Lab #1 Introduction to MATLAB

1. What Is MATLAB?

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

- Math and computation.
- Algorithm development.
- Data acquisition.
- Modeling, simulation, and prototyping.
- Data analysis, exploration, and visualization.
- Scientific and engineering graphics.
- Application development, including graphical user interface building.

2. Basic Operations:

2.1 Computation

```matlab
% This is an example of arithmetic
>> 1+2
Ans =
3
```

2.2 Suppressing Display of Results

```matlab
>> 1+2;
```

2.3 The workspace

Current values are stored in the MATLAB workspace. The `who` command tells you what variable are currently defined. The `whos` command tells you more about the variables. Use the `clear` command to delete variables from the workspace.

```matlab
>> x=2;
>> y=3;
>> z=x+y
z =
5
>> who
Your variables are:
```
2.4 Command Line Editing and Recall:

- MATLAB does not work like a spreadsheet. It will not go back and repeat calculations unless you tell it to.
- A comma (,) separates statements on a line.
- Three periods (... ) continues on the next line.
- The up arrow (↑) goes to the previous command.
- The down arrow (↓) goes to the next command.
- The sideways arrows move within the command.
- MATLAB is case sensitive.

2.5 MATLAB Operations and Conventions:

- Expression follow the standard of precedence
- Exponentiation.
- Multiplication and Division.
- Addition and Subtraction.
- Expressions are evaluated from left to right.
- Paratheses work from inner to outer.
- Complex numbers are entered using the characters 'i' and 'j', unless other definitions have been assigned to them.

2.6 To ask for online help with a command, type 'help' followed by the command name. For instance

```
>> help cos
COS    Cosine.
       COS(X) is the cosine of the elements of X.
```

3. Matrices:
MATLAB works essentially with only one kind of object – a rectangular numerical matrix with possibly complex entries. All variables represent matrices. A scalar is a 1x1 matrix and a vector is a matrix with only one row or column. There are several ways to enter matrices into MATLAB including
- Entering an explicit list of elements
- Using built-in statement or function.
• Created in a diskfile with your local editor.

3.1 Generating Vectors and the Colon Notation

We can create a vector (matrix with only one row) by directly inputting the elements

\[
\begin{align*}
\text{>> } & \text{ W=[1 2 3 4]} \\
\text{W =} \\
& 1 \quad 2 \quad 3 \quad 4
\end{align*}
\]

We could also from a vector by using the colon (:) operator. "Colon Notation" is used to generate vectors and to reference submatrices. The colon notation and subscripting by integral vectors and keys to efficient manipulation in MATLAB. Creative use of these features to factorize operations lets you minimize the use of loops (which slows MATLAB down) and makes code simple and readable.

\[
\begin{align*}
\text{>> } & \text{ Q=[1:5]} \\
\text{Q =} \\
& 1 \quad 2 \quad 3 \quad 4 \quad 5
\end{align*}
\]

We could also use any increment to construct matrices

\[
\begin{align*}
\text{>> } & \text{ E=[2:2:10]} \\
\text{E =} \\
& 2 \quad 4 \quad 6 \quad 8 \quad 10
\end{align*}
\]

3.2 Generating Matrices:
Column matrices can be created with the semicolon (;)

\[
\begin{align*}
\text{>> } & \text{ A=[1,2,3;4,5,6;7,8,9]} \\
\text{A =} \\
& 1 \quad 2 \quad 3 \\
& 4 \quad 5 \quad 6 \\
& 7 \quad 8 \quad 9
\end{align*}
\]

3.3 Referencing Individual Entries:
Round parentheses are used to reference individual elements of a matrix

\[
\begin{align*}
\text{>> } & \text{ A(1,2)} \\
\text{ans =} \\
& 2
\end{align*}
\]

3.4 The colon notation can be used to access submatrices of a matrix:
\(A(1:4,3)\) is the column vector consisting of the first 4 entries of the 3rd column
3.5 Deleting Rows and Columns
You can delete rows and columns from a matrix using just a pair of square brackets. Start with

\[
\begin{bmatrix}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 10 & 11 & 12 \\
13 & 14 & 15 & 16 \\
\end{bmatrix}
\]

>> %To delete the second column of X
>> X(:,2)=[]

\[
\begin{bmatrix}
1 & 3 & 4 \\
5 & 7 & 8 \\
9 & 11 & 12 \\
13 & 15 & 16 \\
\end{bmatrix}
\]

4. Matrix Operations
The following matrix operations are available
+    addition
-    subtraction
*    multiplication
^    power
'    transpose (real) or conjugate transpose (complex)
.\    transpose (real or complex)
\    left division
/    right division

