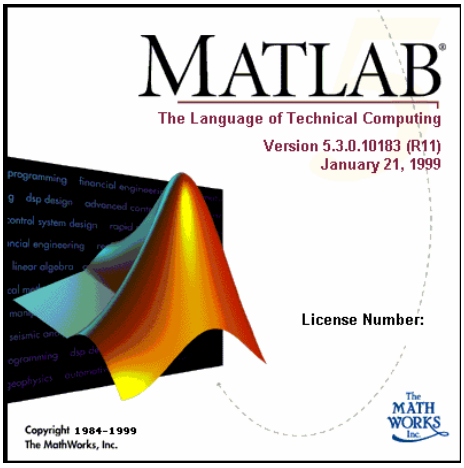
Using Matlab



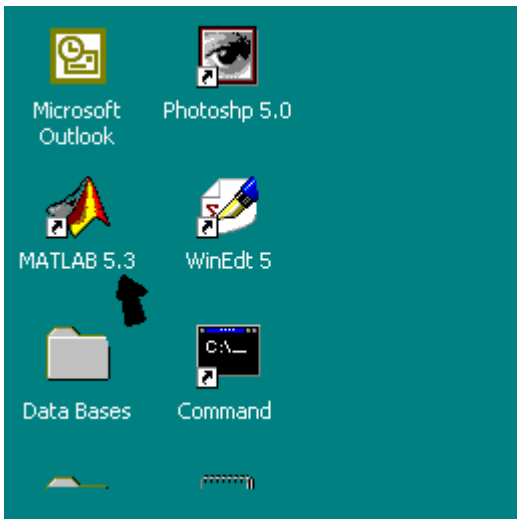
Introduction

Matlab is a special propose package and programming language which has been designed for scientific and engineering applications. Mathworks introduced the 1st version of Matlab in the middle of the 80s and it has since developed into a very popular and powerful software environment. Its name stands for **MAT**rix **LAB**oratory and naturally is very powerful in matrix manipulation. For performing Matlab operations, we have to be inside the Matlab environment and we can use both online commands and/or already-prepared programs (collection of commands that perform a certain algorithm). .

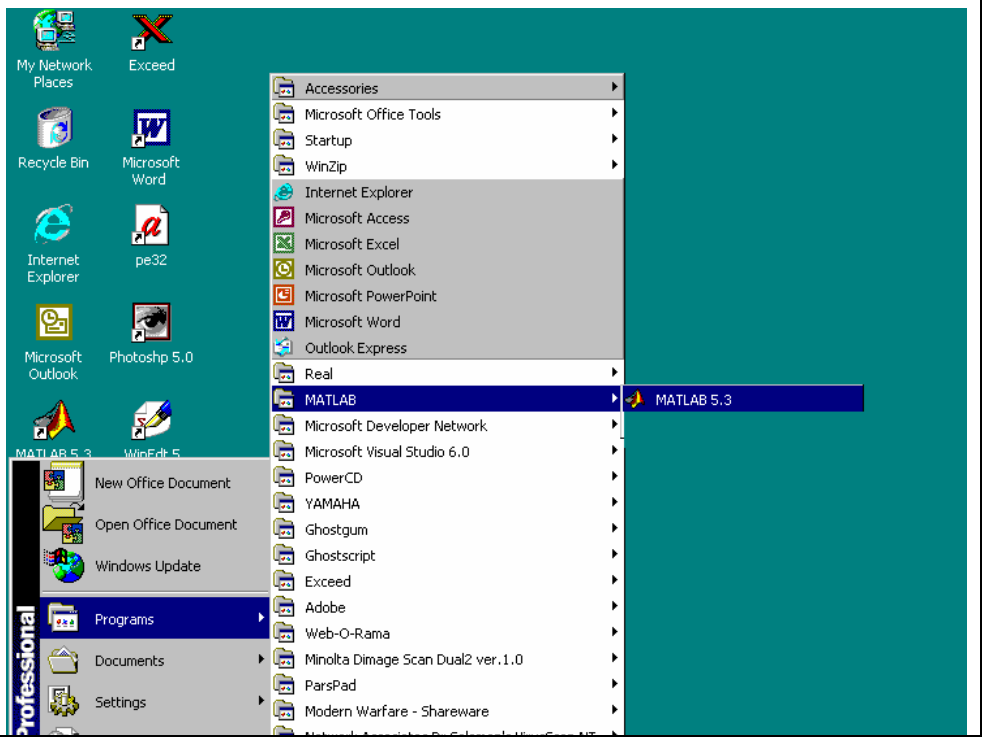
Getting Started

First you must run Matlab. If you can not see the Matlab icon on the desktop, go to the *Start* menu, then into the *Applications* menu, and then select *Matlab 6* or *Matlab 5* . This will take you into the Matlab environment:

