**EXPERIMENT NO. 1**

INTRODUCTION TO MATLAB

MATLAB (matrix laboratory) is a [numerical computing](http://en.wikipedia.org/wiki/Numerical_analysis) environment and [fourth-generation programming language](http://en.wikipedia.org/wiki/Fourth-generation_programming_language). Developed by [MathWorks](http://en.wikipedia.org/wiki/MathWorks" \o "MathWorks), MATLAB allows [matrix](http://en.wikipedia.org/wiki/Matrix_(mathematics)) manipulations, plotting of [functions](http://en.wikipedia.org/wiki/Function_(mathematics)) and data, implementation of [algorithms](http://en.wikipedia.org/wiki/Algorithm), creation of [user interfaces](http://en.wikipedia.org/wiki/User_interface), and interfacing with programs written in other languages, including [C](http://en.wikipedia.org/wiki/C_(programming_language)), [C++](http://en.wikipedia.org/wiki/C%2B%2B), [Java](http://en.wikipedia.org/wiki/Java_(programming_language)), and [Fortran](http://en.wikipedia.org/wiki/Fortran).

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the [MuPAD](http://en.wikipedia.org/wiki/MuPAD" \o "MuPAD) [symbolic engine](http://en.wikipedia.org/wiki/Computer_algebra_system), allowing access to [symbolic computing](http://en.wikipedia.org/wiki/Symbolic_computing) capabilities. An additional package, [Simulink](http://en.wikipedia.org/wiki/Simulink" \o "Simulink), adds graphical multi-domain simulation and [Model-Based Design](http://en.wikipedia.org/wiki/Model_based_design) for [dynamic](http://en.wikipedia.org/wiki/Dynamical_system) and [embedded systems](http://en.wikipedia.org/wiki/Embedded_systems).

**Some of the MATLAB commands and functions :-**

1. XLABEL, YLABEL, ZLABEL

Used to Label the x-, y-, and z-axis

Each axes graphics object can have one label for the x-, y-, and z-axis. The label appears beneath its respective axis in a two-dimensional plot and to the side or beneath the axis in a three-dimensional plot.

* xlabel('string') labels the x-axis of the current axes.
* xlabel(fname) evaluates the function fname, which must return a string, then displays the string beside the x-axis.
* xlabel(axes\_handle,...), ylabel(axes\_handle,...), and zlabel(axes\_handle,...) plot into the axes with handle axes\_handle instead of the current axes (gca).
* h = xlabel(...), h = ylabel(...), and h = zlabel(...) return the handle to the text object used as the label.
* ylabel(...) and zlabel(...) label the y-axis and z-axis, respectively, of the current axes.

1. MAX

Maximum elements of an array

C = max(A) returns the largest elements along different dimensions of an array.

If A is a vector, max(A) returns the largest element in A.

If A is a matrix, max(A) treats the columns of A as vectors, returning a row vector containing the maximum element from each column.

If A is a multidimensional array, max(A) treats the values along the first non-singleton dimension as vectors, returning the maximum value of each vector.

* C = max(A,B) returns an array the same size as A and B with the largest elements taken from A or B.
* C = max(A,[],dim) returns the largest elements along the dimension of A specified by scalar dim. For example, max(A,[],1) produces the maximum values along the first dimension (the rows) of A.
* [C,I] = max(...) finds the indices of the maximum values of A, and returns them in output vector I. If there are several identical maximum values, the index of the first one found is returned.

1. MIN

Minimum elements of an array

C = min(A) returns the smallest elements along different dimensions of an array.

If A is a vector, min(A) returns the smallest element in A.

If A is a matrix, min(A) treats the columns of A as vectors, returning a row vector containing the minimum element from each column.

If A is a multidimensional array, min operates along the first nonsingleton dimension.

* C = min(A,B) returns an array the same size as A and B with the smallest elements taken from A or B.
* C = min(A,[],dim) returns the smallest elements along the dimension of A specified by scalar dim. For example, min(A,[],1) produces the minimum values along the first dimension (the rows) of A.
* [C,I] = min(...) finds the indices of the minimum values of A, and returns them in output vector I. If there are several identical minimum values, the index of the first one found is returned.

1. MOD

M = mod(X,Y) if Y ~= 0, returns X - n.\*Y where n = floor(X./Y) . If Y is not an integer and the quotient X./Y is within roundoff error of an integer, then n is that integer. By convention, mod(X,0) is X. The inputs X and Y must be real arrays of the same size, or real scalars.

The mod function is useful for congruence relationships: x and y are congruent (mod m) if and only if mod(x,m) == mod(y,m).

