Islamic University Of Gaza
Faculty of Engineering
Electrical Engineering department

SIGNALS AND SYSTEMS LABORATORY

LABVIEW & MATLAB Programs

Eng. Mohammed Albhaisi
1.1) Introduction:

MATLAB, which stands for Matrix Laboratory, is an interactive program for numerical computation and data visualization; it is used extensively by control engineers for analysis and design. There are many different toolboxes available which extend the basic functions of MATLAB into different application areas; in addition we will make extensive use of the Control Systems Toolbox in our lab.

1.2) 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.
1.3) The MATLAB Environment:

1.4) The program interface components:

- **Command Window:**
To execute commands in the MATLAB environment.

- **Current Directory Window:**
To quickly access files on the MATLAB path.

- **Workspace Window**
To view variable definitions and variable memory allocations.

- **Command History Window**
Displays all commands were used in MATLAB since the last session.

- **Figure Window:**
To display graphical output from MATLAB code.

1.5) Making Folders:
- Use folders to keep your programs organized.
- To make a new folder, click the ‘Action’ button next to ‘Current Directory’.
- Click the ‘New Folder’ button, and change the name of the folder.
- Highlight the folder you just made and click ‘OK’.
- The current directory is now the folder you just created.
1.6) Help/Docs:

- Help:
The most important function for learning MATLAB on your own
- To get info on how to use a function:
  ```
  »help sin
  ```
  Help lists related functions at the bottom and links to the doc
- To get a nicer version of help with examples and easy-to-read descriptions:
  ```
  »doc sin
  ```

1.7) Basic Operations:

- Computations:
In this document, boxes are used to indicate items directly copied from MATLAB.
When MATLAB starts the MATLAB prompt `>>` appears, all MATLAB commands are executed from this prompt.
Unless you assign the result to a variable name, MATLAB assigns the result to variable name ‘ans’.
A percent (%) sign is a comment and is ignored.

```
>> % This is an example of mathematical operation
>> 2*3
ans = 6
```

- Suppress Display of Results:
A semi-colon (;) after an expression suppresses the output.

```
>> % This is an example of mathematical operation
>> 2*3;
```

- The Workspace:
Current values are stored in the MATLAB workspace.

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLC</td>
<td>Delete commands from the command window</td>
</tr>
<tr>
<td>CLEAR</td>
<td>Delete variables from the workspace</td>
</tr>
</tbody>
</table>
• **Command Line Editing and Recall:**
  o The up arrow (↑) goes to the previous command.
  o The down arrow (↓) goes to the next command.
  o A comma (,) separates statements on a line.

1.8) **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.

• **Generating Vectors and the Colon Notation:**
We can create a vector (matrix with only one row) by directly inputting the elements,

```
>> a=[1, 2, 3, 4]
  a =
      1 2 3 4
```
We could also form a vector by using the colon (:) operator. “Colon notation” is

```
>> b=[1:4]
  b =
      1 2 3 4
```
```
>> c=[2: 2: 10]
  c =
      2 4 6 8 10
```

• **Generating Matrices:**
Column matrices can be created with the semicolon (;).

```
Z=[1 2 3; 4 5 6; 7 8 9]
 Z =
      1 2 3
      4 5 6
      7 8 9
```
• **Referencing Individual Entries:**
Round parentheses are used to reference individual elements of a matrix

```
» Z(3,2)
an =
   8
```

• **Accessing Sub matrices:**
The colon notation can be used to access sub matrices of a matrix.
Z(1:3,3) is the column vector consisting of the first 3 entries of the 3rd column of matrix Z.

• **Deleting Rows and Columns:**
You can delete rows and columns from a matrix using just a pair of square brackets. Start with

```
X = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
Then, to delete the second column of X, use
X(:,2) = []
This changes X to
X =
  16  2   13
  5  11   8
  9   7   12
  4  14   1
```

1.9) **Matrix Operations:**
The following matrix operations are available.