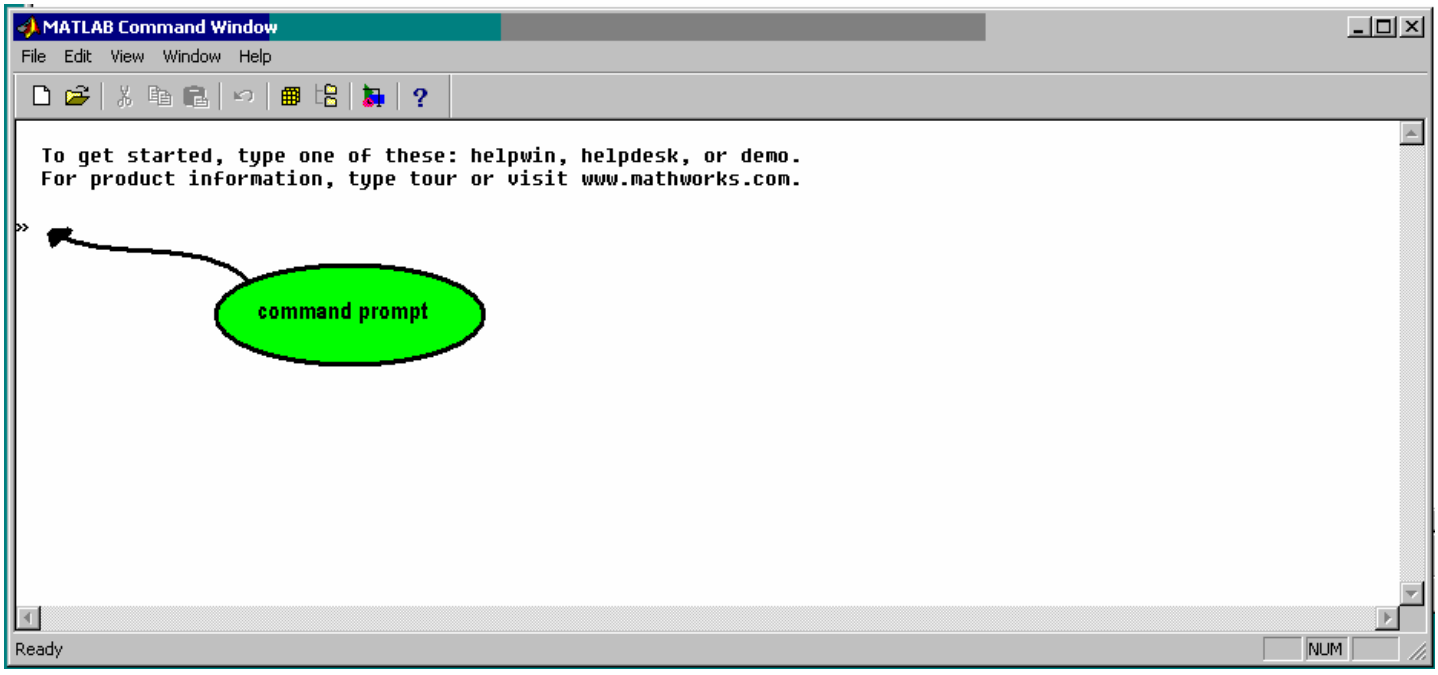
Either double click on the Matlab icon on your desktop,



