Signal and System Lab

Department of Computer Engineering

Khwaja Fareed University of Engineering and
Information Technology
Rahim Yar Khan, Pakistan

Contents

Chapter 1

LAB NO. 01

1.1 Introduction to Matlab

PERFORMANCE OBJECTIVES
Upon successful completion of this experiment student will be able to:
Become familiar with MATLAB environment
Become familiar with Basic MATLAB commands
Write small programs using MATLAB

EQUIPMENT REQUIRED:
A computer running Matlab 7.0 or higher version
THEORETICAL BACKGROUND:
Matlab is an interactive system for doing numerical computations. A numerical analyst called Cleve Moler wrote the first version of Matlab in the 1970s. It has since evolved into a successful commercial software package. Matlab relieves you of a lot of the mundane tasks associated with solving problems numerically. This allows you to spend more time thinking, and encourages you to experiment. Matlab makes use of highly respected algorithms and hence you can be confident about your results. Powerful operations can be performed using just one or two commands. You can build up your own set of functions for a particular application. Excellent graphics facilities are available, and the pictures can be inserted into LATEX and Word documents.

MATLAB is a high level computer language used for the programming of complex engineering problems. MATLAB stands for MATrix LABoratory. In MATLAB we have many tool boxes each containing functions related to specific operations, just as we have libraries in JAVA or C language. Communications, Control System, Image processing and Image acquisition are some of the most commonly used tool boxes. MATLAB has extensive capabilities of two and three dimensional plotting. We will use these plotting facilities of MATLAB to aid in the physical interpretation of equations, particularly those related to fields and potentials. In MATLAB, there are two modes of executing a program; the first one is command line execution and the second one is M-File or DOT M (.m) file execution. For command line execution commands are entered in the command window where (.m) file, complete program is first written in a file and saved with DOT M (.m) extension. The complete program is then RUN like any other programming language. We will use DOT M files throughout this lab.
BASIC TERMINOLOGY:
MATLAB has some basic terms and predefined variables you should get familiar with before going any further. One of these is the variable ans. When you type most commands to MATLAB, they will return values. You can assign these values to variables by typing in equations. For example, if you type

»x = 5
MATLAB will print
x =
5
If you terminate a command with a semi-colon, MATLAB will suppress the printing of the variable name and value resulting from the calculation. For example, if you type

»x = 5;
MATLAB will assign the value five to the variable x, but rather than tell you it did that, it will just return another « prompt. MATLAB works with two basic types of data objects: scalars and matrices. MATLAB also has vectors. Vectors are a special case of matrices which are only 1 row by any number of columns. To assign x to be a matrix by explicitly entering the elements, you type the list of elements separated by blanks or commas surrounded by [and], and use semi-colons to separate the rows. For example, typing

»x = [2 4 6 8 ; 1 3 5 7]
Results in
x = 2 4 6 8
1 3 5 7

The MATLAB workspace is defined as the collection of all the variables you have defined during the current MATLAB session.

BASIC ARITHMETIC:
<inserted text> MATLAB uses a straightforward notation for basic arithmetic on scalars. The symbol + is used to add scalars, so $x = 1 + 5$ will give x the value 6. Similarly, MATLAB uses $-$ for subtraction, * for multiplication,/ for division, and $\hat{}$for exponentiation.All of these work for two scalars. In addition, you can add, subtract, multiply or divide all the elements of a vector or matrix by a scalar. For example, if x is a matrix or vector, then x+1 will add one to each element of x, and x/2 will divide each element of x by 2. all of the basic operations ($_{7}$ $-$, , /, and $\hat{}$) are defined to work with complex scalars. Another useful operator is the colon. You can use the colon to specify a range of numbers. Typing

»x = 1 : 4

Will return

X=
1 2 3 4

MATLAB HELP:
MATLAB has a fairly good help facility. The help function knows about all the commands listed in this manual. Typing help function will tell you the syntax for the function, i.e. what arguments it expects. It will also give you a short description of claims you are in error, try looking at the help for the functions you are using.
Check MATLAB help for PLOT, FOR, SUBPLOT, ROOTS.
»help plot
»help for
»help subplot
»help roots

Try the following commands
»doc plot
»doc for
»doc subplot
»doc roots
roots
Polynomial roots
Syntax
r = roots(c)
Description
r = roots(c) returns a column vector whose elements are the roots of the polynomial c. Row vector c contains the coefficients of a polynomial, ordered in descending powers. If c has n+1 components, the polynomial it represents is $C1Sn + C2Sn - 1 + C3Sn - 2 + + CnS + Cn + 1$
Examples:- The polynomial $S3 + S2 + S - 27$ is represented in MATLAB as
p = [1 -6 -72 -27]
The roots of this polynomial are returned in a column vector by
»r = roots(p)
r =
12.1229
-5.7345
-0.3884

Conclusion:

Summarize, in a paragraph or two, what you conclude from the results of your experiment and whether they are what you expected them to be. Compare the results with theoretical expectations and include percent error when appropriate. Don't use terms such as "fairly close" and "pretty good;" give explicit quantitative deviations from the expected result. Evaluate whether these deviations fall within your expected errors and state possible explanations for unusual deviations. Discuss and comment on the results and conclusions drawn, including the sources of the errors and the methods used for estimating them.

Chapter 2

LAB NO. 02

2.1 Basics Of MATLAB

PERFORMANCE OBJECTIVES:
Upon successful completion of this experiment student will be able to:
Become familiar with MATLAB environment Know the basics of MATLAB EQUIPMENT REQUIRED:
A computer running Matlab 7.0 or higher version
THEORETICAL BACKGROUND:
In this lab section we will discuss basics of MATLAB that will able to use different functions.
Suppressing the output:
One often does not want to see the result of intermediate calculations—terminate the assignment statement or expression with semi-colon » $x = -13; y = 5 * x, z = x^2 + y$
y =
-65
z =
104
»
The value of x is hidden. Note also we can place several statements on one line, separated by commas or semi-colons. Products, Division and Powers of Vectors We can perform this product in Matlab by
»$u = [10, -11, 12], v = [20; -21; -22]$ » prod = u*v
prod =
167
»
Suppose we also define a row vector w and a column vector z by
» w = [2, 1, 3], z = [7; 6; 5]
w =
2 1 3
z =
7
6
5
and we wish to form the scalar products of u with w and v with z.
» u*w
??? Error using ==>
Inner matrix dimensions must agree.

There is no mathematical definition for the division of one vector by another. However, in Matlab, the operator ./ is defined to give element by element divisionâĂŤit is therefore only defined for vectors of the same size and type.

