# ***Lab 1***

## **Introduction to MATLAB**

**Learning Goals**

* Learn how to start MATLAB and set up your workspace
* Learn to enter arrays, perform basic operations and discover properties of matrices
* Keywords: default folder, array entry, operations, element-wise, transpose, augmentation

***Getting started with MATLAB***

**Start-up MATLAB and set up a default folder.** Step one is to start the MATLAB program. Normally you will double click on the MATLAB icon (see the graphic below) or select MATLAB from the start menu. Begin by setting up a default subdirectory folder where your work will be stored. This folder is where routines written to run in MATLAB, commonly referred to as *m-files*, will need to reside to allow a user to run any nonstandard MATLAB routines. To identify a default folder, select the small rectangular icon with 3 small dots, … , at the top right of the MATLAB toolbar following the listed *Current Directory*. Selection of the icon opens up a *Browse For Folder* window. Select the appropriate folder such as a folder set up for a user of a university system, a particular folder on a personal computer, or another memory device like a *jump drive*. Work completed and saved will be saved to the default folder chosen so choose wisely. In addition, you may wish to download or copy and paste m-files associated with these labs or other m-files provided by the instructor. Once a folder is selected, enter the command, *ls* , to see what is in the folder. Those familiar with the Unix operating system will recognize this as the list directory command that lists all of the files currently in the folder. File names are listed in alphabetical order, first by upper case then by lower case. MATLAB is case sensitive so that files named Myvector.m and myvector.m are *not* the same files to MATLAB, nor are the variable names A1 or a1. So, be careful with lower or upper case name choices for variable or file names.

***Arrays of numbers***

*Assignment statement:* assign the value of a number or array to a variable, that is, a named memory location. Format: myvariable = number. For example, to store 29 in the variable x, the MATLAB command and displayed results are

x=27

27

x=27;

To suppress the parroting of the variable and value(s), type a semicolon after the expression prior to pressing Enter; see above. To display the value(s) stored in a variable, type the variable name and press enter.

*Vectors:* arrays with one row or column. Format: v = [ list of numbers separated by spaces or comma ]. Examples of *row* vector entry:

vr=[1 23 3.4 5 -6]

1.0000 23.0000 3.4000 5.0000 -6.0000

vr=[1,23,3.4,5,-6]

1.0000 23.0000 3.4000 5.0000 -6.0000

Column vectors are entered in one of two ways, a.), enter the vector as a row vector with a single quote after the closing bracket, ]’ or b.) enter the numbers with semicolons, ; , in between them. Examples of *column* vector entry:

vc=[1 23 3.4 5 -6.0]'

1.0000

23.0000

3.4000

5.0000

-6.0000

vc=[1; 23; 3.4; 5; -6.0]

1.0000

23.0000

3.4000

5.0000

-6.0000

*Two-dimensional arrays:* Format: A = [row1 ; row2 ; ... ; rowN] an array with N rows. Entry of arrays begins with a left square bracket, [ , and ends with a right square bracket, ] . Within the brackets rows of numbers are separated by semicolons, ; , each row must have the same number of elements and elements of the rows are separated by spaces or commas. In the following  is an array with 3 rows and 3 columns. The array 2, with 4 rows and 3 columns is entered using an alternate method. Try both.

A=[2 3 4;-5 4 2;4 -3 5]

2 3 4

-5 4 2

4 -3 5

A2=[1 3 4

-5 54 2

4 -3 15

2 2 -9]

1 3 4

-5 54 2

4 -3 15

2 2 -9

*The display command:* disp(variable). To display , enter *disp(A)*, to display  and 2 together enter *disp(A); disp(A2)*

disp(A); disp(A2)

2 3 4

-5 4 2

4 -3 5

1 3 4

-5 54 2

4 -3 15

2 2 -9

*Revision of array elements:* Assignment: change a single element. Use row and column position numbers as an argument of the array variable name. Note: the use of the colon (wild card) in arrays, : , means to use all elements of a specified row or column. The notation, A(:,:) represents all elements of array . To change the row 2 column 3 element of  from 2 to 17, type A(2,3)=17

A(2,3)=17

2 3 4

-5 4 17

4 -3 5

To replace row 3 of  by [1 23 8], type A(3,:)=[1 23 8]

A(3,:)=[1 23 8]

2 3 4

-5 4 17

1 23 8

To replace column 1 of  by [ 4 6 -14]’, type A(:,2)=[4 6 -14]’

A(:,2)=[4 6 -14]'

2 4 4

-5 6 17

1 -14 8

*Up Arrow:* Use of the *up arrow*, ↑ , key of the keyboard, scrolls through previously entered commands. MATLAB remembers the last 30 or so commands you applied. This is a useful shortcut to reapply or revise command line entries. *Example:* repeatedly apply the up arrow key until you get to the line on which you originally entered , then use the ← and → keys to move to the character(s) you wish to change, press the *Delete* key (Backspace if you are to the right of the character), enter the new character and press enter. Tryit: change the row 1, column 2 element of  to 48.