Or select it from start/program menu

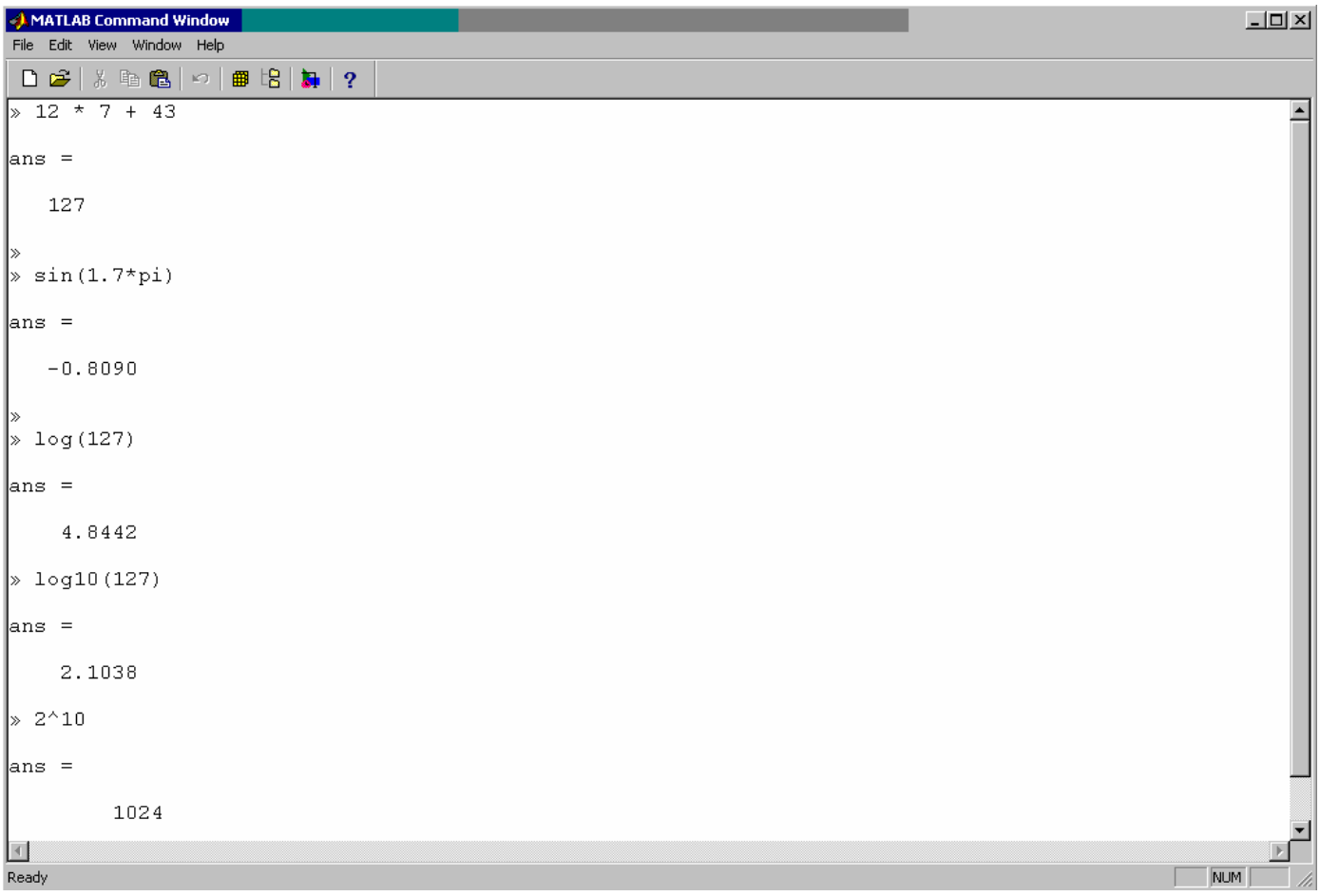


Then Matlab will run and you can see the Matlab *command* (main) *window* and its *command prompt*.



Now you can easily type your commands in front of the command prompt and get the answer after pressing the "Enter" or "Return" key. Here is an example.