» a = 1:5, b = 6:10, a./b
a =
1 2 3 4 5
b =
6 7 8 9 10
ans =
0.1667 0.2857 0.3750 0.4444 0.5000
» a./a
ans =
1 1 1 1 1
» c = -2:2, a./c
c =
-2 -1 0 1 2
Warning: Divide by zero
ans =
-0.5000 -2.0000 Inf 4.0000 2.5000

User Defined Function

Now we will draw the user defined function. i.e if we are to draw the exponential function $x(t) = e^(at + jt) foraandinputsfort.$

MATLABB CODE

$a = input('enterthea =');$
$b = input('enterthephi =');$
$T_min = input('Enterthestartingtime =');$
$T_ss = input('EntertheStepSizeontimeinterval =');$
$T_max = input('Entertheendingtime =');$
$t = T_min : T_ss : T_max;$
» title('Graph of complex exponential)')
» xlabel('x axis')
» ylabel('y-axis')
$xt = exp(-a. * t + i * b. * t); plot(t, xt)$

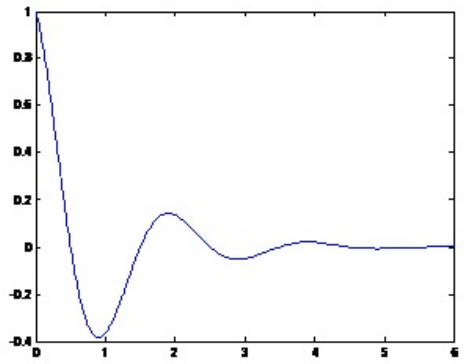


Figure 2.1: Figure 2.01 plot for a given values of A and Îÿ

Conclusion:
Summarize, in a paragraph or two, what you conclude from the results of your experiment and whether they are what you expected them to be. Compare the results with theoretical expectations and include percent error when appropriate. Don't use terms such as "fairly close" and "pretty good;" give explicit quantitative deviations from the expected result. Evaluate whether these deviations fall within your expected errors and state possible explanations for unusual deviations. Discuss and comment on the results and conclusions drawn, including the sources of the errors and the methods used for estimating them.

Chapter 3

LAB NO. 03

3.1 Introduction to SIMULINK in MATLAB

PERFORMANCE OBJECTIVES
Upon successful completion of this experiment student will be able to:
Become familiar with MATLAB environment
Become familiar with simulink

EQUIPMENT REQUIRED:
A computer running Matlab 7.0 or higher version

THEORETICAL BACKGROUND:

The MATLAB and Simulink environments are integrated into one entity, and thus we can analyze, simulate, and revise our models in either environment at any point. We invoke Simulink from within MATLAB. We begin with a few examples and we will discuss generalities in subsequent chapters. Throughout this text, a left justified horizontal bar will denote the beginning of an example, and a right justified horizontal bar will denote the end of the example. These bars will not be shown whenever an example begins at the top of a page or at the bottom of a page. Also, when one example follows immediately after a previous example, the right justified bar will be omitted. In the Command Window, we type:
»simulink
Exercise with Simulink:

Now we are to draw the sine wave and cosine wave on Simulink.

```
$y_1= Sin x and y_2= cos x$    >>> (2.11)\\
```

Scope Result

To generate the cos wave we have to change the parameter as follows
$Phase(rad) = pi/2$

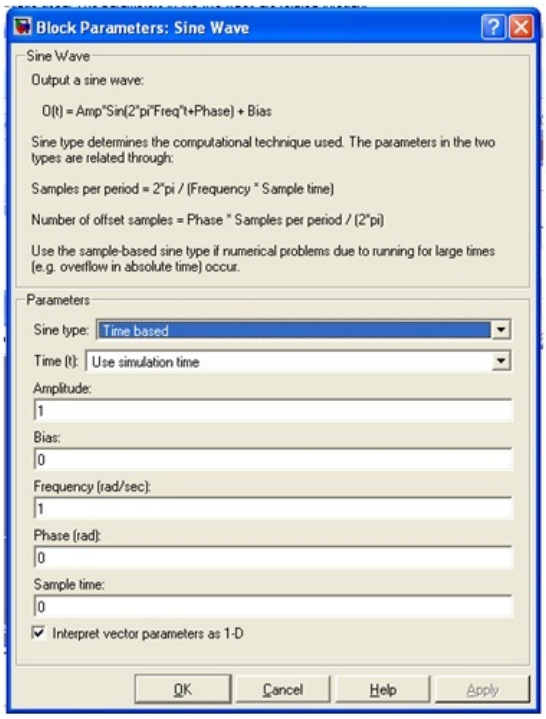


Figure 3.1: Dragging the Sine Wave block function into File 'Untitled'

Conclusion:
Summarize, in a paragraph or two, what you conclude from the results of your experiment and whether they are what you expected them to be. Compare the results with theoretical expectations and include percent error when appropriate. Don't use terms such as "fairly close" and "pretty good;" give explicit quantitative deviations from the expected result. Evaluate whether these deviations fall within your expected errors and state possible explanations for unusual deviations. Discuss and comment on the results and conclusions drawn, including the sources of the errors and the methods used for estimating them.

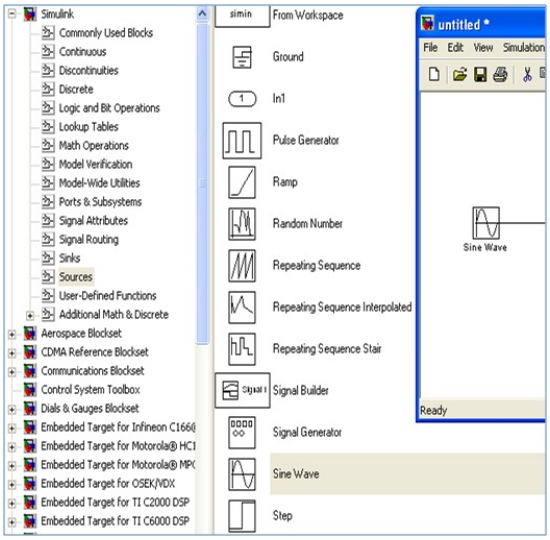


Figure 3.2: Dragging the Sine Wave block function into File 'Untitled'