Examples

mod(13,5) =>ans =3

mod([1:5],3) =>ans = 1 2 0 1 2

1. ABS

abs(X) returns an array Y such that each element of Y is the absolute value of the corresponding element of X. If X is complex, abs(X) returns the complex modulus (magnitude), which is the same as sqrt(real(X).^2 + imag(X).^2)

Examples

abs(-5) =>ans =5

abs(3+4i) =>ans =5

1. MATRIX OPERATIONS

* Adding and Subtracting Matrices

Addition and subtraction of matrices is defined just as it is for arrays, element-by-element. Adding A to B and then subtracting A from the result recovers B. A = pascal(3);

B = magic(3);

X = A + B

X =9 2 7

4 7 10

5 12 8

Y = X - A

Y =8 1 6

3 5 7

4 9 2

Addition and subtraction require both matrices to have the same dimension, or one of them be a scalar. If the dimensions are incompatible, an error results. C = fix(10\*rand(3,2))

X = A + C

Error using ==> +

Matrix dimensions must agree.

* Creating and Concatenating Matrices

MATLAB is a matrix-based computing environment. All of the data that you enter into MATLAB is stored in the form of a matrix or a multidimensional array. Even a single numeric value like 100 is stored as a matrix (in this case, a matrix having dimensions 1-by-1): A = 100;

Name Size Bytes Class

A 1x1 8 double array

Regardless of the data type being used, whether it is numeric, character, or logical true or false data, MATLAB stores this data in matrix (or array) form. For example, the string 'Hello World' is a 1-by-11 matrix of individual character elements in MATLAB. You can also build matrices composed of more complex data types, such as MATLAB structures and cell arrays.

* Constructing a Simple Matrix

The simplest way to create a matrix in MATLAB is to use the matrix constructor operator, []. Create a row in the matrix by entering elements (shown as E below) within the brackets. Separate each element with a comma or space: row = [E1, E2, ..., Em] row = [E1 E2 ... Em]

For example, to create a one row matrix of five elements, type A = [12 62 93 -8 22];

To start a new row, terminate the current row with a semicolon: A = [row1; row2; ...; rown]

This example constructs a 3 row, 5 column (or 3-by-5) matrix of numbers. Note that all rows must have the same number of elements: A = [12 62 93 -8 22; 16 2 87 43 91; -4 17 -72 95 6]

A = 12 62 93 -8 22

16 2 87 43 91

-4 17 -72 95 6

The square brackets operator constructs two-dimensional matrices only, (including 0-by-0, 1-by-1, and 1-by-n matrices). To construct arrays of more than two dimensions, see Creating Multidimensional Arrays.

* Vector Products and Transpose

A row vector and a column vector of the same length can be multiplied in either order. The result is either a scalar, the inner product, or a matrix, the outer product. u = [3; 1; 4];

v = [2 0 -1];

X = u\*v

X = 6 0 -3

2 0 -1

8 0 -4

For real matrices, the transpose operation interchanges and . MATLAB uses the apostrophe (or single quote) to denote transpose. Our example matrix A is symmetric, so A' is equal to A. But B is not symmetric. B = magic(3);

X = B'

X = 8 3 4

1 5 9

6 7 2

Transposition turns a row vector into a column vector. x = v'

x = 2

0

-1

If x and y are both real column vectors, the product x\*y is not defined, but the two products x'\*y

and y'\*x are the same scalar. This quantity is used so frequently, it has three different names: inner product, scalar product, or dot product.

For a complex vector or matrix, z, the quantity z' denotes the complex conjugate transpose, where the sign of the complex part of each element is reversed. The unconjugated complex transpose, where the complex part of each element retains its sign, is denoted by z.'. So if z = [1+2i 3+4i]

then z' is 1-2i

3-4i

while z.' is 1+2i

3+4i

For complex vectors, the two scalar products x'\*y and y'\*x are complex conjugates of each other and the scalar product x'\*x of a complex vector with itself is real.

* Multiplying Matrices

Multiplication of matrices is defined in a way that reflects composition of the underlying linear transformations and allows compact representation of systems of simultaneous linear equations. The matrix product C = AB is defined when the column dimension of A is equal to the row dimension of B, or when one of them is a scalar. If A is m-by-p and B is p-by-n, their product C is m-by-n. The product can actually be defined using MATLAB for loops, colon notation, and vector dot products. A = pascal(3);

B = magic(3);

m = 3; n = 3;

for i = 1:m

for j = 1:n

C(i,j) = A(i,:)\*B(:,j);

end

end

1. GRID

The grid function turns the current axes' grid lines on and off.

