TRIBHUVAN UNIVERSITY

Institute of Engineering

Central Campus, Pulchowk

Lab Report On:

Signal Analysis

Submitted To:

Department of Electronics and Computer Engineering

Submitted By:

Rajil Bajracharya (073BEX430)

Submitted On:

10-06-019

BASICS OF MATLAB

1.1 Objectives

- To understand the basic syntax structure of MATLAB.
- To learn to use the command window for debugging and using help.
- To learn about functions and use user-defined functions to solve problems in MATLAB.
- To use library functions of MATLAB to handle matrices and complex numbers.
- To learn to use MATLAB to plot various common signals.

1.2 Theory

1.1.1 About MATLAB:

MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, C#, Java, Fortran and Python. Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine, allowing access to symbolic computing abilities. An additional package, Simulink, adds graphical multi-domain simulation and model-based design for dynamic and embedded systems.

1.1.2 Variables and Operators:

The variables in MATLAB are quite different compared to other programming languages in that all the variables declared in MATLAB are actually matrices. Ergo, MATLAB supports a plethora of operators and functions that deal with matrices implicitly as well as explicitly. Following are some of the most commonly used operators in MATLAB:

Operator	Purpose
+	Plus; addition operator.
-	Minus; subtraction operator.
*	Scalar and matrix multiplication operator.
.*	Array multiplication operator.
^	Scalar and matrix exponentiation operator.
.^	Array exponentiation operator.

\	Left-division operator.
/	Right-division operator.
.\	Array left-division operator.
./	Array right-division 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
;	Semicolon; separates columns and suppresses display.
%	Percent sign; designates a comment and specifies formatting.
_	Quote sign and transpose operator.
._	Nonconjugated transpose operator.
=	Assignment operator.

MATLAB also supports some special variables such as:

Name	Meaning
ans	Most recent answer.
eps	Accuracy of floating-point precision.
i,j	The imaginary unit √-1.
Inf	Infinity.
NaN	Undefined numerical result (not a number).
pi	The number π

1.1.3 Functions:

MATLAB supports a wide range in-built functions local to the type of data type being operated on. The list of library functions vary according to the data type and can be read about in Mathworks' documentation page At the same time, it also allows user-defined functions. The user-defined function must be saved as a file whose name is the name of the function. The syntax for defining functions inn MATLAB is:

```
function [output argument list] = [input argument list]
                    {function body}
end
```

1.1.4 Plotting:

MATLAB allows us to plot various signals using its graphical toolbox. Functions can be plotted in MATLAB by evaluating the dependent variable at distinct values of the independent variable. Both the dependent and the independent variables have to be arrays/matrices in order to plot the points using MATLAB's library function called plot(). For plotting a point (x, y) on the Cartesian plane for various values of y corresponding to the same no. of values of x, one simply uses the command plot(x, y). The whole of the graphical plane in MATLAB can also be divided into various subplots each with a different type of function, by including the function subplot(m, n, p) above each plot function. The result is that the subplot() function divides the plot area into a m X n separate sub-areas and selects the p^{th} one among them for the plot statement that immediately follows.

1.1.5 Loops:

The for loop is the most common type of loop used in MATLAB, with syntax:

```
for <variable>=<start value>: [step size]: <stop value>
                    <loop body>
end
```

1.1.6 Conditional statement (if):

The if statement is the most common type of conditional statement in MATLAB. Its syntax is:

