

## ***Experiment I. Introduction to MATLAB I***

### **Introduction**

MATLAB, which stands for **Matrix Laboratory**, is a very powerful program for performing numerical and symbolic calculations, and is widely used in science and engineering, as well as in mathematics.

### **1.1 Objectives and Expectations:**

- This lab is designed to give you a quick way to become familiar with the MATLAB software by introducing you the basic features, commands, and functions.
- In this lab, you will discover that entering and solving complex numbers in MATLAB is as easy as entering and solving real numbers, especially with the help of MATLAB built-in complex functions.
- The lab is intended to be very interactive. You should have the required software running while you are reading the pages, and you should perform along with the examples.
- Upon completion, you should know how to start MATLAB, how to get HELP, how to assign variables in MATLAB and to perform the typical complex numbers operations (i.e., complex conjugate, addition, subtraction, multiplication, division, expression simplification) and the conversions of complex numbers in both rectangular and polar forms with and without using MATLAB built-in functions.

### **1.2 What is MATLAB?**

A high-performance language for technical computing.

Typical uses of MATLAB:

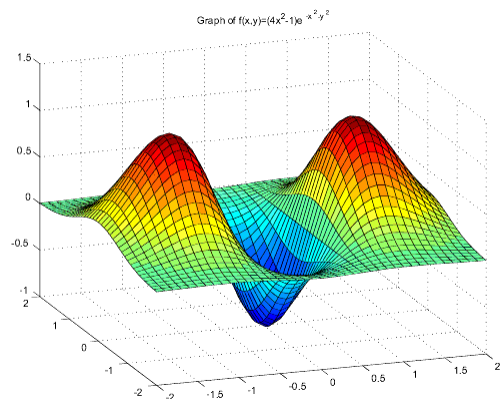
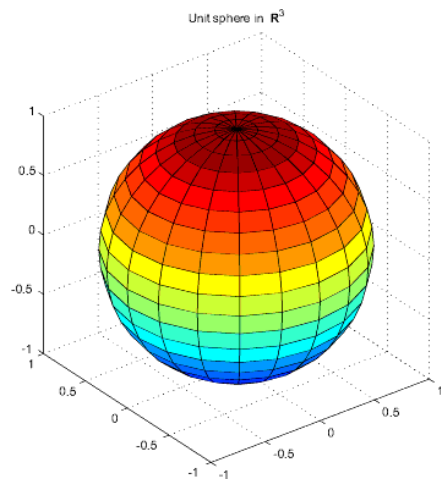
- Mathematical computations.
- Algorithmic development.
- Model prototyping (prior to complex model development).
- Data analysis and exploration of data (visualization).
- Scientific and engineering graphics for presentation.
- Complex analysis using MATLAB toolboxes (i.e., statistics,
- neural networks, fuzzy logic, H-infinity control, economics, etc.) .