* grid on adds major grid lines to the current axes.
* grid off removes major and minor grid lines from the current axes.

grid toggles the major grid visibility state.

grid(axes\_handle,...) uses the axes specified by axes\_handle instead of the current axes.

grid sets the XGrid, YGrid, and ZGrid properties of the axes.

grid minor sets the XGridMinor, YGridMinor, and ZGridMinor properties of the axes.

1. PLOT

Linear 2-D plot

* plot(Y) plots the columns of Y versus their index if Y is a real number. If Y is complex, plot(Y) is equivalent to plot(real(Y),imag(Y)). In all other uses of plot, the imaginary component is ignored.
* plot(X1,Y1,...) plots all lines defined by Xn versus Yn pairs. If only Xn or Yn is a matrix, the vector is plotted versus the rows or columns of the matrix, depending on whether the vector's row or column dimension matches the matrix.
* plot(X1,Y1,LineSpec,...) plots all lines defined by the Xn,Yn,LineSpec triples, where LineSpec is a line specification that determines line type, marker symbol, and color of the plotted lines. You can mix Xn,Yn,LineSpec triples with Xn,Yn pairs: plot(X1,Y1,X2,Y2,LineSpec,X3,Y3).
* plot(...,'PropertyName',PropertyValue,...) sets properties to the specified property values for all lineseries graphics objects created by plot. (See the "Examples" section for examples.)
* plot(axes\_handle,...) plots into the axes with handle axes\_handle instead of the current axes (gca).
* h = plot(...) returns a column vector of handles to lineseries graphics objects, one handle per line.

hlines = plot('v6',...) returns the handles to line objects instead of lineseries objects.

1. 2D STEM PLOTS

A stem plot displays data as lines (stems) terminated with a marker symbol at each data value. In a 2-D graph, stems extend from the x-axis.

The stem function displays two-dimensional discrete sequence data. For example, evaluating the function with the values alpha = .02; beta = .5; t = 0:4:200;

y = exp(-alpha\*t).\*cos(beta\*t);

yields a vector of discrete values for y at given values of t. A line plot shows the data points connected with a straight line. plot(t,y)

A stem plot of the same function plots only discrete points on the curve. stem(t,y)

1. SUBPLOT

Create and control multiple axes

subplot divides the current figure into rectangular panes that are numbered rowwise. Each pane contains an axes. Subsequent plots are output to the current pane.

* subplot(m,n,p) creates an axes in the pth pane of a figure divided into an m-by-n matrix of rectangular panes. The new axes becomes the current axes. If p is a vector, it specifies an axes having a position that covers all the subplot positions listed in p.
* subplot(m,n,p,'replace') If the specified axes already exists, delete it and create a new axes.
* subplot(m,n,p,'align') positions the individual axes so that the plot boxes align, but does not prevent the labels and ticks from overlapping.
* subplot(h) makes the axes with handle h current for subsequent plotting commands.
* h = subplot(...) returns the handle to the new axes.

1. WHO

Use who to list the current workspace variables. Use whos to list the variables and information about their size and class. For example: who

Your variables are: A M S V

whos

Name Size Bytes Class

A 1x4 32 double array

M 3x1 202 cell array

S 1x2 598 struct array

V 1x35 70 char array

Grand total is 76 elements using 902 bytes

Use exist to see if the specified variable is in the workspace

1. THE IDENTITY MATRIX

Generally accepted mathematical notation uses the capital letter to denote identity matrices, matrices of various sizes with ones on the main diagonal and zeros elsewhere. These matrices have the property that IA=A and AI=A whenever the dimensions are compatible. The original version of MATLAB could not use for this purpose because it did not distinguish between upper and lowercase letters and already served double duty as a subscript and as the complex unit. So an English language pun was introduced. The function

eye(m,n)

returns an m-by-n rectangular identity matrix and eye(n) returns an n-by-n square identity matrix.

1. ONES

Create an array of all ones

* Y = ones(n),Y = ones(m,n),Y = ones([m n]), Y = ones(d1,d2,d3...) ,Y = ones([d1 d2 d3...]),Y = ones(size(A)) etc.
* Y = ones(n) returns an n-by-n matrix of 1s. An error message appears if n is not a scalar.
* Y = ones(m,n) or Y = ones([m n]) returns an m-by-n matrix of ones.
* Y = ones(d1,d2,d3...) or Y = ones([d1 d2 d3...]) returns an array of 1s with dimensions d1-by-d2-by-d3-by-....
* Y = ones(size(A)) returns an array of 1s that is the same size as A.

ones(m, n,...,classname) or ones([m,n,...],classname) is an m-by-n-by-... array of ones of data type classname. classname is a string specifying the data type of the output. classname can have the following values: 'double', 'single', 'int8', 'uint8', 'int16', 'uint16', 'int32', or 'uint32'.

1. FZERO

Find zero of a function of one variable

x = fzero(fun,x0), x = fzero(fun,x0,options), [x,fval] = fzero(...), [x,fval,exitflag] = fzero(...)

1. PRINT AND PRINTOPT

Print and Printopt produce hardcopy output. All arguments to the print command are optional. You can use them in any combination or order.

print sends the contents of the current figure, including bitmap representations of any user interface controls, to the printer using the device and system printing command defined by printopt

1. IMREAD

Read image from graphics file

The imread function supports four general syntaxes, described below. The imread function also supports several other format-specific syntaxes. See Special Case Syntax for information about these syntaxes.

* A = imread(filename,fmt) reads a greyscale or color image from the file specified by the string filename, where the string fmt specifies the format of the file. If the file is not in the current directory or in a directory in the MATLAB path, specify the full pathname of the location on your system. For a list of all the possible values for fmt, see Supported Formats. If imread cannot find a file named filename, it looks for a file named filename.fmt.
* [X,map] = imread(filename,fmt) reads the indexed image in filename into X and its associated colormap into map. The colormap values are rescaled to the range [0,1].
* [...] = imread(URL,...) reads the image from an Internet URL. The URL must include the protocol type (e.g., http://).

1. ALL

Test to determine if all elements are nonzero

* B = all(A) tests whether all the elements along various dimensions of an array are nonzero or logical 1 (true). If A is a vector, all(A) returns logical 1 (true) if all the elements are nonzero and returns logical 0 (false) if one or more elements are zero.

If A is a matrix, all(A) treats the columns of A as vectors, returning a row vector of logical 1's and 0's. If A is a multidimensional array, all(A) treats the values along the first nonsingleton dimension as vectors, returning a logical condition for each vector.

* B = all(A,dim) tests along the dimension of A specified by scalar dim

1. HOLD

hold on, hold off, hold all etc.

The hold function determines whether new graphics objects are added to the graph or replace objects in the graph.

* hold on retains the current plot and certain axes properties so that subsequent graphing commands add to the existing graph.
* hold off resets axes properties to their defaults before drawing new plots. hold off is the default.

1. VECTOR PRODUCTS

A row vector and a column vector of the same length can be multiplied in either order. The result is either a scalar, the inner product, or a matrix, the outer product. u = [3; 1; 4];

v = [2 0 -1];

x = v\*u

x = 2

X = u\*v

X = 6 0 -3

2 0 -1

8 0 -4

1. RMPATHS

Remove directories from MATLAB search path

As an alternative to the rmpath function, use the Set Path dialog box. To open it, select Set Path from the File menu in the MATLAB desktop.

rmpath('directory') removes the specified directory from the current MATLAB search path. Use the full pathname for directory. rmpath directory is the unquoted form of the syntax.

1. MINIMIZING FUNCTION OF A VARIABLE

Given a mathematical function of a single variable coded in an M-file, you can use the fminbnd function to find a local minimizer of the function in a given interval. For example, to find a minimum of the humps function in the range (0.3, 1), use x = fminbnd(@humps,0.3,1)

which returns x = 0.6370

You can ask for a tabular display of output by passing a fourth argument created by the optimset command to fminbnd x = fminbnd(@humps,0.3,1,optimset('Display','iter'))

1. THE MAGIC FUNCTION

MATLAB actually has a built-in function that creates magic squares of almost any size. Not surprisingly, this function is named magic. B = magic(4)

B = 16 2 3 13

5 11 10 8

9 7 6 12

4 14 15 1

This matrix is almost the same as the one in the Dürer engraving and has all the same "magic" properties; the only difference is that the two middle columns are exchanged.

This says, for each of the rows of matrix B, reorder the elements in the order 1, 3, 2, 4. It produces A =

16 3 2 13

5 10 11 8

9 6 7 12

4 15 14 1

1. DIR

Display directory listing

As an alternative to the dir function, use the Current Directory browser.

Dir, dir name, files = dir('name')

* dir lists the files in the current working directory. Results are not sorted, but presented in the order returned by the operating system.
* dir name lists the specified files. The name argument can be a pathname, filename, or can include both. You can use absolute and relative pathnames and wildcards (\*).
* files = dir('directory') returns the list of files in the specified directory (or the current directory, if dirname is not specified) to an m-by-1 structure with the fields