Example: using Matlab as an advanced calculator

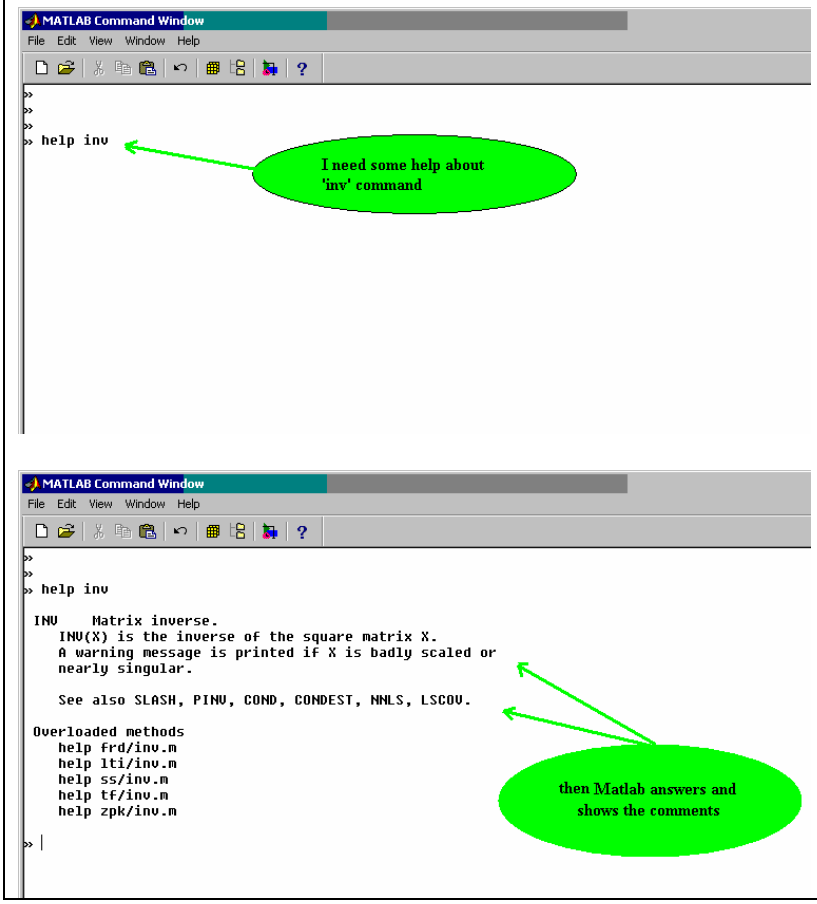


as that example shows, we can simply type a mathematical expression and get the answer immediately. This is the use of Matlab in on-line mode. As your first exercise, run Matlab on your machine, type the commands above, or make up your own mathematical expressions, and see the results.

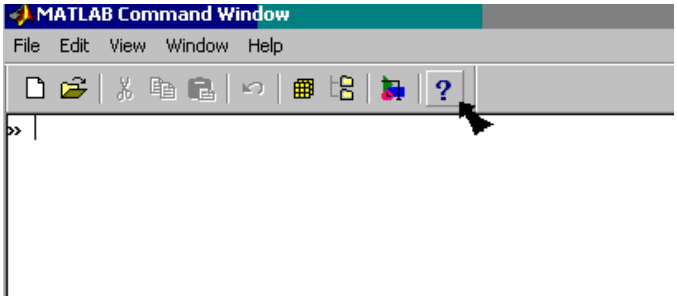
Help Facility

Matlab has got a very powerful and useful Help facility. There are two major ways of using Matlab help:

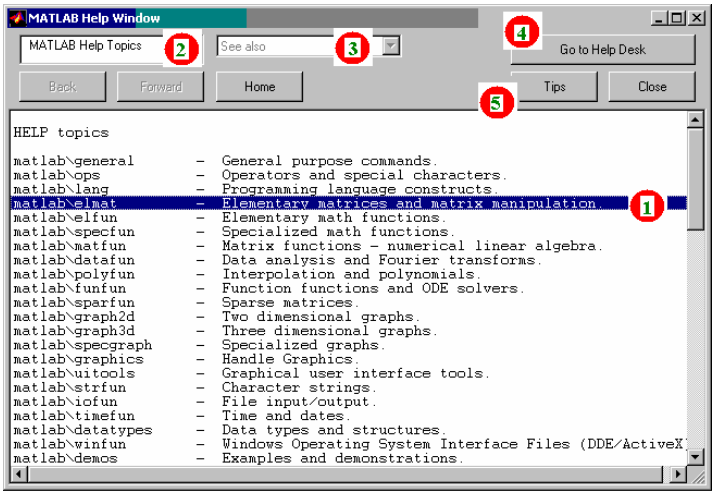
Type "help" and then your desired command to see the format for its use and any associated comments:



Click on the Help icon at the Matlab main window,



then ?Help? menu will be opened :



More details about the different options of the "Help Window":
(See the numbers in the right-hand image above)

1. Select a topic by double clicking on it and see the results.
2. Type your desired expression on "Matlab Help Topic" then hit the "Enter".
3. Open "See Also" menu for related topics.
4. Click on "Goto Help Desk" to browse the original Mathworks web-based help desk online.
5. Click on "Tips" for some brief help on "help"!

Some other useful Commands

Table below shows some useful general commands of Matlab and a brief description on each.

Command	Description
who	Lists all current workspace variables briefly.
whos	Lists all current workspace variables with details.
clc	Clears the command windows screen
clear	Deletes all the variables and make the workspace empty
dir	Lists the files in current directory
cd <new directory path>	Changes the current directory into the new desired one.
help	Online help facility, see below for more comments
quit	Closes all Matlab windows and quits from Matlab environment.

Amir Monadjemi, A.Monadjemi@bristol.ac.uk,
Majid Mirmehdi, M.Mirmehdi@bristol.ac.uk.

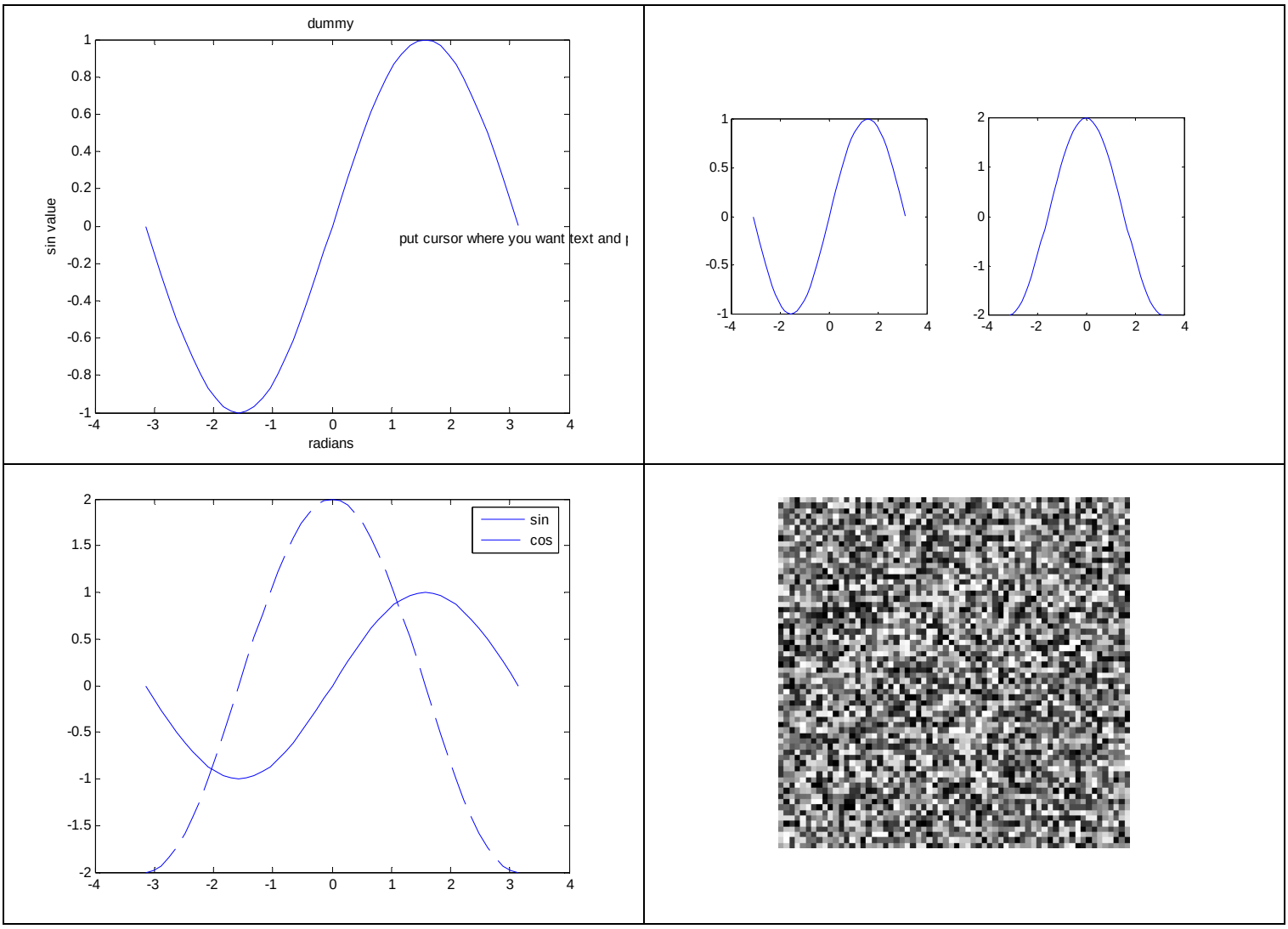
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (1) Help and basics

% The symbol "%" is used in front of a comment.

% To get help type "help" (will give list of help topics) or "help topic"

% If you don't know the exact name of the topic or command you are looking
for, type "lookfor keyword" (e.g., "lookfor regression")

% When writing a long matlab statement that exceeds a single row use ...
% to continue statement to next row.

% When using the command line, a ";" at the end means matlab will not
% display the result. If ";" is omitted then matlab will display result.

% Use the up-arrow to recall commands without retyping them (and down
% arrow to go forward in commands).

% Other commands borrowed from emacs and/or tcsh:
% C-a moves to beginning of line (C-e for end), C-f moves forward a
% character (C-b moves back), C-d deletes a character, C-k deletes
% the line to the right of the cursor, C-p goes back through the
% command history and C-n goes forward (equivalent to up and down arrows).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (2) Objects in matlab -- the basic objects in matlab are scalars,
% vectors, and matrices...


N     = 5                       % a scalar
v     = [1 0 0]                 % a row vector
v     = [1;2;3]                 % a column vector
v     = v'                      % transpose a vector
                                      (row to column or column to row)
v     = [1:.5:3]                % a vector in a specified range:
v     = pi*[-4:4]/4                   %     [start:stepsize:end]
v     = []                      % empty vector


m     = [1 2 3; 4 5 6]          % a matrix: 1ST parameter is ROWS
                                %         2ND parameter is COLS
m     = zeros(2,3)                    % a matrix of zeros
v     = ones(1,3)                     % a matrix of ones
m     = eye(3)                  % identity matrix
v     = rand(3,1)               % rand matrix (see also randn)

load matrix_data                % read data from a file:
                                % create a file 'matrix_data' containing:
                                %     2     3     4
                                %     5     6     7
                                %     1     2     3
```

```
v     = [1 2 3];                % access a vector element
v(3)                            %     vector(number)


m     = [1 2 3; 4 5 6]
m(1,3)                          % access a matrix element
                                %     matrix(rownumber, columnnumber)
m(2,:)                          % access a matrix row (2nd row)
m(:,1)                          % access a matrix column (1st row)


size(m)                         % size of a matrix
size(m,1)                       % number rows
size(m,2)                       % number of columns

m1    = zeros(size(m))          % create a new matrix with size of m

who                             % list of variables
whos                            % list/size/type of variables


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (3) Simple operations on vectors and matrices

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (A) Pointwise (element by element) Operations:

% addition of vectors/matrices and multiplication by a scalar
% are done "element by element"
a     = [1 2 3 4];              % vector
2 * a                           % scalar multiplication
a / 4                           % scalar multiplication
b     = [5 6 7 8];              % vector
a + b                           % pointwise vector addition
a - b                           % pointwise vector addition
a .^ 2                          % pointise vector squaring (note .)
a .* b                          % pointwise vector multiply (note .)
a ./ b                          % pointwise vector multiply (note .)

log( [1 2 3 4] )                % pointwise arithmetic operation
round( [1.5 2; 2.2 3.1] )       % pointwise arithmetic operation


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (B) Vector Operations (no for loops needed)
% Built-in matlab functions operate on vectors, if a matrix is given,
% then the function operates on each column of the matrix

a     = [1 4 6 3]               % vector
sum(a)                          % sum of vector elements
mean(a)                         % mean of vector elements
var(a)                          % variance
std(a)                          % standard deviation
```

```
max(a)                          % maximum


a     = [1 2 3; 4 5 6]          % matrix
mean(a)                         % mean of each column
max(a)                          % max of each column
max(max(a))                     % to obtain max of matrix
max(a(:))                       %     or...