### 1.3 Why is MATLAB Good for Me?

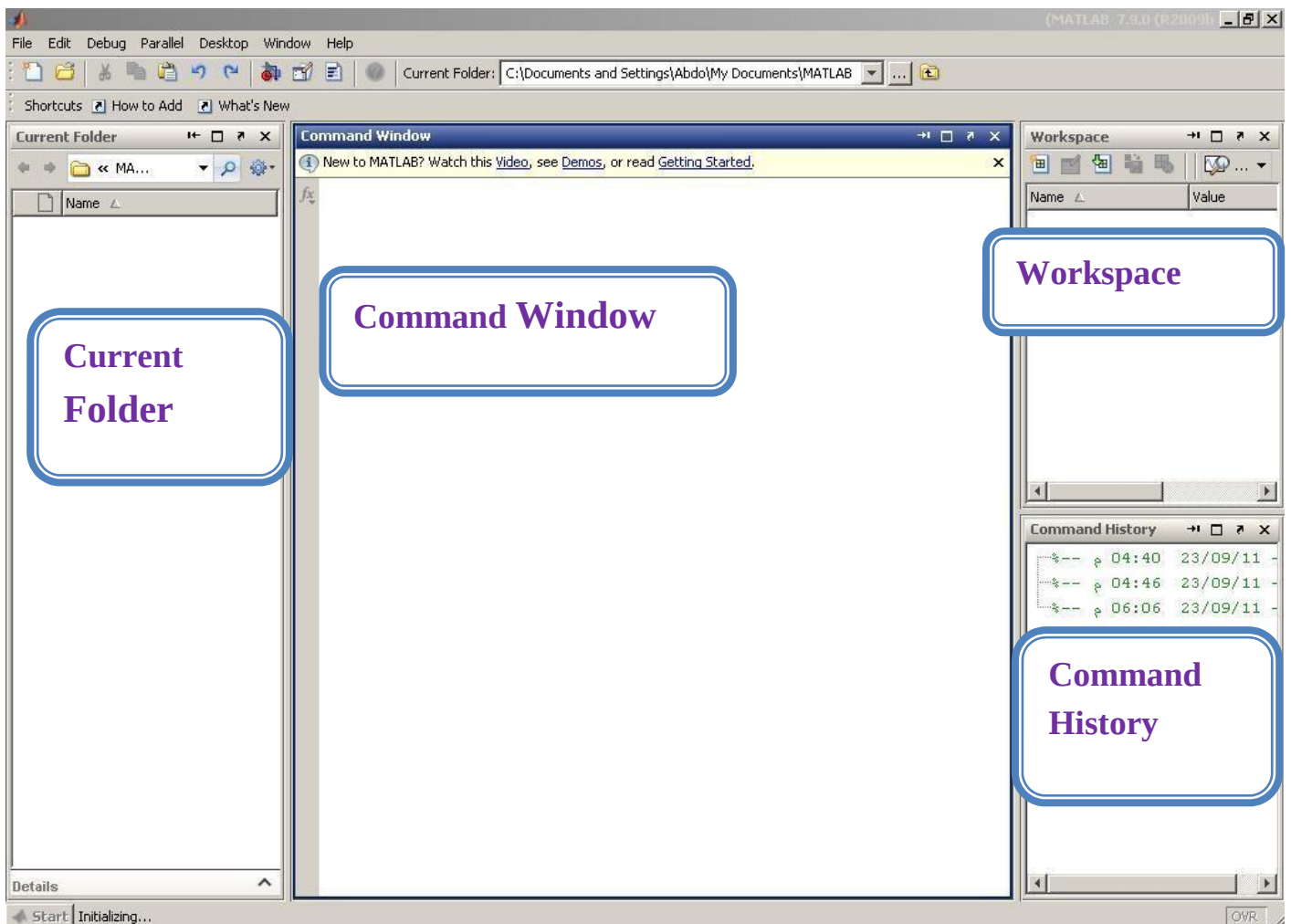
- Because it simplifies the analysis of mathematical models .
- It frees you from coding in high-level languages (saves a lot of time - with some computational speed penalties)
- Provides an extensible programming/visualization environment.
- Provides professional looking graphs.
- Provide a lot of toolbox that help me.
- MATLAB is usually faster than Mathematica and Maple in numeric intensive tasks.
- MATLAB has more textbooks than other packages combined (350+ books). Perhaps this speaks on the acceptance by the user community.

### 1.4 Basics of the Technical Language

- MATLAB is a technical language to ease scientific computations.
- The name is derived from **MAT**rix **LAB**oratory .
- It provides many of the attributes of spreadsheets and programming languages.
- MATLAB is a case sensitive language (a variable named “c” is different than another one called “C”).
- In interactive mode MATLAB scripts are platform independent (good for cross platform portability).
- MATLAB works with matrices.
- Everything MATLAB understands is a matrix (from text to large cell arrays and structure arrays).



## 1.5 The MATLAB Environment



## 1.6 Basic Components of the MATLAB Environment

MATLAB has the following basic window components:

- Command Window
  - to execute commands in the MATLAB environment
- Current Directory Window
  - to quickly access files on the MATLAB path
- Figure Window
  - to display graphical output from MATLAB code
- Workspace Window
  - to view variable definitions and variable memory allocations
- Command History Window
  - displays all commands issued in MATLAB since the last session (good for learning and verification)

## Getting Started

- When MATLAB starts the MATLAB prompt `>>` appears.
- All MATLAB commands are executed from this prompt.

```
>> 2.3+4.2
ans =
    6.5000
```

- MATLAB assigns the result to variable name 'ans'. A percent sign is a comment and is ignored.

```
>> 1+2
ans =
    3
```

- By default MATLAB returns numerical expressions as decimals with 5 digits. The format function is used to change the format of the output. Type format rat to have MATLAB return rational expressions.

```
>> format rat
>> 5.1-3.3
ans =
    9/5
```