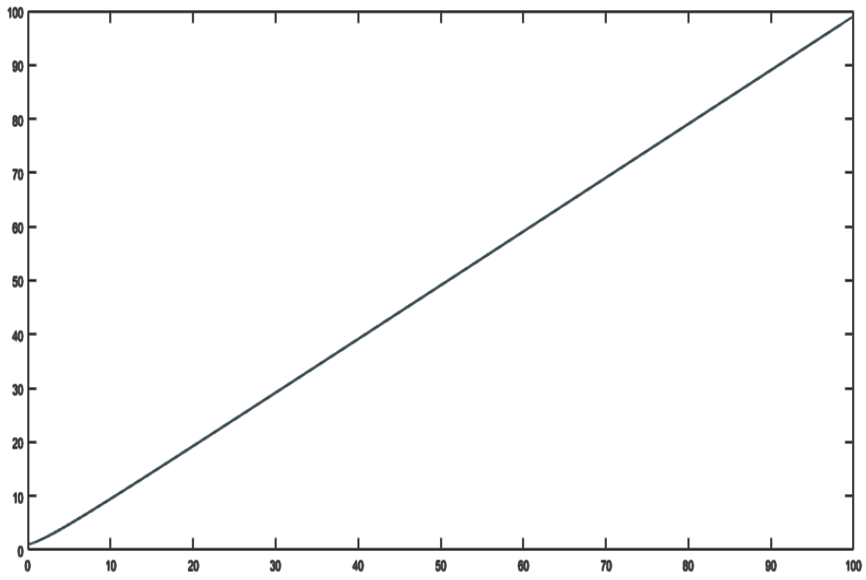
```
if <condition>
   <statement(s)>
[elseif <condition>]
   [<statement(s)>]
...
[<else>]
   [<statements>]
end
```

1.3 Code and Outputs:

1.3.1

```
%plot f(x) = (x^2 + 2x + 3)/(x + 3) for 0<x<=100
Fs = 10;
dt = 1/Fs;
x = 0:dt:100;
f = (x .* x + 2 * x + 3) ./ (x + 3);
plot(x, f);
```

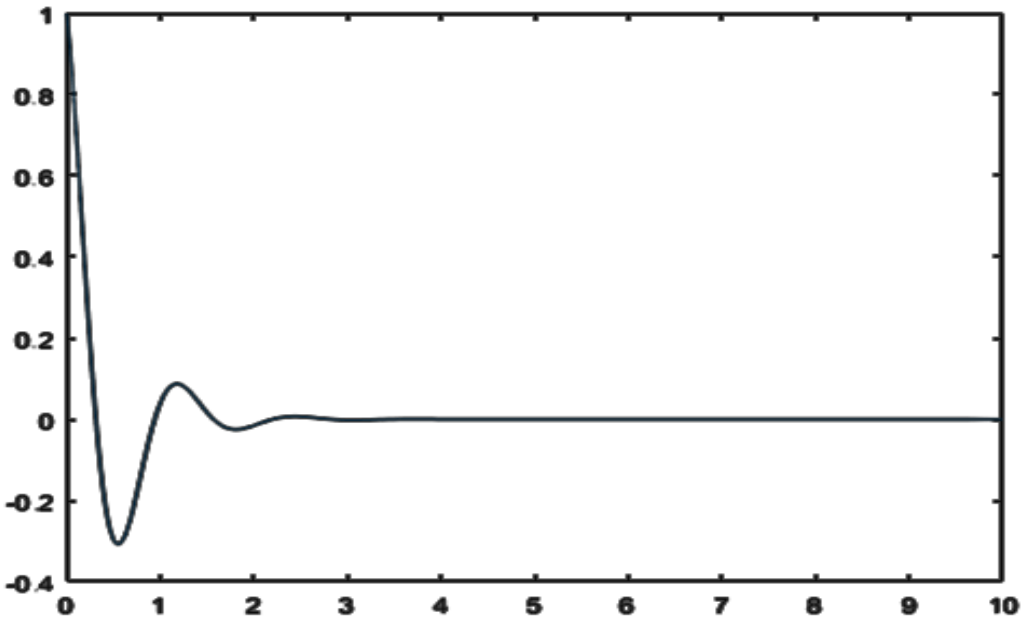
OUTPUT:



