**A Short Matlab/Octave Tutorial through Examples**

**General instructions:**

Matlab**:**

* Log onto your GW IT computer account, and bring up the Matlab application
* This brings a Matlab window that has a “>>” prompt

Octave:

* Octave is a public-domain equivalent to Matlab
* You can download Octave to your PC/laptop, and run the app; or you can access Octave in the cloud at <https://octave-online.net>

Guidelines:

* Enter of the commands below, one after another, to learn about the system by example
* Pay attention to the comments (after each %), as they provide valuable information

1. **Matrix constructions and operations**

* A=[2 3 5 3 2 9 -13 14 10] % assigns to A a row vector
* B=[5 7 3 10 -4 8 9 1 3]
* C=A+B % adds the two vectors A and B, and puts the result in C
* D=A-B; % subtracts the two vectors, putting the result in D
* E=2\*A % multiplies each component of vector A by 2
* F=A+5 % adds 5 to each component of A
* sum(A) % returns the sum of all the elements of A
* mean(A) % returns the mean, that is, average value of vector A
* length(A) % returns the length of vector A
* help sum % gives you a description of the command “sum”
* help mean % gives you a description of the command “mean”
* size(A) % returns the dimensions of matrix A
* [n,m]=size(A) % assign to n the # of rows in A, and to m the # of columns
* M=[2 3; 4 7; 8 5] % assigns to M a 3x2 matrix
* M’ % transpose of M
* P=[3 6 9 10 -5; 1 2 4 6 9] % P is a 2x5 matrix
* M\*P
* size(M)
* I=eye(3); % creates and stores in I the 3x3 identity matrix
* I % displays I; with a “;”, the variable is not displayed
* ones(3,2) % displays a 3x2 matrix of all 1’s
* zeros(3,6) % displays a 3x6 matrix of all 0’s
* M\*M’ % multiplies the two matrices M and M’
* A\*B %error. The matrix dimensions are not right
* A.\*B % works. It does pointwise multiplication
* A./B % it computes pointwise division
* A.^3 % raises every element of A to the 3rd power
* M=rand(7,7); % generates a 7x7 random matrix
* M^3 % computes M\*M\*M (note that M^3 ≠ M.^3)
* MI=inv(M); % computes the inverse of M, and stores it in MI
* det(M) % computes the determinant of matrix M
* floor(M) % computes the floor value of every entry in M
* ceil(M) % computes the ceiling value of every entry in M
* round(M) % computes the nearest integer of every entry in M
* Q=M(2:5,3:6) % Selects and assign to Q the window (submatrix) that falls

% between rows 2 and 5, and between columns 3 and 7

* M(2:4,3:5)=0 % assigns 0 to M(i,j) for all i=2,3,4 and all j=3,4,5
* G=floor(10\*rand(3,3)); % creates a 3x3 random matrix where 0<G(i,j)<10
* M(2:4,3:5)=G % assigns G to the submatrix M(2:4,3:5)
* J=1:5 % creates a row vector [1 2 3 4 5]
* J=2:3:15 % creates a row vector [2 5 8 11 15]
* J=1:0.1:2 % creates the vector [1 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2]
* J=10:-2:1 % creates the vector [10 8 6 4 2]

1. **Graphing**

* X=1:30; Y=X.^2; plot(X,Y)
* plot(X,Y, '--');
* X=0:0.1:2
* plot(X,X.^2, '-', X,X.^3-2\*X.^2+1, '--')
* X=0:0.01:4\*pi;
* plot(X, sin(X), '-', X, cos(X), '--')
* legend('sin','cos') % you can use the GUI of the figure to write legend – click on insert
* xlabel('time') % you can use the GUI of the figure to label the axes – click on insert
* ylabel(‘Y’)
* plot(X, sin(X),'-', X, X.^2/(16\*pi\*pi),'--')
* plot(X, sin(X), '-')
* hold
* plot(X, cos(X), '--')
* hold
* plot(X, sin(X),'-', X, X.^2/(16\*pi\*pi),'--')
* fplot(@(x) sin(x), [0 4\*pi]) % alternatively, do: x=0:0.01:4\*pi; plot(x, sin(x));
* fplot(@sin, [0 4\*pi]) % same as above
* fplot ("[cos(x), sin(x)]", [0, 2\*pi])
* ylim([-1.2 1.2])
* Graphical editing of figures: Use the icons and menus in top of the figure window
* help plot
* help fplot % fplot is used later in this tutorial
* check also: xlim, ylim, title, subplot
* Though not needed here, check: plot3, polar, errorbar, loglog, semilogx, semilogy.