%%%%%%%%%%%%%%%%%%%%%%%%
% (C) Matrix Operations:

[1 2 3] * [4 5 6]'              % row vector 1x3 times column vector 3x1
                                % results in single number, also
                                % known as dot product or inner product

[1 2 3]' * [4 5 6]              % column vector 3x1 times row vector 1x3
                                % results in 3x3 matrix, also
                                % known as outer product

a     = rand(3,2)               % 3x2 matrix
b     = rand(2,4)               % 2x4 matrix
c     = a * b                   % 3x4 matrix

a     = [1 2; 3 4; 5 6]         % 3 x 2 matrix
b     = [5 6 7];                % 3 x 1 vector
b * a                           % matrix multiply
a' * b'                         % matrix multiply


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%(4) Saving your work

save mysession                  % creates session.mat with all variables
save mysession a b              % save only variables a and b

clear all                       % clear all variables
clear a b                       % clear variables a and b

load mysession                  % load session
a
b


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%(5) Relations and control statements
% Example: given a vector v, create a new vector with values equal to
% v if they are greater than 0, and equal to 0 if they less than or
% equal to 0.


v     = [3 5 -2 5 -1 0]         % 1: FOR LOOPS
u     = zeros( size(v) );       % initialize
```

```
for i = 1:size(v,2)
      if( v(i) > 0 )
            u(i) = v(i);
      end
end
u

v     = [3 5 -2 5 -1 0]        % 2: NO FOR LOOPS
u2    = zeros( size(v) );      % initialize
ind   = find( v>0 )            % index into >0 elements
u2(ind)     = v( ind )


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%(6) Creating functions using m-files:
% Functions in matlab are written in m-files. Create a file called
% 'thres.m' In this file put the following:

function res = thres( v )

u     = zeros( size(v) );      % initialize
ind   = find( v>0 )            % index into >0 elements
u(ind)= v( ind )

return

v     = [3 5 -2 5 -1 0]
kk= thres( v )                 % call from command line or other routines


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%(7) Plotting

x     = [0 1 2 3 4];           % basic plotting
plot( x );
plot( x, 2*x );
axis( [0 8 0 8] );

x     = pi*[-24:24]/24;
plot( x, sin(x) );
xlabel( 'radians' );
ylabel( 'sin value' );
title( 'dummy' );
gtext( 'put cursor where you want text and press mouse' );

figure;                        % multiple functions in separate graphs
subplot( 1,2,1 );
plot( x, sin(x) );
axis square;
subplot( 1,2,2 );
plot( x, 2.*cos(x) );
axis square;

figure;                        % multiple functions in single graph
```

```
plot( x,sin(x) );
hold on;
plot (x, 2.*cos(x), '--' );
legend( 'sin', 'cos' );
hold off;

figure;                              % matrices as images
m     = rand(64,64);
imagesc(m)
colormap gray;
axis image
axis off;
```



% to plot the histogram of an image ....

fnam = input('enter the image file name : ' , 's') ;
[a,map] = imread(fnam) ;

% ---- the BAD ------------
[m,n] = size(a) ;
hist = zeros(1,256) ;

```
for i=1:m
   for j=1:n
      hist(a(i,j)+1) = hist(a(i,j)+1) + 1 ;
   end
end

% --- the GOOD ----------
hist2 = zeros(1,256) ;
for i=1:256
   hist2(i) = hist2(i) + sum(sum( (a==i) )) ;
end

% --- the UGLY -----
figure
subplot(2,2,1) , imshow(a) , colormap(map) ;
subplot(2,2,3) , plot(hist) , title('the 1st') ;
subplot(2,2,4) , plot(hist2) , title('the 2nd') ;
return
```