<table>
<thead>
<tr>
<th>OPERATION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
</tr>
<tr>
<td>^</td>
<td>Power</td>
</tr>
<tr>
<td>'</td>
<td>Transpose (real) or conjugate transpose (complex)</td>
</tr>
<tr>
<td>,</td>
<td>Transpose (real or complex)</td>
</tr>
<tr>
<td>\</td>
<td>Left division</td>
</tr>
<tr>
<td>/</td>
<td>Right division</td>
</tr>
</tbody>
</table>
• **Matrix Division:**
If \( A \) is an invertible square matrix and \( b \) is a compatible column vector, or respectively a compatible row vector, then
\[
x = A \backslash b \quad \text{is the solution of } A \times x = b
\]
\[
x = b/A \quad \text{is the solution of } x \times A = b
\]

• **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, *, ^, , \, / can be made to operate entry-wise by preceding them with a period.

```matlab
>> B=[1 2 ; 3 4]
B =
    1   2
    3   4

>> B*B
ans =
    7  10
   15  22

>> B.*B
ans =
    1   4
    9  16
```

1.10) **Control Flow Statements:**

• **Relations:**
The relational operators in MATLAB are:

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal</td>
</tr>
<tr>
<td>==</td>
<td>Equal</td>
</tr>
<tr>
<td>~=</td>
<td>Not equal</td>
</tr>
<tr>
<td>&amp;</td>
<td>And</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>~</td>
<td>Not</td>
</tr>
</tbody>
</table>

Note that `=` is a direct assignment while `==` is the logical equal. Relations may be connected or quantified by the logical operators.
• **Variable Controlled Loops (for):**

The general form is

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

If the increment ‘inc’ is not specified, a default value of 1 is used. For example

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

**Result:**

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

• **Relational Controlled Loops (while):**

The general form is

```plaintext
while relation
    statements
end
```

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

**Result:**

```
i =
  1
i =
  2
i =
  3
```
1.11) Plotting:

- **Basic Two-Dimensional Plot:**
  MATLAB has extensive plotting capabilities. We will examine a simple two-dimensional plot and add features. The ‘plot’ command graphs the numbers in one array versus the numbers in a second array of the same length. For example,

```matlab
% Define time array
T = 0:0.01:2;
% Define temperature array
Temp = exp(-T);
% Plot the temperature vs. time
plot(T, Temp)
xlabel('Time')
ylabel('Temp')
title('Transient Temperatures')
```

![Graph of transient temperatures](image.png)
• Multiple Plots on a Single Graph:

```matlab
x=0:pi/16:2*pi;
y1=sin(x);
y2=cos(x);
plot(x,y1,'-x',x,y2,'rs-')
xlabel('x')
ylabel('sin(x), cos(x)')
title('Trig Functions')
legend('sin','cos')
```
• **Subplots**

You can create graphics arrays using the ‘subplot’ command.

```matlab
x=0:pi/16:2*pi;
y1=sin(x);
y2=cos(x);

subplot(2,1,1)
plot(x,y1,'*-' )
xlabel('x')
ylabel('sin(x)')

subplot(2,1,2)
plot(x,y2,'rs -')
xlabel('x')
ylabel('cos(x)')
```

• **Function of plot:**

You may use the following functions; try to find out its function:

hold, grid, axis, ezplot, gtext, ginput
**Exercise**

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) \[
   \begin{bmatrix}
   1 & 1 & 0 \\
   1 & 2 & -1 \\
   5 & -1 & 0 \\
   \end{bmatrix}
   \begin{bmatrix}
   1 & 3 & 2 \\
   0 & 0 & -1 \\
   \end{bmatrix}
   \]
   
   c) \[
   \begin{bmatrix}
   0 & 4 & 2 \\
   -1 & 2 & -1 \\
   1 & 1 & 0 \\
   \end{bmatrix}
   \begin{bmatrix}
   -1 \\
   0 \\
   2 \\
   \end{bmatrix}
   \]
   
   d) \[
   \begin{bmatrix}
   1 & 1 \\
   \end{bmatrix}
   \begin{bmatrix}
   -1 & 2 & 2 \\
   -1 & 3 & -6 \\
   2 & -5 & 4 \\
   \end{bmatrix}
   \]

2. Obtain plots of the following function:

   \[ y(t) = 1 - e^{-t} \sin(2\pi t - 90), \quad 0 \leq t \leq 10 \]

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

   \[ f(x) = x^{\sin x} - (x^x)^x \]

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

4. Write the code that makes the following plot: