% (1) Help and basics

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

% The current directory is displayed at the top-right corner of MATLAB window

% To change the current directory, click the "..." botton or use "cd" if

% you like

% Use "dir" to see the list files in current directory

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

% When using the command line, a ";" at the end means matlab will notdisplay 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)

save matrix\_data m % save the variable m to a file named matrix\_data.mat

clear all % clear all variables currently used by MATLAB

load matrix\_data % read data from the saved file

m % display it - it is still there!

v = [1 2 3]; % access a vector element

length(v) % length of a vector

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 (sigma^{2})

std(a) % standard deviation (sigma)

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(:)) % another way to obtain max of matrix

%%%%%%%%%%%%%%%%%%%%%%%%

% (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

%%%%%%%%%%%%%%%%%%%%%%%%

% (D) 1D linear convolution:

a = [1,2,3,4]; % input signal I

b = [1,-1]; % input signal II

conv(a,b) % convolution of two signals

%%%%%%%%%%%%%%%%%%%%

%(4) Saving your work

save mysession % creates session.mat with all variables

save mysession a b % save only variables a and b

clear a b % clear variables a and b

clear all % clear all variables

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 )

% other control commands

% while, switch-case, etc

% Tip 1: try to avoid the use of loops because MATLAB has excellent support of vector(matrix)-based operations. You will be amazed by how many things can be done without loops.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%(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 u = thres( v )

u = zeros( size(v) ); % initialize

ind = find( v>0 ) % index into >0 elements

u(ind) = v( ind )

x = [3 5 -2 5 -1 0]

thres( x ) % call from command line

% you can write you own variance calculation function

function v=var\_mine(x)

N=length(x); % get the length

m=sum(x)/N; % obtain the mean, you can also use m=mean(x);

v=sum(x.^2)/N-m\*m; % calculate the variance cut and paste Lines 208-211 into a new file and save it as var\_mine.m

% now you can test your own function vs. the one offered by MATLAB

var(x)

var\_mine(x)

% function can have multiple inputs and outputs

% Example 1: a function that packs two matrices A,B into a larger matrice

% C=[A,B;B,A]

function C=matrice\_packing(A,B)

C=[A,B;B,A];

% Example 2: calculate both mean and variance of a vector

function [m,v]=mean\_and\_var(x)

m=mean(x);

v=var(x);

% !!!Cautious Notes!!!: if the name of your own MATLAB file is the same as some

% function name used by MATLAB itself, MATLAB will use the one with a higher priority defined in the PATH setting.

% Tip 2: When to write a new function? A thumb rule is when you need to

% call this function again and again (the same rule as in C programming).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%(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

% figure command creates a new figure window

subplot( 1,2,1 ); % creates 1x2 subgraphs, window 1 as current graph

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) % Scale data and display as image

colormap gray;

axis image

axis off;

n = zeros(64); % generate a square image

n(17:48,17:48)=1;

imshow(n,[]); % use >help imshow to find out what "[]" means

subplot(1,2,1), subimage(m) % display two images in parallel

subplot(1,2,2), subimage(n)

% Tip 3: showing your results visually often helps the debugging of your

% MATLAB programs.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%(8) Image I/O operations

% cameraman.tif is a test image provided by MATLAB

% if you want to read your own image into MATLAB, make sure that the image

% is under a directory recognized by MATLAB path setting

x=imread('cameraman.tif'); % read in the image

whos x % display information about x

figure(1);imshow(x,[]); % display the image

x(1:8,1:8) % elements in x are unsigned integers

y=fliplr(x); %flip image left to right

figure(2);imshow(y,[])

imwrite(y,'camera\_flip.bmp');

% if you want to apply arithmetic operations on uint8 type of data,

% you need to convert them into double first

z=(double(x)+double(y))/2; % linearly mixed two images

figure(3);imshow(z,[])

% since z is in double format, we need to translate it back to uint8

% or scale every pixel by 255 when writing it out

w=im2uint8(z/255);

imwrite(w,'camera\_mix.pgm'); % write the mixed image out as a PGM file

imwrite(z/255,'camera\_mix.png'); % write the mixed image out as a PNG file