4.1 Matrix Division
If A is an invertible square matrix and b is a compatible column vector, or respectively a compatible row vector, then

\[
x = A \backslash b \text{ is the solution of } A \cdot x = b
\]
\[
x = b / A \text{ is the solution of } x \cdot A = b
\]

4.2 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 operation, *, , ^, \, / can be made to operate entry-wise by preceding them with a period.
5. Matrix Building Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identity matrix</td>
<td>Eye</td>
</tr>
<tr>
<td>Matrix of zeros</td>
<td>Zeros</td>
</tr>
<tr>
<td>Matrix of ones</td>
<td>Ones</td>
</tr>
<tr>
<td>Upper triangular part of matrix</td>
<td>Triu</td>
</tr>
<tr>
<td>Lower triangular part of matrix</td>
<td>Tril</td>
</tr>
<tr>
<td>Matrix with random elements</td>
<td>Rand</td>
</tr>
</tbody>
</table>

For example,

```
>> zeros(2,3)
ans =
     0     0     0
     0     0     0
```

```
>> A=rand(3)
A =
     0.9501    0.4860    0.4565
     0.2311    0.8913    0.0185
     0.6068    0.7621    0.8214
```

```
>> triu(A)
ans =
     0.9501    0.4860    0.4565
     0.0000    0.8913    0.0185
     0.0000    0.0000    0.8214
```
Matrices can be built from blocks, for instance try
\[ \text{A=rand(3); } \]
\[ \text{B=[A, zeros(3,2); zeros(2,3),eye(2)];} \]

6. **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 number, 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.**

6.1 **Function for Complex Numbers:**

<table>
<thead>
<tr>
<th>This return the</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return a complex number (x+yi)</td>
<td>Complex(x,y)</td>
</tr>
<tr>
<td>Real part of a complex number (x)</td>
<td>Real(x)</td>
</tr>
<tr>
<td>Imaginary part of a complex number (x)</td>
<td>imag(x)</td>
</tr>
<tr>
<td>Magnitude of the complex number (x)</td>
<td>Abs(x)</td>
</tr>
<tr>
<td>Angle of a complex number (x)</td>
<td>angle(x)</td>
</tr>
<tr>
<td>Complex conjugate of the complex number (x)</td>
<td>conj(x)</td>
</tr>
<tr>
<td>Convert Cartesian of Polar from of complex number</td>
<td>Cart2pol</td>
</tr>
<tr>
<td>Convert Polar of Cartesian from of complex number</td>
<td>Pol2cart</td>
</tr>
</tbody>
</table>

7. **Control Flow Statements**

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
Exercises (For the Report):

1. Perform the matrix products in MATLAB:

   a. \[
   \begin{bmatrix}
   2 & -7 & 1 \\
   -6 & 2 & -3 \\
   4 & 3 & 2
   \end{bmatrix}
   \begin{bmatrix}
   -1 \\
   0 \\
   2
   \end{bmatrix}
   \]

   b. \[
   \begin{bmatrix}
   1 & 1 & 0 \\
   0 & 0 & -1 \\
   5 & -1 & 0
   \end{bmatrix}
   \begin{bmatrix}
   1 \\
   3 \\
   2
   \end{bmatrix}
   \]

2. A nice example of a function that would be difficult to graph without a computer or calculator is

   \[ F(x) = x^{(xx)} - (x^x)^x \]

   Plot \( f(x) \). (Hint: By itself, \texttt{ezplot} uses the range \(-2 \pi \leq x \leq 2 \pi \). This is inappropriate here: a better choice is to try the range \( 0 \leq x \leq 2 \).

3. Obtain plots of the following using plot function:

   \[ Y(t) = 1 - 2e^{-t} \sin(2\pi - 35^\circ), \ 0 \leq t \leq 10 \]

4. A vector \( x \) has been obtained from measurements. Suppose we want to consider any data value in the range \(-0.1 < x < 0.1\) as bearing erroneous we want to remove all such elements and replace them with zeros at the end of the array. Develop two ways of doing this. An example is given in the following table. (Hint: use function \texttt{find}, \texttt{length})

<table>
<thead>
<tr>
<th>After</th>
<th>Before</th>
<th>X(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.92</td>
<td>1.92</td>
<td></td>
</tr>
<tr>
<td>-2.43</td>
<td>0.05</td>
<td>X(2)</td>
</tr>
<tr>
<td>0.85</td>
<td>-2.43</td>
<td>X(3)</td>
</tr>
<tr>
<td>0</td>
<td>-0.02</td>
<td>X(4)</td>
</tr>
<tr>
<td>0</td>
<td>0.09</td>
<td>X(5)</td>
</tr>
<tr>
<td>0</td>
<td>0.85</td>
<td>X(6)</td>
</tr>
<tr>
<td>0</td>
<td>-0.06</td>
<td>X(7)</td>
</tr>
</tbody>
</table>