1.3.2

```
%plot f(x) = exp(-at) * cos(wt); w = 5, a = 2, 0 < t <= 10
Fs = 100;
dt = 1/Fs;
t = 0: dt: 10;
w = 5;
a = 2;
f = exp(-a * t) .* cos(w * t);
plot(t, f);
```

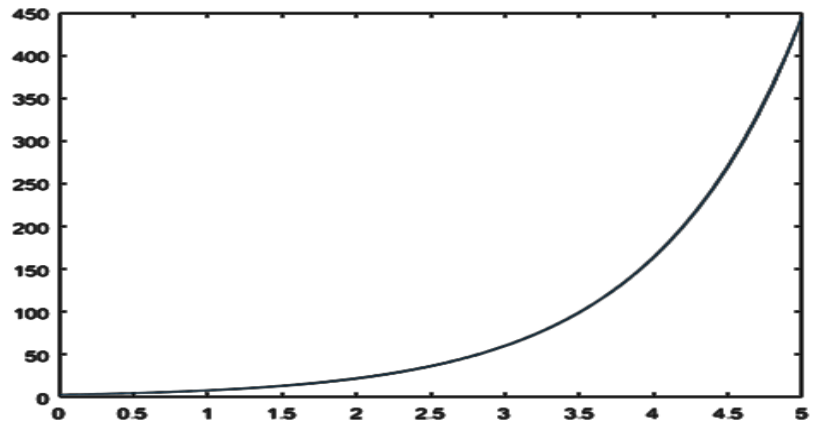
OUTPUT:



1.3.3

```
%plot y = 3e^x
Fs = 10;
dt = 1/Fs;
x = 0: dt: 5;
y = 3 * exp(x);
plot(x, y);
```

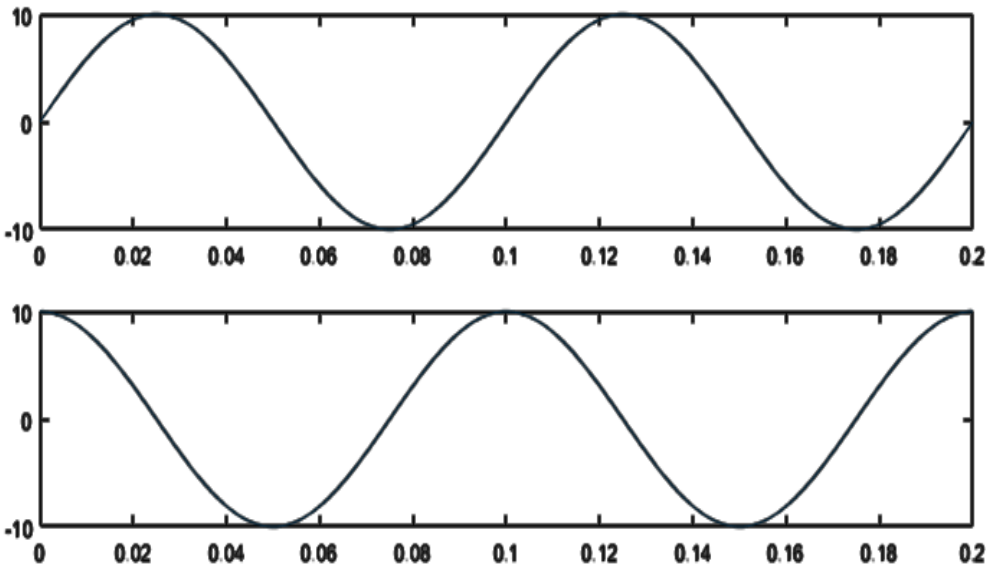
OUTPUT:



1.3.4

```
%plot of sine and cosine curves
a = 10;
Fs = 100;
Fc = 10;
dt = 1/Fs;
t = 0: dt: 2 * Fc;
ysin = a * sin(2*pi*Fc/Fs * t);
ycos = a * cos(2*pi*Fc/Fs * t);
subplot(2,1,1);
plot(t/Fs, ysin);
subplot(2,1,2);
plot(t/Fs, ycos);
```

OUTPUT:



1.3.5

```
%finding factorial of a number using function
N = input('Enter a number:');
fprintf('The factorial of %d is %ld.\n', N,
factorial(N));
%factorial.m
function y = factorial(N)
    if N <= 1
        y = 1;
    else
        y = N * factorial(N - 1);
    end
end
```

OUTPUT:

```
Enter a number:10
The factorial of 10 is 3628800.
```

1.3.6

```
%generate the fibonacci sequence
first = input('Enter the first number: ');
second = input('Enter the second number: ');
terms = input('Enter the no. of terms: ');
fibonacci(first, second, terms);
%fibonacci.m
function [] = fibonacci(first, second, terms)
%fibonacci function takes the first two args and
%generates 'term' no. of Fibonacci terms
    fprintf('%d %d ', first, second);
    t1 = first;
    t2 = second;
    for cnt = 3:terms
        t3 = t1 + t2;
        fprintf('%d ', t3);
        t1 = t2;
        t2 = t3;
    end
    fprintf('\n');
end
```

OUTPUT:

```
Enter the first number: 2
Enter the second number: 3
Enter the no. of terms: 15
2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597
```

1.3.7

```
%basic functions operating on a single matrix
a = [2, 3, 5; 1, 2, 9; 9, 8, 1];
disp(a)
disp(size(a))
disp(numel(a))
disp(sum(a))
disp(mean(a))
b = 2: 10;
disp(b)
disp(size(b))
disp(numel(b))
disp(sum(b))
disp(mean(b))
```

OUTPUT:

```
2     3     5
1     2     9
9     8     1

3     3

9
```

```
12    13    15

4.0000    4.3333    5.0000

2     3     4     5     6     7     8     9     10

1     9

9

54

6
```

1.3.8

```
%basic operations on complex numbers
num = input('Enter a complex number:');
disp(num);
real_part = real(num);
imag_part = imag(num);
magnitude = abs(num);
angle_r = angle(num);
angle_d = angle_r * 180/pi;
disp('(x, y) => r /_theta')
fprintf('(%.2f, %.2f) => %.2f /_%.2f\n', real_part,
imag_part, magnitude, angle_d);
```

OUTPUT:

```
Enter a complex number:3+4i

3.0000 + 4.0000i

(x, y) => r /_theta

(3.00, 4.00) => 5.00 /_53.13
```

1.3.9

```
%basic matrix operations:
matA = [1, 5, 7; 4, 1, 3; 1, 4, 5];
matB = 3:5;
matB = matB';
disp('MatA = ')
disp(matA)
disp('MatB = ')
disp(matB)
disp('Transpose of A:')
disp(matA')
disp('Transpose of B:')
disp(transpose(matB))
disp('Inverse of A:')
disp(inv(matA))
disp('Scalar multiplication of MatA with itself:')
disp(matA.*matA)
```

```
disp('Matrix Multiplication of MatA and MatB:')
disp(matA * matB)
```

OUTPUT:

```
MatA =
     1     5     7
     4     1     3
     1     4     5
MatB =
     3
     4
     5
Transpose of A:
     1     4     1
     5     1     4
     7     3     5
Transpose of B:
     3     4     5
Inverse of A:
   -0.5385    0.2308    0.6154
   -1.3077   -0.1538    1.9231
    1.1538    0.0769   -1.4615
Scalar multiplication of MatA with itself:
     1    25    49
    16     1     9
     1    16    25
Matrix Multiplication of MatA and MatB:
    58
    31
    44
```

1.4 Conclusion

In this way, we used MATLAB to plot various functions, manipulate matrices, operate on complex numbers, generate a sequence using functions, loops and conditions; as well as the factorial of a number. Through these programs, we learnt to use MATLAB to plot functions and solve simple problems in programming.