- To eliminate the extra spacing type format compact.

```
>> format compact
>> 5*7
ans =
    35
```

- Operators and Special Characters

|     |                                                                                    |
|-----|------------------------------------------------------------------------------------|
| +   | Plus; addition operator.                                                           |
| -   | Minus; subtraction operator.                                                       |
| *   | Scalar and matrix multiplication operator.                                         |
| /   | division operator                                                                  |
| .*  | Array multiplication operator.                                                     |
| ^   | Scalar and matrix exponentiation operator.                                         |
| .^  | Array exponentiation operator.                                                     |
| :   | Colon; generates regularly spaced elements and represents an entire row or column. |
| ()  | Parentheses; encloses function arguments and array indices; overrides precedence.  |
| []  | Brackets; enclosures array elements.                                               |
| .   | Decimal point.                                                                     |
| ... | Ellipsis; line-continuation operator.                                              |
| ,   | Comma; separates statements and elements in a row.                                 |

|          |                                                      |
|----------|------------------------------------------------------|
| <b>;</b> | Semicolon; separates columns and suppresses display. |
| <b>=</b> | Assignment (replacement) operator.                   |

```
>> 2^7
ans =
128
```

- A semi-colon (;) after an expression suppresses the output.

```
>> 2+5
ans =
7
>> 2+5 ;
```

- MATLAB has most standard mathematical functions built-in. The **sqrt** function computes the square root.

```
>> sqrt(2)
ans =
1.4142
```

- The basic trigonometric functions (**cos**, **sin**, **tan**, **sec**, **csc**, **cot**), their inverses (**acos**, **asin**, **atan**, **asec**, **acsc**, **acot**), the exponential function **exp**, and the natural logarithm **log** are also built-in. For instance, **ln(4)+cos(pi/6)** is computed as follows.

```
>> log(4)+cos(pi/6)
ans =
2.2523
```

- For information about any MATLAB function, type **help** followed by the name of the function.

```
>> help abs
ABS Absolute value.
ABS(X) is the absolute value of the elements of X. When
X is complex, ABS(X) is the complex modulus (magnitude) of
the elements of X.
```

- To avoid having to retype long expressions use the up arrow key ↑ to scroll through lines previously typed. Typing one or more characters and then the up arrow key displays previous lines that begin with those characters.

### 3 Variables

- To assign a value to a variable in MATLAB simply type the name of the variable, followed by the assignment operator, =, followed by the value.

```
>> x=9  
x =  
    9
```

- Note that variable names in MATLAB are case sensitive, so X and x are not equal.
- We can perform all of the usual operations with x.

```
>> x^2-3*x+2  
ans =  
    56  
>> log(x)  
ans =  
    2.1972  
>> sin(x)  
ans =  
    0.4121
```

New variables may be defined using existing variables.

```
>> y=x*3+6  
y =  
    33
```

- This, however, does not imply any permanent relationship between x and y. If we change x, the value of y does not change.

```
>> x=x+36  
x =  
    45  
>> y  
y =  
    33
```

- The command who returns a list of all variables in the current workspace, while whos returns the same list with more detailed information about each variable.

```
>> who
Your variables are:
ans x y
>> whos
```

| Name | Size | Bytes | Class  | Attributes |
|------|------|-------|--------|------------|
| ans  | 1x1  | 8     | double |            |
| x    | 1x1  | 8     | double |            |
| y    | 1x1  | 8     | double |            |

Notice that the size of each variable is 1×1. All variables in MATLAB are matrices. Scalars such as x and y are just 1×1 matrices. We will explain how to enter matrices in the next section. To clear one or more variables from the workspace, type **clear** followed by the names of the variables. Typing just clear clears all variables.

```
>> clear
>> who
>> x
??? Undefined function or variable 'x'.
```

### MATLAB Operations and Conventions

- Expressions follow the standard order of precedence
  - Exponentiation
  - Multiplication and division
  - Addition and subtraction
- Expressions are evaluated from left to right
- Parentheses work from inner to outer

## 4. Matrices and Vectors

To enter a matrix in MATLAB, use square brackets and separate entries within a row by spaces or colon and separate rows using semicolons.

```
>> A=[2 1 -1 8; 1 0 8 -3; 7 1 2 4]
A =
     2     1    -1     8
     1     0     8    -3
     7     1     2     4
```