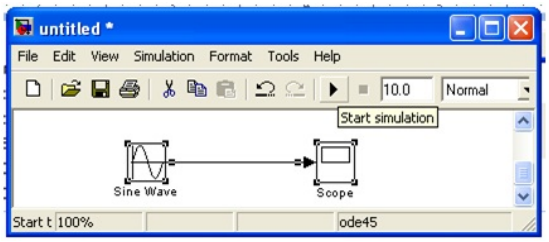


Figure 3.3: File Equation with added Sine wave and Scope block

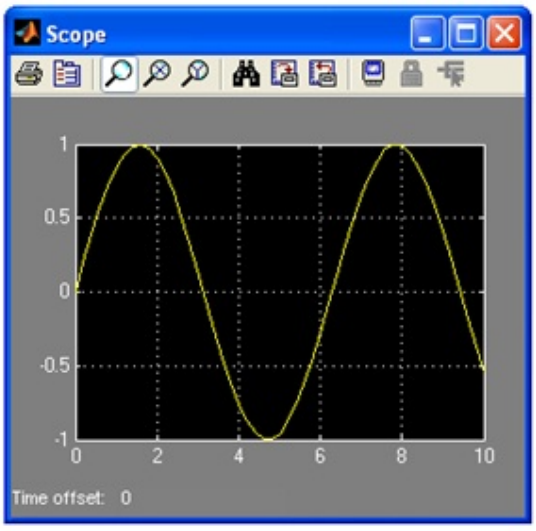


Figure 3.4: Sine wave

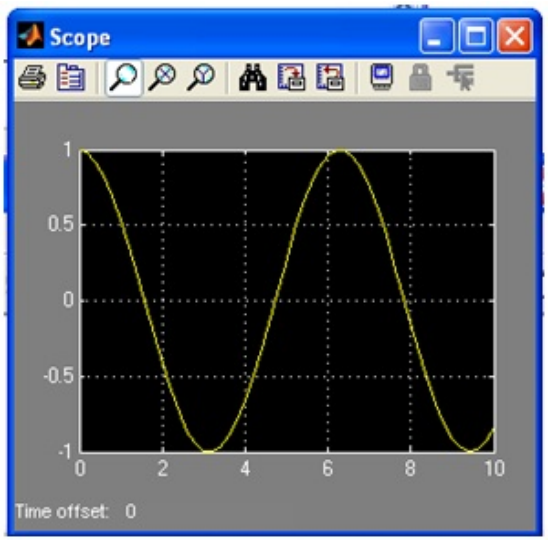


Figure 3.5: Cos wave

Chapter 4

LAB NO. 04

4.1 Arrays Basic Signals And Sounds In MATALB

PERFORMANCE OBJECTIVES
Upon successful completion of this experiment student will be able to:
Explore arrays t o use them to construct the signals and arrays
Familiar with the fundamentals of MATLAB
EQUIPMENT REQUIRED
A computer running Matlab 7.0 or higher version

THEORETICAL BACKGROUND:
Matlab provides an online help system accessible by using the help command. For example to get information size enter the following
»help size
There is help desk desk with useful introductory material. it is accessed from the HEPL menu. Now hare are exercise for the âĂIJArraysâĂİ

ARRAYS:
A variable in Matlab is an array. An array has dimension NÃŮM, where N and M are in naturals.
N= no. of rows, M=no. of columns
If M=N=1 then variable is scalar, and if N=1 and M>1 the variable is the row vector, if N.1 and m=1 then variable is the column vector, if both M and N are greater than one, then variable is a matrix, and if M=N >1 then itâĂŹs a square matrix. The coefficients of the matrix may b real or complex numbers.
Following are the example to create the arrays.

»Array1=[1 2 3 4 5 6]

Array1 =

1 2 3 4 5 6
» array2=[1 2 3; 4 5 6; 7 8 9]

array2 =

1 2 3
4 5 6

7 8 9
» array3=[1; 2; 3; 4]
array3 =
1
2
3
4
Now we will create the array by using the step size
» v = [1 3, sqrt(5)]
v =
1.0000 3.0000 2.2361
» length(v)
ans =
3
Spaces can be vitally important:
»$v2 = [3 + 45]$
v2 =
7 5
» $v3 = [3 + 45]$
v3 =
3 4 5
» array3=0:5
array3 =
0 1 2 3 4 5
» start=0;
» step=0.25;
» stop=4;
» array4=start:step:stop
MATLAB will return us
array4 =
Columns 1 through 7
0 0.2500 0.5000 0.7500 1.0000 1.2500 1.5000
Columns 8 through 14
1.7500 2.0000 2.2500 2.5000 2.7500 3.0000 3.2500
Columns 15 through 17
3.5000 3.7500 4.0000
We can sum up the entire array by using the command
» sum4=sum(array4)
sum4 =
34
Any built-in function that that operates on scalar can also operate on arrays. For example
»sin (pi/4)
Ans=
0.7071
»$sin([0pi/4pi/23 * pi/4pi])$
Ans=
0 0.707 1.00 0.7071
This feature is called vectorization. A discrete- time signal may be approximated in Matlab. We will built a few such vectors and plot.

»$u = zeros(1, 10)$ u =

0 0 0 0 0 0 0 0 0 0
» u(5)=1
» stem(u)

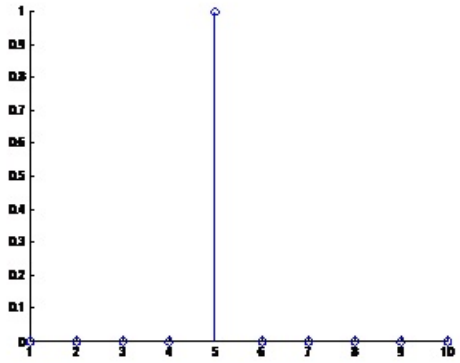


Figure 4.1: Stem of given discrete signal

Now we will sketch the sine wave x:[-1, 1] reals, given by
$x(t) = sine(25t + /6)$
MATLAB CODE
» t=-1:.0005:1;
» xt=sin(2* pi* 5.*t +pi/6);
» plot(t,xt,'b–')
»

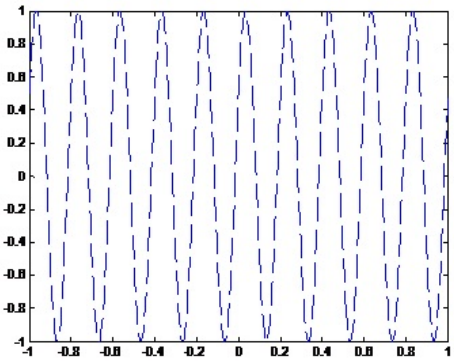


Figure 4.2: plot of given signal

MATLAB function sound
Syntax
»sound (sampledSignal, frequency)
Sends the one dimension array or vector sampledSignal to the audio card im the pc.the 2nd

argument specifies the sampling frequency in Hz. The value in the sampledSignal are assumed to be real no in the range [-1.00 100]. Values out side this range are clipped to -1 to 1. We will use this function to listen the signal.
» sound(xt, 2)
» sound(2* xt, 5)
» t=0:0.005:1;
» $ft = exp(-5. * t). * sin(2 * pi * 440 * t);$
» plot(ft)
» plot(t,ft)
» sound(ft, 8000)

Special Matrices
Matlab provides a number of useful built-in matrices of any desired size.
ones(m,n) gives an m * n matrix of 1's,
» P = ones(2,3)
P =
1 1 1
1 1 1

Conclusion:

Summarize, in a paragraph or two, what you conclude from the results of your experiment and whether they are what you expected them to be. Compare the results with theoretical expectations and include percent error when appropriate. Don't use terms such as "fairly close" and "pretty good;" give explicit quantitative deviations from the expected result. Evaluate whether these deviations fall within your expected errors and state possible explanations for unusual deviations. Discuss and comment on the results and conclusions drawn, including the sources of the errors and the methods used for estimating them.