1. **Image processing**

* [L,mapL]=imread('http://www2.seas.gwu.edu/~ayoussef/cs6351/lena512.gif');

% reads into L an image, and into mapL the color map of the image

* imagesc(L) % displays the image poorly
* colormap(mapL) % sets the color map to mapL, and results in

% a well-displayed image

* [I,mapi]=imread('http://www2.seas.gwu.edu/~ayoussef/cs6351/sky-and-birds448x640.gif'); % reads a color image
* imagesc(I) % displays the image poorly
* colormap(mapi) % sets the color-map to map
* GL=ind2gray(L,mapL); % converts L from an indexed image to true pixel values
* GI=ind2gray(I,mapi); % converts I from an indexed image to true pixel values
* imagesc(GI); colormap(gray)
* You can edit images much like you edit plots

1. **Transforms**

* A=[ 1 3 5 4.3 4 3 2.8 3.2 3.6 4 4.3]';
* F=fft(A) % the Fourier transform
* plot(abs(F)) % plots the magnitudes of the components of F
* D=dct(A) % the discrete cosine transform
* plot(D)
* FL=fft2(GL); % 2D FFT of GL (which was created in section C)
* DL=dct2(GL); % 2D DCT
* Lr=idct2(DL); % inverse dct2
* H = hadamard(n) % returns the Hadamard matrix of order n. H’\*H=n\*eye(n)
* HA=H\*A; % the Hadamard transform of column vector A

1. **Defining functions, applying and plotting them**

* f = @(x) 3\*x^2-2\*x+10 % defines the function
* f(3) % returns 31, derived by evaluating
* fplot(@(x) f(x),[-5,5]) % Intended to plot f(x) where x ranges from -5 to 5; but it

% gives an error message. Need to redefine f to work on arrays.

* f = @(x) 3\*x.^2-2\*x+10 % redefines f so it applies on arrays pointwise
* fplot(@(x) f(x),[-5,5]) % now it works! It plots f(x) where x ranges from -5 to 5.
* g=@(A) sum(A.^2) % function of an array:
* h=@(A, z) sum (A .\* (z .^ (0:length (A) - 1))) % a function of two variables A (array) and z

% .

% that is, h is a polynomial, specified by the array of coefficients A.

* h(1:10,2) % returns 9217
* u=@(x) abs (h (A, exp (i \* x))) % here A is assumed to have been defined.

%

* A=ones(1,5)/5; % A=[1 1 1 1 1]/5
* B=arrayfun(u,0:0.1:pi); % applies function at every element of array 0:0.1:pi
* plot(0:0.1:pi,B) % plots u for A=[1 1 1 1 1]/5
* A=[1 -1]; u=@(x) abs (h (A, exp (i \* x))); B=arrayfun(u,0:0.1:pi); plot(0:0.1:pi,B)

% plots u for A=[1 -1]

* A=[-.5 1 -0.5]; u=@(x) abs(h(A,exp(i\*x)));B=arrayfun(u,0:0.1:pi); plot(0:0.1:pi,B)

% plots u for A=[-.5 1 -0.5]

* A=[0.02674876 -0.01686412 -0.07822327 0.26686412 0.60294902 0.26686412

-0.07822327 -0.01686412 0.02674876]; u=@(x) abs(h(A,exp(i\*x))); B=arrayfun(u,0:0.1:pi); plot(0:0.1:pi,B) % plots u for this new A

* A=[0.04563588 -0.02877176 -0.29563588 0.55754353 -0.29563588 -0.02877176 0.04563588]; u=@(x) abs(h(A,exp(i\*x))); B=arrayfun(u,0:0.1:pi); plot(0:0.1:pi,B)

% plots u for this new A