***Array operations***

*Transpose:* The transpose of an array  is done as ´ . The transpose operation in MATLAB is especially useful for vector entry, a column vector may be entered as a row vector followed by a single quote as illustrated below.

v = [1 23 3.4 5 -6]'

1.0000

23.0000

3.4000

5.0000

-6.0000

***Operations in MATLAB***

|  |  |  |
| --- | --- | --- |
| Symbol | Operation | MATLAB array operation |
| **+/-** | addition/subtraction | elementwise addition or subtraction |
| **\*** | multiplication | standard matrix multiplication |
| **.\*** | multiplication | elementwise multiplication,  e.g., [4 3].\*[2 -2] = [8 -6] |
| **/** | Division | B/A = (A’\B’)’, solution of X\*A=B |
| **. /** | Division | elementwise division,  e.g., [4 3] . / [2 -2] = [2 -1.5] |
| **\** | None | A\B, efficient solution of A\*X=B |
| **^** | exponentiation | matrix exponentiation,A^3=A\*A\*A |
| **. ^** | exponentiation | elementwise exponentiation,  e.g., [4 3] . ^ 3 = [64 27] |

***Augmenting arrays***

|  |  |
| --- | --- |
| Symbol | Action ( meaning ) |
| **;** | Begin new row of array, separate MATLAB commands on a line |
| **:** | Wildcard for all rows or all columns, separator for limits |
| **,** | Separator for elements or array parts |

*Rowwise augmentation*: [B ; C], the new augmented array has *B* on top and *C* on bottom, if *B* and *C* have the same number of columns. If *B* = [1 2], *C* = [3 4],

B=[1 2];C=[3 4];[B;C]

1 2

3 4

*Columnwise augmentation*: [B C], the new array has *B* and *C* side by side, if *B* and C have the same number of rows. If *B* = [1 2], *C* = [3 4],

[B C]

1 2 3 4

***Picking off parts of arrays: how to select sub-arrays***

For the array *D*, the command D(*row i* : *row j , column k* : *column l*), picks off subarrays of array *D*. For the following examples, enter *D = hilb(6); format rat*. The command *E=D(2:4,5:6)* stores elements from rows 2 through 4 and in columns 5 through 6 of *D* into *E*. Note: anything entered after the % symbol is a comment and is ignored.

D=hilb(6);format rat; E=D(2:4,5:6)

1/6 1/7

1/7 1/8

1/8 1/9

F=D(:,4:6) %stores elements from all rows and in columns 4 through 6 of *D* into *F*.

1/4 1/5 1/6

1/5 1/6 1/7

1/6 1/7 1/8

1/7 1/8 1/9

1/8 1/9 1/10

1/9 1/10 1/11

G=D(2:4,:) %stores the elements in rows 2 through 4, all columns, of *D* into *G*.

1/2 1/3 1/4 1/5 1/6 1/7

1/3 1/4 1/5 1/6 1/7 1/8

1/4 1/5 1/6 1/7 1/8 1/9

***Reshaping arrays***

Use of the MATLAB command *reshape*. Type *help reshape* for more detail and options of the command. E=reshape(D,9,4) (reshapes *D* into an array of 9 rows and 4 columns, *try it*. Notice where the row or column values of the original *D* end up. Repeat the command to create 2×18 and 18×2 sub matrices of *D*.

***Save and load commands***

These commands save workspace variables to the disk or load saved workspace variables from the disk. These options allow the user to save work completed thus not having to reenter all of the variables or arrays to continue the project. See also the diary command. The following definitions are directly from the MATLAB help files.

SAVE FILENAME saves all workspace variables to the binary "MAT-file" named FILENAME.mat. The data may be retrieved with LOAD. If FILENAME has no extension, .mat is assumed.

LOAD FILENAME retrieves all variables from a file in the default subdirectory. If FILENAME has no extension LOAD looks for FILENAME.mat and, if found, LOAD treats the file as a binary "MAT-file". If FILENAME.mat is not found, or if FILENAME has an extension other than .mat it is treated as an ASCII file.

DIARY FILENAME causes a copy of all subsequent command window input and most of the resulting command window output to be appended to the named file.

*Exercises*

1. Given the following set of arrays, enter the arrays into MATLAB, complete each of the operations indicated and respond to the indicated query. The matrices  and  may be entered as A = [1 5 10; 12 -3 20; 4 6 30] and the vector B = [3 ; 2 ; -4] directly as a column vector, or enter *B* as a row vector and transpose it to create a column, B = [ 3 2 -4] ' . Note: MATLAB does not place brackets around the displayed array; however, you should include brackets around response arrays in your answer sheet.



* 1. *Scalar addition of a real or complex number*. Complete the following and write out the result displayed in MATLAB for i.), ii.) and iii.):

i.)  ii.)  iii.) 

iv.)  v.) 

Based on your calculations, make a conjecture on the new matrix that results

from adding, or subtracting, a constant number to a matrix.

* 1. *Scalar multiplication by a real or complex number*. Complete the

following and write out the result displayed by MATLAB (remember your

brackets).

i.)  ii.)  iii.) 

Based on your calculations, make a conjecture on the new matrix that

results from multiplication of a matrix by a constant.

Enter the additional arrays in MATLAB. The arrays will be used in problems 2-5.



1. *Matrix multiplication, standard or elementwise*. Complete the following computations and write out the result for the multiplied array in i.) - iv.). Indicate which operations are standard matrix multiplication and which are elementwise matrix multiplication; see the Operations in MATLAB table. The command sequence of the type *a1,a2,a3* implemented below causes MATLAB to display all three arrays.

i.)  (standard, elementwise) ii.)  (standard, elementwise)

iii.)  (standard, elementwise) iv.)  (standard, elementwise)

v.)  (standard, elementwise) vi.)  (standard, elementwise)

vii.)  (standard, elementwise) viii.)  (standard, elementwise)

Based on the results of your calculations, discuss your observations of any differences in the matrix resulting from standard multiplication or elementwise multiplication.

1. *Multiplication of an array by a diagonal matrix*. A diagonal matrix is a square matrix with zeros in all locations not on the *main* diagonal (top left corner to bottom right corner of the matrix) and where any nonzero numbers must be on the *main* diagonal. Enter the following matrix in MATLAB. The matrix may be entered as: D = [10 0 0; 0 20 0 ; 0 0 30] or D = diag([10 20 30]), try both versions to verify that the Matlab shortcut works. Write out the result of the multiplied matrices displayed by MATLAB in each case.



*Left multiplication*. Complete the following.

i.)  ii.) iii.)

Based on your calculations, state a conjecture about the new matrix

resulting from multiplication, on the *left*, by a diagonal matrix.

*Right multiplication*. Complete the following.

i.)  ii.)  iii.) 

Based on your calculations, state a conjecture on the result of

multiplication, on the *right*, by a diagonal matrix.

1. *Transposed or augmented matrices*. Investigate the result of applying the *transpose* operation and discover how to *augment* matrices to make new bigger ones.

The *transpose* of a matrix. Apply the transpose operation ´ (single quote) in MATLAB. Complete the following.

i.)  ii.)  iii.) 

Based on your calculations, state a conjecture on the result of application of the transpose operation.

*Augmented matrices*. The augmentation of one or more matrix with another is straight forward in MATLAB. For example, to augment the coefficient matrix  with the right hand side vector  one enters: [A B]. The left bracket, [, indicates the beginning of the entry of an array and the right bracket, ], indicates that the array entry is completed. Elements entered with a space (or comma) between them indicate they go on the same row or group of rows associated with the first array. The semi-colon ( ; ) in an array entry sequence indicates the end of a row(s) and causes the next elements entered to be placed on the next row(s) of the array. Complete the following computations and write out the displayed result.

i.)  ii.) 

Complete the following computations. Prior to entering the command, conjecture what the shape of the resultant matrix will be. Indicate (circle one) if the new array will be augmented rowwise (vertically) or columnwise (horizontally).

iii.)  iv.) 

Conjecture: rowwise, columnwise Conjecture: rowwise, columnwise

Based on your observations from the calculations completed in i.)-iv.), write valid MATLAB commands to create the following augmented matrices from matrices  of the previous exercises.







1. The *big* command, a heavily used MATLAB command and/or tool of linear algebra is *rref*, the command to convert a matrix into its row reduced echelon form. To apply the *rref* command to array , for example, *rref(W)* is entered into MATLAB. Complete the following computations to apply *rref* to various matrices or augmented matrices and write out the displayed result (arrays  are as entered earlier).

i.) rref() ii.) rref()

iii.) rref() iv.) format rat; rref()

v.) rref() vi.) rref( )

Based on your observations of the *rref* results make a conjecture of what row reduced echelon forms of a matrix entail. What are the consistent characteristics of the row reduced echelon form of a matrix?