Chapter 5

LAB NO. 05

5.1 Calculation Of Signal Power

PERFORMANCE OBJECTIVES
Upon successful completion of this experiment student will be able to:
See how to calculate average of power of continuous and discrete time periodic signals. Let us remember again that we can only simulate continuous time case in a digital environment like Matlab.
EQUIPMENT REQUIRED:
A computer running Matlab 7.0 or higher version

THEORETICAL BACKGROUND:
Part 1: Continuous Time Case
Here, we will simulate or generate one period of a continuous time cosine and calculate its average power by using the formula
We first generate a fine time axis extending from âĂŞ1 to 0.995 as follows.

$$P = \frac{1}{T}\int_{-T/2}^{T/2} |x(t)|^2 dt$$

Figure 5.1:

```
t=-1:0.005:0.995;
```

We selected the period T to be 2 and to cover one period exactly, we selected the upper limit of our time interval as 0.995. Now, we form our cosine signal and plot it.

```
xt=cos(2*pi*t/2);
plot(t,xt)
```

We then element-wise square xt and take the absolute value.

```
absxt=abs(xt.^2);
```

Note the use of dot (.) operator above. When it encounters a multiplication, division

or power operation, Matlab normally tries to do the corresponding matrix operations but we can override this and make those operations element-wise by putting a dot in front of them. For a real signal, after squaring taking the absolute value does not make sense or is not necessary but for a complex signal this step is necessary as dictated by the formula above. We also note that the order of squaring and absolute value operations can be reversed or changed. Therefore, what we did by MCL 4 can also be accomplished by the following line.

absxt=abs(xt).$\hat{}$2;

Now, we will use Euler's approximation to carry out the integral. This formula simply tells that we can approximate a definite integral using a sum. In this technique, we divide the

$$\int_a^b x(t)dt \cong \sum_{n=0}^{N-1} x(a+n\Delta t)\Delta t, \quad \Delta t = \frac{b-a}{N}$$

Figure 5.2:

region over which we will do the integral into N parts or intervals, each of duration t, and assume that our function stays constant over those short intervals. Approximating a function in this (staircase) manner is depicted in Figure below. As we increase the number of intervals N, the approximation gets better.
Approximating integrals using sums as we did above, is a deep subject of numerical analysis

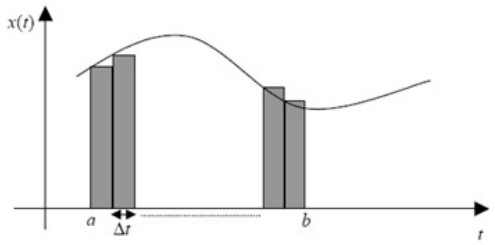


Figure 5.3: EulerâĂŹs approximation

or methods by itself, therefore we will not get into more detail here. It suffices for us to know that Euler's formula or technique is easy to implement and produces good results for almost all the signals that we will deal with, as long as N is selected large enough. Back to our main subject, to approximate or compute the average power of our cosine signal, we first note that we have already selected t as 0.005. (Can you tell where we did this? .Let us now define this parameter and the period T.
deltat=0.005; T=2;
Finally we do the sum and multiply it with t/T.
pxt=sum(absxt)*deltat/T pxt =
0.5000
Part 2: Discrete Time Case
We will now generate one period of a discrete time cosine and calculate its average power using the following formula

$$P = \frac{1}{N}\sum_{n=0}^{N-1}\left|x[n]\right|^2$$

Figure 5.4:

We create a vector of indices first and form our discrete time cosine signal as follows.

```
n=0:19;
xn=cos(2*pi*n/20);
stem(n,xn)
```

To cover exactly one period of the cosine we let the index run from 0 to 19, as the period N is 20. The formula above to be implemented is already a sum and the calculation of average power is really easy this time, as we are not dealing with an integral.

```
N=20; absxn=abs(xn.^2);
pxn=sum(absxn)/N
pxn =
0.5000
```

Conclusion:

Summarize, in a paragraph or two, what you conclude from the results of your experiment and whether they are what you expected them to be. Compare the results with theoretical expectations and include percent error when appropriate. Don't use terms such as "fairly close" and "pretty good;" give explicit quantitative deviations from the expected result. Evaluate whether these deviations fall within your expected errors and state possible explanations for unusual deviations. Discuss and comment on the results and conclusions drawn, including the sources of the errors and the methods used for estimating them.

Chapter 6

LAB NO. 06

6.1 Transformations of the Independent variable

PERFORMANCE OBJECTIVES

Upon successful completion of this experiment student will be able to:

In this project, we will see how to implement a class of simple signal transformations that are based on the modification of the independent variable, which is mostly the 'time' in the context of signals and systems

EQUIPMENT REQUIRED:

A computer running Matlab 7.0 or higher version

THEORETICAL BACKGROUND:

In order to see the effects of transformations on the independent variable, we will first generate a test signal that we will play with. Our test signal will be as shown in Figure. As our signal has different linear segments, we can generate those parts one by one and then combine them to obtain the whole signal. We will generate the segments of time axis first and then use linspace1 commands to create the corresponding signal segments. Linspace command is useful generating linearly changing or equally spaced points between two limits. The two limits can be the same, in which case the linspace command produces vectors filled with that same limit or constant.

The 1st segment of the signal goes from t =-2 to 0.

t1=-2:0.05:0;

MCL 1 We have just generated a time vector of length 41. Now, we will create the corresponding signal segment.

x1=linspace(0,0,41);

MCL 2

It is better to use Matlab's capabilities wherever possible. One little service that we can get from Matlab is to ask for the length of vectors via the length command. If we utilize this at MCL 2, it would be as follows.

x1=linspace(0,0,length(t1));

Instead of using MCL 2, we could have also generated x1, the 1st signal segment, by the following method that utilizes Matlab's zeros2 utility.

x1=zeros(1,length(t1));

We can now generate the other segments in a similar way.

t2=0.05:0.05:1; MCL 3

x2=linspace(1,1,length(t2)); MCL 4

t3=1.05:0.05:2; MCL 5

x3=linspace(1,0,length(t3)); MCL 6
t4=2.05:0.05:3; MCL 7
x4=linspace(0,0,length(t4)); MCL 8
Note how we selected the limits of consecutive time segments above.
We can now combine both time and signal segments into single vectors and do a plot of our signal.
t=[t1 t2 t3 t4]; MCL 9
x=[x1 x2 x3 x4]; MCL 10
plot(t,x); MCL 11
We observe in Figure 1 that Matlab's automatic axis scaling did not produce a good-looking plot. We can override Matlab's default axes scales by using the axis3 command.
axis([-2 3 -0.5 1.5]) MCL 12
This is how we obtained Figure 2, plot of our signal with a different axis scaling.
We spent a big effort to create our test signal but the remaining part is relatively easy. Let