Often we do not want MATLAB to display a response, especially when dealing with very large matrices. To suppress the output, place a semicolon at the end of the line. Typing

```
>> B=[2 0 -3; -1 1 3];
```

To view the contents of the variable **B**, just type its name.

```
>> B
B =
     2     0    -3
    -1     1     3
```

Vectors (column vectors) are simply matrices with a single column.

```
>> v = [ 2; 3; -4]
v =
     2
     3
    -4
```

A row vector is a matrix with a single row.

```
>> w=[3 -2 5 11]
w =
     3    -2     5    11
```

It is often necessary to define vectors with evenly spaced entries. In MATLAB, the colon (:) provides a shorthand for creating such vectors.

```
>> 2:5
ans =
     2     3     4     5
```

Typing **j:i:k** defines a row vector with increment i starting at j and ending at k.

```
>> 3:2:9
ans =
     3     5     7     9
```

In MATLAB, A' represents the transpose of the matrix A.

```
>> A=[5 -2 9; 11 7 8]
A =
     5    -2     9
    11     7     8
>> A'
ans =
     5    11
    -2     7
     9     8
```

The entry in row  $i$ , column  $j$  of a matrix A is A(i,j).

```
>> A=[3 -2 7 8; 4 3 2 1; 10 15 -2 9]
A =
     3    -2     7     8
     4     3     2     1
    10    15    -2     9

>> A(3,2)
ans =
    15
```

It is also possible to view multiple entries within any row or column. For instance, the second and fourth entries in the third row are accessed as follows.

```
>> A(3,[2 4])
ans =
    15     9
```

Row  $i$  of A is A(i,:) and column  $j$  of A is A(:,j).

```
>> A(3,:)
ans =
    10    15    -2     9
>> A(:,3)
ans =
     7
     2
    -2
```

Next we display the first, second and fourth columns.

```
>> A(:,[1 2 4])
ans =
     3    -2     8
     4     3     1
    10    15     9
```

The entries of a vector (row or column) may be accessed using a single index.

```
>> w=[7; 13; 11]
w =
    7
   13
   11
>> w(2)
ans =
   13
```

Matrices with the same number of rows may be concatenated horizontally, and matrices with the same number of columns may be concatenated vertically.

```
>> A=[1 2 3; 4 5 6]
A =
    1    2    3
    4    5    6
>> B=[7 8; 9 10]
B =
    7    8
    9   10
>> [A B]
ans =
    1    2    3    7    8
    4    5    6    9   10
>> C=[7 8 9]
C =
    7    8    9
>> [A;C]
ans =
    1    2    3
    4    5    6
    7    8    9
```

To **remove** rows or columns from a matrix, simply redefine them to be empty matrices.

```
>> A=[ 4 7 2 1 3; 8 7 12 -2 5; 11 1 14 -2 0]
A =
    4    7    2    1    3
    8    7   12   -2    5
   11    1   14   -2    0
>> A(2,:)=[]
A =
    4    7    2    1    3
   11    1   14   -2    0
>> A(:,[1 3])=[]
A =
    7    1    3
    1   -2    0
```

The following matrix operations are available.

- $+$  Addition
- $-$  Subtraction
- $*$  Multiplication
- $^$  Power
- $'$  Transpose (real) or conjugate transpose (complex)
- $.'$  transpose (real or complex)
- $\backslash$  left division
- $/$  right division

If  $\mathbf{A}$  is an invertible square matrix and  $\mathbf{b}$  is a compatible column vector, or respectively a compatible row vector, then

$\mathbf{x} = \mathbf{A} \backslash \mathbf{b}$  is the solution of  $\mathbf{A} * \mathbf{x} = \mathbf{b}$

$\mathbf{x} = \mathbf{b} / \mathbf{A}$  is the solution of  $\mathbf{x} * \mathbf{A} = \mathbf{b}$

### 3.1 Entry-Wise Operations

The matrix operations addition and subtraction are already entry-wise but the other operations are not; they are matrix operations. The other operations,  $*$ ,  $^$ ,  $\backslash$ ,  $/$  can be made to operate entry-wise by preceding them with a point.

```
>> B=[1 2 ; 3 4]
B =
     1     2
     3     4
>> B*B
ans =
     7    10
    15    22
>> B.*B
ans =
     1     4
     9    16
```

### 3.2 Matrix Building Functions

Table1: Some matrix building functions.

|       |                 |
|-------|-----------------|
| Eye   | Identity matrix |
| Zeros | Matrix of zeros |
| Ones  | Matrix of ones  |

|      |                                   |
|------|-----------------------------------|
| Triu | Upper triangular part of a matrix |
| Tril | Lower triangular part of a matrix |
| Rand | Matrix with random elements       |

For example,

```
>> zeros(2,3)
ans =
    0    0    0
    0    0    0
>> rand(4)
ans =
    0.8147    0.6324    0.9575    0.9572
    0.9058    0.0975    0.9649    0.4854
    0.1270    0.2785    0.1576    0.8003
    0.9134    0.5469    0.9706    0.1419
>> triu(A)
ans =
     2     1    -1     8
     0     0     8    -3
     0     0     2     4
```

#### 4. Complex number

Entering complex numbers from the keyboard has to be done carefully. The symbol "i" identifies the imaginary part and has to be typed immediately after the *numerical* value of the imaginary part: for example,  $2 + 3i$ . If you insert a space - for instance,  $2 + 3 i$  - it *looks* like the same expression but it will be processed as a number ( $2 + 3$ ) and a string ( $i$ ), and not as the complex number ( $2 + 3i$ ). It is also important to point out that termination with the character  $i$  only works with simple numbers, not expressions. For example, the expression  $(1 - 2i)i$  has no meaning to MATLAB. If you want to multiply a complex expression by  $i$ , you have to use the multiplication operation symbol (\*). In the example above, you must write  $(1 - 2i)*i$ . Similarly, the number  $1 - \sin(2)i$  has no meaning for MATLAB. It has to be written as  $1 - \sin(2)*i$  to make sense to the program.

Note: you can use  $j$  instead of  $i$ .

```
>> z=-3-4i
z =
   -3.0000 - 4.0000i
>> theta=angle(z)*180/pi
theta =
  -126.8699
```

## Functions for Complex Numbers

**Table2:** some complex building function.

| Command      | This returns the                                  |
|--------------|---------------------------------------------------|
| Complex(x,y) | Return a complex number $x+yi$                    |
| real(x)      | real part of a complex number                     |
| imag(x)      | imaginary part of a complex number                |
| Abs(x)       | magnitude of the complex number                   |
| angle(x)     | angle of a complex number $x$                     |
| conj(x)      | complex conjugate of the complex number $x$       |
| Cart2pol     | Convert Cartesian to Polar form of complex number |
| Pol2cart     | Convert Polar to Cartesian form of complex number |

## 5. Control Flow Statements

### 5.1 Relations

The relational operators in MATLAB are:

|    |                       |
|----|-----------------------|
| <  | Less than             |
| >  | Greater than          |
| <= | less than or equal    |
| >= | greater than or equal |
| == | Equal                 |
| ~= | not equal             |

Note that '=' is a direct assignment while '==' is the logical equal. Relations may be connected or quantified by the logical operators

|   |     |
|---|-----|
| & | and |
|   | or  |
| ~ | not |

## 5.2 Variable Controlled Loops (for)

The general form is

```
for variable = first: inc: last
    statements
end
```

If the increment 'inc' is not specified, a default value of 1 is used.

For example

```
x=[];
for i=1:4
    x=[x,i^2]
end

x =
     1
x =
     1     4
x =
     1     4     9
x =
     1     4     9    16
```

## 5.3 Relational Controlled Loops (while)

The general form is

```
while relation
    statements
end
```

```
i=0
while i<3
    i=i+1
end

i =
     1
i =
     2
i =
     3
```

## 5.4 Branching (if)

The general form is,

```
if relation
    true alternative
else
    false alternative
end
```

```
if 1>2
    a=1
else
    a=2
end
a =
    2
```

## 5.5 Switch

MATLAB includes a switch structure. The general form is

```
switch variable
case value 1
    statement group 1
case value 2
    statement group 2
...
otherwise
    last statement group
end
```

For example, try the following *script M-file*

```
angle = 75;  
switch angle  
case 0  
    disp('East')  
case 90  
    disp('North')  
case 180  
    disp('West')  
case 270  
    disp('South')  
otherwise  
    disp('Your lost')  
end  
  
>> switchangle  
Your lost
```



The End