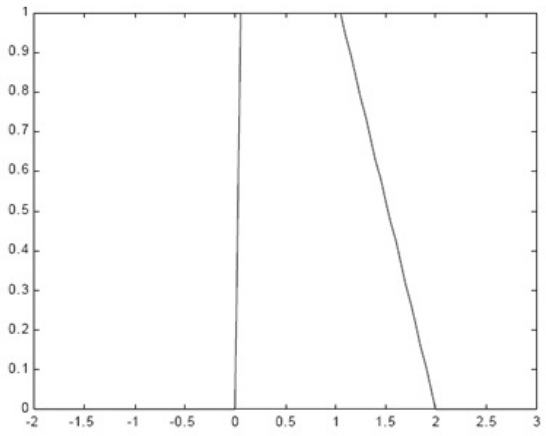


Figure 6.1: The test signal that will be used in transformation examples

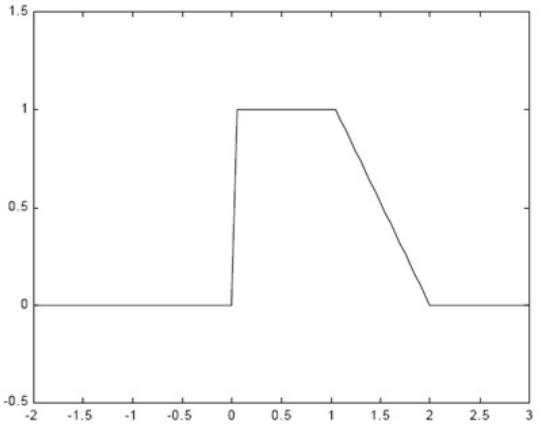


Figure 6.2: The test signal with better axes scaling

us see the effect of time shift first. For instance, let us try to find out how x(t +1) versus t will look. We know that x(t +1) should look like x(t) shifted towards left by 1 unit in time. You

may think that we simply can do this by the following line.
plot(t+1,x)
But this is not true! Look at the result of this misleading idea at Figure 3,which we produced by the following lines.
subplot(2,1,1);plot(t,x);axis([-3 4 -0.5 1.5]) MCL 13
subplot(2,1,2);plot(t+1,x);axis([-3 4 -0.5 1.5]) MCL 14

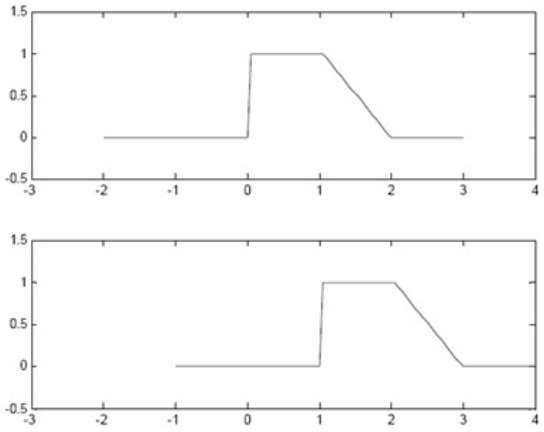


Figure 6.3: Plot of x(t) versus t (upper panel) and erroneous plot of x(t) versus t+1 (lower panel)

Above, we utilized the subplot command that enabled us to divide the figure window into several parts where we can examine different variables or plots simultaneously. To return back to the default single axes (or plot) view on the figure window, we can either issue subplot(1,1,1) or clf (clear figure) commands. Coming back to the real issue, we realize that what we did is actually the plot of x(t) versus t +1, but we were after x(t +1) versus t. How can we accomplish this, i.e. do the plot of x(t +1) versus t? Let y(t)=x(t+1) and u=t+1; then
t=u-1 and y(u-1)=x(u).
As u is just a dummy variable, we can replace it with t and obtain
y(t-1)=x(t).

This indicates that our new function y(t)=x(t+1) takes the value x(t) at t-1.
Therefore, if we plot our original function x(t) with respect to a new time variable or vector that we will form as t-1, we will get x(t +1) versus t. Let us realize this idea now.
subplot(2,1,1);plot(t,x);axis([-3 4 -0.5 1.5]) MCL 15
xlabel('t');ylabel('x(t)') MCL 16 title('x(t) versus t') MCL 17 subplot(2,1,2);plot(t-1,x);axis([-3 4 -0.5 1.5]) MCL 18
xlabel('t');ylabel('x(t+1)') MCL 19 title('x(t+1) versus t') MCL 20

We again used subplot commands to see the plots of x(t) and x(t +1) at the same time, i.e. in the same figure window. The result is shown in Figure 4 and as we expected x(t +1) is same as x(t) shifted towards left by 1 unit in time. Here, we also used xlabel and title commands to make our plots more informative.
Let us now make the plot of x(-2t) versus t to study the effects of time reversal and time scaling. We expect x(-2t) to be a time reversed and compressed version of x(t) . To verify this, let y(t)=x(-2t) and u= -2t; then

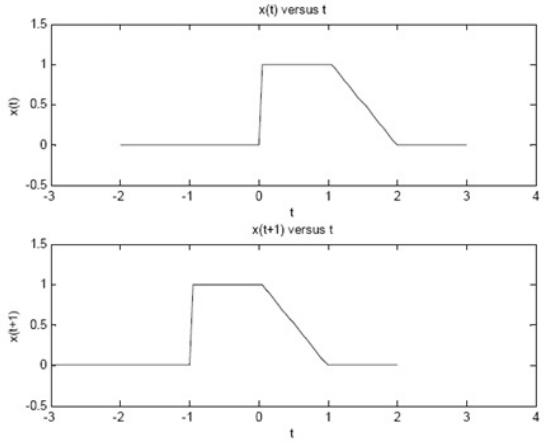


Figure 6.4: Plot of x(t) versus t (upper panel) and x(t+1) versus t (lower panel)

t= -u/ 2 or y(-u/2)=x(u).
We replace u by t and obtain y(-t/2)=x(t). Hence, y(t)=x(-2t) takes the value x(t) at -t / 2 . In other words, plot of x(-2t) versus t is equivalent to the plot of x(t) versus -t / 2 . The following lines will plot x(t) and x(-2t). The result is shown in Figure 5.

```
subplot(2,1,1);plot(t,x);axis([-3 4 -0.5 1.5]) MCL 21
xlabel('t');ylabel('x(t)') MCL 22
title('x(t) versus t') MCL 23
subplot(2,1,2);plot(-t/2,x);axis([-3 4 -0.5 1.5]) MCL 24
xlabel('t');ylabel('x(-2t)') MCL 25
title('x(-2t) versus t') MCL 26
```

As another exercise, let us plot x((t/3)-1) versus t. This example involves time scaling and

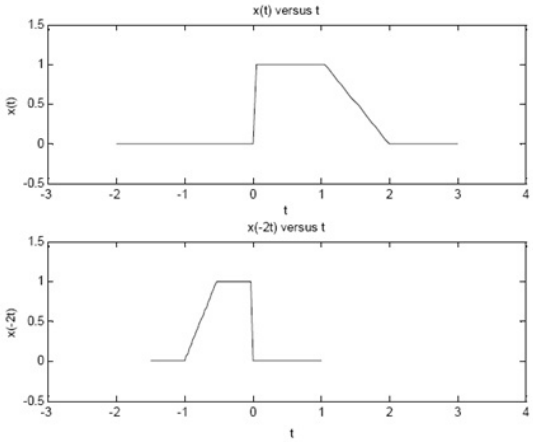


Figure 6.5: Plot of x(t) versus t (upper panel) and x(-2t) versus t (lower panel).

time shifting. We will use the same approach again, where we find the time at which the new or transformed signal y(t) takes the value of the original signal x(t) . This way, we keep the original signal the same and plot it with respect to a transformed time variable or vector, to find the transformed signal.

Let y(t)=x((t /3)-1) and u=(t/3)-1; then t=3(u+1) and y(3(u+1))=x(u) . We replace u by t and obtain y(3(t+1))=x(t) . We can now complete our task by the following lines. The result is shown in Figure and as we see x((t/3)-1) is expanded/stretched and right shifted (by 3 units in time) version of x(t) .

```
subplot(2,1,1);plot(t,x);axis([-4 14 -0.5 1.5]) MCL 27
xlabel('t');ylabel('x(t)') MCL 28
title('x(t) versus t') MCL 29
tnew=3*(t+1); MCL 30
subplot(2,1,2);plot(tnew,x);axis([-4 14 -0.5 1.5]) MCL 31
xlabel('t');ylabel('x((t/3)-1)') MCL 32
title('x((t/3)-1) versus t') MCL 33
```

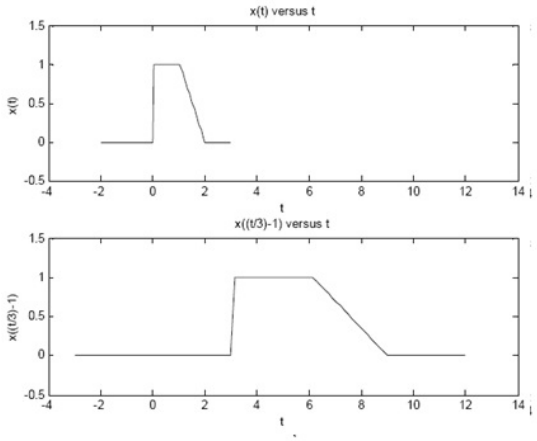


Figure 6.6: Plot of x(t) versus t (upper panel) and x((t/3)-1) versus t (lower panel).

Conclusion:

Summarize, in a paragraph or two, what you conclude from the results of your experiment and whether they are what you expected them to be. Compare the results with theoretical expectations and include percent error when appropriate. Don't use terms such as "fairly close" and "pretty good;" give explicit quantitative deviations from the expected result. Evaluate whether these deviations fall within your expected errors and state possible explanations for unusual deviations. Discuss and comment on the results and conclusions drawn, including the sources of the errors and the methods used for estimating them.

Chapter 7

LAB NO. 07

7.1 To Sketch The Even And Odd Part Of Signals In MATLAB

PERFORMANCE OBJECTIVES:
Upon successful completion of this experiment student will be able to:
Become familiar with MATLAB environment Draw the even signals Draw the odd signals

EQUIPMENT REQUIRED:
A computer running Matlab 7.0 or higher version

THEORETICAL BACKGROUND:
The signal can be divided into 2 parts i.e., evenpart and odd part. They are calculated as
Even part of signal x(t) = (x(t)+ x(-t))/2
Odd part of signal x(t) = (x(t)- x(-t))/2
PART (1)

MATLAB CODE:

```
n=-10:10;
xn=zeros(1,length(n));
xn(find(n==-2))=-1;
xn(find(n==0))=3;
xn(find(n==1))=2;
xn(find(n==2))=1;
xn(find(n==7))=1;
xnf=fliplr(xn);
xne=(xn+xnf)/2;
xno=(xn-xnf)/2;
subplot(411);stem(n,xn);
xlabel('n');ylabel('xn');
axis([-10 10 -2 4]);
subplot(412);stem(n,xnf);
xlabel('n');ylabel('xnf');
axis([-10 10 -2 4]);
subplot(413);stem(n,xne);
xlabel('n');ylabel('xne');
axis([-10 10 -2 4]);
```

```
subplot(414);stem(n,xno);
xlabel('n');ylabel('xno');
axis([-10 10 -2 4]);
```

OUTPUT GRAPH

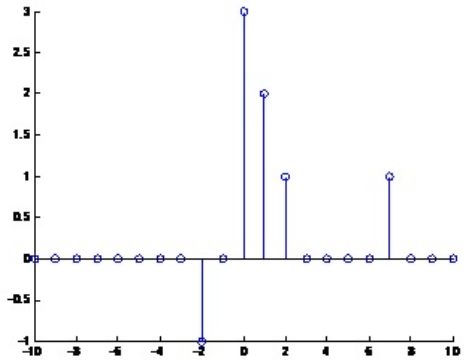


Figure 7.1: Discrete Signal.

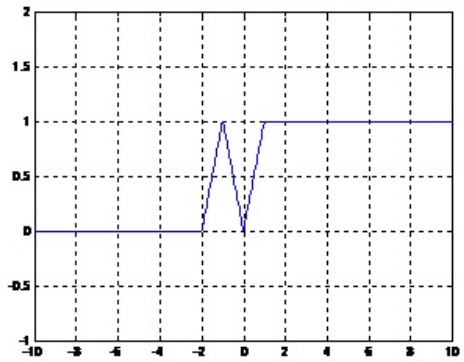


Figure 7.2: Discrete Signal, horizontal mirror,even part and odd part of the disrete signal

Part(2)

To sketch The Even And Odd Part Of The Given Signals

MATLAB CODE:

```
t1=-10:0.01:-2;
t2=-2:0.01:-1;
t3=-1:0.01:0;
t4=0:0.01:1;
t5=1:0.01:10;
t=[t1 t2 t3 t4 t5];
x1=linspace(0,0,length(t1));
x2=linspace(0,1,length(t2));
x3=linspace(1,0,length(t3));
x4=linspace(0,1,length(t4));
```

```
x5=linspace(1,1,length(t5));
x=[x1 x2 x3 x4 x5];
xhm=fliplr(x);
xe=(x+xhm)/2;
xo=(x-xhm)/2;
subplot(411);plot(t,x);
xlabel('t');ylabel('x');
axis([-10 10 -2 4]);
grid on;
subplot(412);plot(t,xhm);
xlabel('t');ylabel('xhm');
axis([-10 10 -2 4]);
grid on;
subplot(413);plot(t,xe);
xlabel('t');ylabel('xe');
axis([-10 10 -2 4]);
grid on;
subplot(414);plot(t,xo);
xlabel('t');ylabel('xo');
axis([-10 10 -2 4]);
```

PLOT :

Part(3)

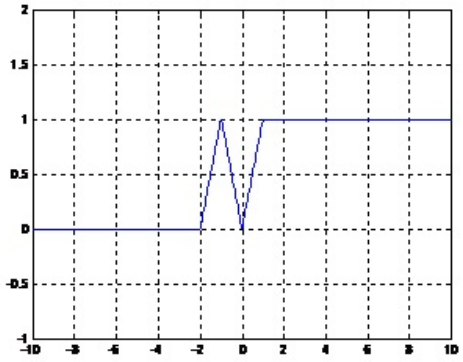


Figure 7.3: Given Continuous Signal

To sketch The Even And Odd Part Of The Given Signals

MATLAB CODE:

```
t1=-10:0.01:-2;
t2=-2:0.01:-1;
t3=-1:0.01:0;
t4=0:0.01:1;
t5=1:0.01:2;
t6=2:0.01:3;
t7=3:0.01:10;
t=[t1 t2 t3 t4 t5 t6 t7];
x1=linspace(0,0,length(t1));
x2=linspace(0,1,length(t2));
```

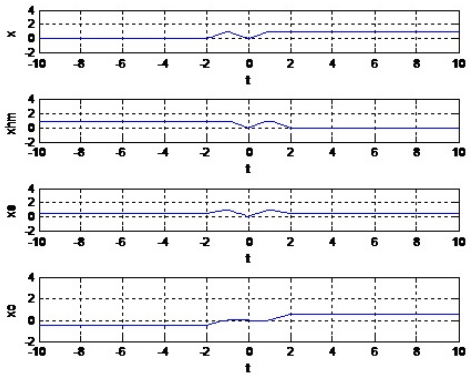


Figure 7.4: Continuous Signal, horizontal mirror,even part and odd part of the Continuous signal

```
x3=linspace(1,1,length(t3));
x4=linspace(1,2,length(t4));
x5=linspace(2,2,length(t5));
x6=linspace(2,0,length(t6));
x7=linspace(0,0,length(t7)); x=[x1 x2 x3 x4 x5 x6 x7];
xhm=fliplr(x);
xe=(x+xhm)/2;
xo=(x-xhm)/2;
subplot(411);plot(t,x);
xlabel('t');ylabel('x');
axis([-10 10 -2 4]);
grid on;
subplot(412);plot(t,xhm);
xlabel('t');ylabel('xhm');
axis([-10 10 -2 4]);
grid on;
subplot(413);plot(t,xe);
xlabel('t');ylabel('xe');
axis([-10 10 -2 4]);
grid on;
subplot(414);plot(t,xo);
xlabel('t');ylabel('xo');
axis([-10 10 -2 4]);
grid on:
```

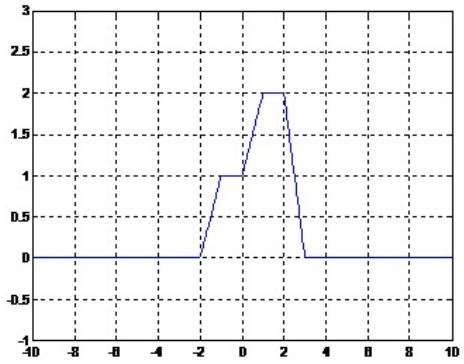


Figure 7.5: Given Continuous Signal

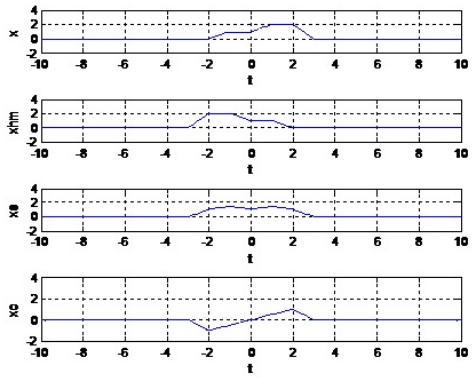


Figure 7.6: Output graph Continuous Signal, horizontal mirror,even part and odd part of the Continuous signal

Conclusion:

Summarize, in a paragraph or two, what you conclude from the results of your experiment and whether they are what you expected them to be. Compare the results with theoretical expectations and include percent error when appropriate. Don't use terms such as "fairly close" and "pretty good;" give explicit quantitative deviations from the expected result. Evaluate whether these deviations fall within your expected errors and state possible explanations for unusual deviations. Discuss and comment on the results and conclusions drawn, including the sources of the errors and the methods used for estimating them.

Chapter 8

LAB NO. 08

8.1 To observe the different exponential signals

PERFORMANCE OBJECTIVES
Upon successful completion of this experiment student will be able to:
Become familiar with MATLAB environment
Draw the even signals
Draw the odd signals

EQUIPMENT REQUIRED:
A computer running Matlab 7.0 or higher version
PART 1 : Real Exponential Signal

```
A=4;
a1=0.1;a2=0.2;a3=-0.1;a4=-0.2
wo=2;
t=-3:0.01:3;
x1=A*exp(a1*wo*t);
x2=A*exp(a2*wo*t);
x3=A*exp(a3*wo*t);
x4=A*exp(a4*wo*t);
plot(t,x1,'r-',t,x2,'k–',t,x3,'g:',t,x4,'b+');
legend('a1=0.1','a2=0.2','a3=-0.1','a4=-0.2');
```

PART 2 : Periodic Exponential Signal

```
A=5;
wo=1.5;
t=-6*pi:0.01:6*pi;
x=A*exp(j*wo*t);
plot(t,real(x),'b',t,imag(x),'g');
legend('Real part','imaginary part');
```

PART 3 : Harmonic Exponential Signal A=5;

```
wo=2;
t=-3:0.01:3;
x1=A*exp(j*wo*t);
x2=A*exp(j*2*wo*t);
x3=A*exp(j*3*wo*t);
subplot(211);
plot(t,real(x1),'r-',t,real(x2),'g–',t,real(x3),'k.');
legend('Real part(x1)','Real part(x2)','Real part(x3)');
```

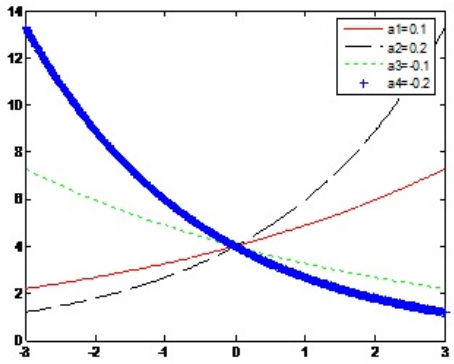


Figure 8.1: General exponential signal

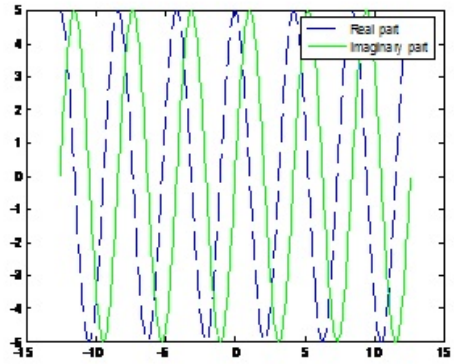


Figure 8.2: Periodic exponential signal

```
subplot(212);
plot(t,imag(x1),'r-',t,imag(x2),'g–',t,imag(x3),'k.');
legend('Imaginary part(x1)','Imaginary part(x2)','Imaginary part(x3)');
 PART 4 : General Exponential Signal
A=5; wo=2;
a=-0.01;
t=-10:0.01:10;
x=A*exp((a+j*wo)*t);
plot(t,real(x),'r-',t,imag(x),'k–');
legend('Real part(x)','Imaginary part(x)');
```

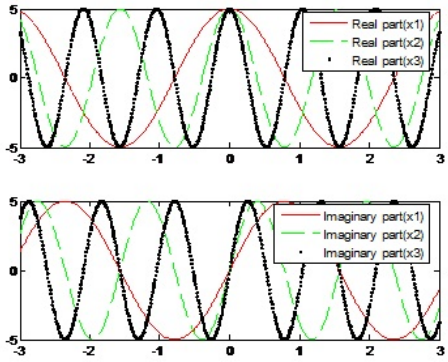


Figure 8.3: Periodic exponential signal

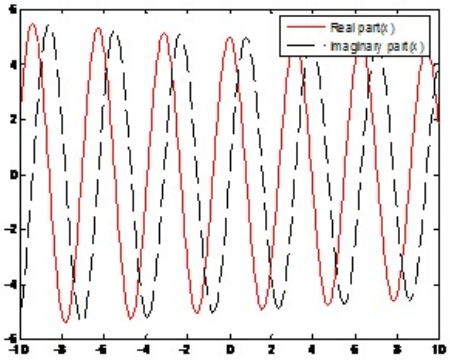


Figure 8.4: General exponential signal

Conclusion:

Summarize, in a paragraph or two, what you conclude from the results of your experiment and whether they are what you expected them to be. Compare the results with theoretical expectations and include percent error when appropriate. Don't use terms such as "fairly close" and "pretty good;" give explicit quantitative deviations from the expected result. Evaluate whether these deviations fall within your expected errors and state possible explanations for unusual deviations. Discuss and comment on the results and conclusions drawn, including the sources of the errors and the methods used for estimating them.

Chapter 9

LAB NO. 09

9.1 To Draw The Fourier Series Of Given Signals

PERFORMANCE OBJECTIVES

Upon successful completion of this experiment student will be able to:
Become familiar with MATLAB environment
Draw the even signals
Draw the odd signals

EQUIPMENT REQUIRED:

A computer running Matlab 7.0 or higher version
THEORETICAL BAKGROUND:
In mathematics, a Fourier series decomposes any periodic function or periodic signal into the sum of a (possibly infinite) set of simple oscillating functions, namely sines and cosines (or complex exponentials). The study of Fourier series is a branch of Fourier analysis. Fourier series were introduced by Joseph Fourier (1768-1830) for the purpose of solving the heat equation in a metal plate.
Fourier series are used in the analysis of periodic functions. Many of the phenomena studied in engineering and science are periodic in nature eg. the current and voltage in an alternating current circuit. These periodic functions can be analysed into their constituent components (fundamentals and harmonics) by a process called Fourier analysis.

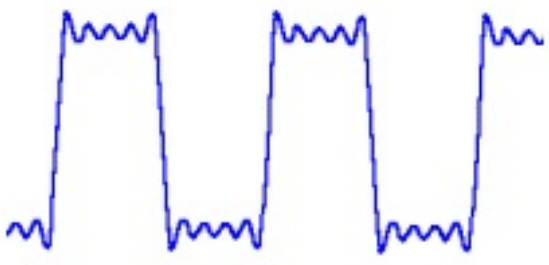


Figure 9.1: A periodic function

```
t=-4:0.001:4;
xo=1;
x1=1/4*(exp(-j*2*pi*t))+1/4*(exp(j*2*pi*t));
```

```
x2=1/2*(exp(-j*4*pi*t))+1/2*(exp(j*4*pi*t));
x3=1/3*(exp(-j*6*pi*t))+1/3*(exp(j*6*pi*t));

    subplot(421);
plot(t,xo)
title('xo');

    subplot(422);
plot(t,x1)
title('x1');

    subplot(423);
plot(t,x2)
title('x2');

    subplot(424);
plot(t,x3)
title('x3');

    subplot(425);
plot(t,(xo+x1))
title('(xo+x1)');

    subplot(426);
plot(t,(xo+x1+x2))
title('(xo+x1+x2)');

    subplot(427);
plot(t,(xo+x1+x2+x3))
title('(xo+x1+x2+x3)');
```

PLOT OF FORIER SERIES

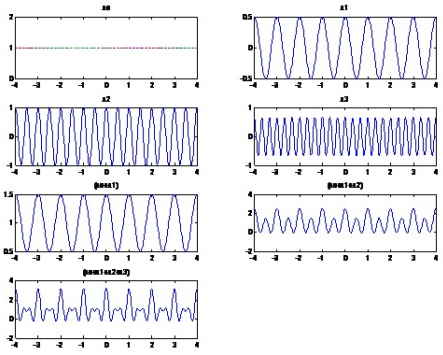


Figure 9.2: Plot of FOURIER for different values of k

```
wo=pi;
t=-3:0.001:3;
s1=0;s2=0;
```

```
for k2=-1000:1:-1;
ak2=j/(k2*pi)*(-1).k̂2;
x2=ak2.*exp(j.*k2*wo*t);
s2=s2+x2;
end;
for k1=1:1:1000;
ak1=j/(k1*pi)*(-1).k̂1;
x1=ak1.*exp(j.*k1*wo*t);
s1=s1+x1;
end;
x=s1+s2;
plot(t,x)
 t=-3:.001:3;
```

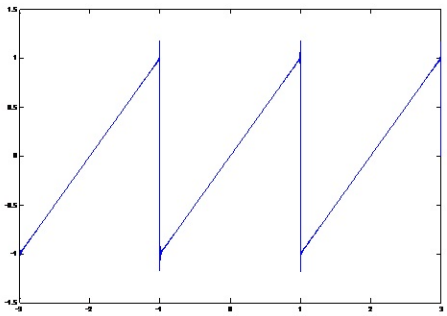


Figure 9.3: plot of Fourier of given function

```
w0=2*pi/3;a0=1;
s1=0;s2=0;
for k1=1:5000;
ak1=(j/(k1*2*pi))*(exp(-j*(2/3)*k1*pi)+exp(-j*(4/3)*k1*pi)-2);
x1=ak1.*exp(j*k1*w0*t);
s1=s1+x1;
end
for k2=-5000:-1;
ak2=(j/(k2*2*pi))*(exp(-j*(2/3)*k2*pi)+exp(-j*(4/3)*k2*pi)-2);
x2=ak2.*exp(j*k2*w0*t);
s2=s2+x2;
end
x=s1+s2+a0;
plot(t,x,'-'),grid on
title('3.22: A(f)')
xlabel('time'),ylabel('x(t) ')
 t=-3:.001:3;
w0=pi;a=exp(1)-exp(-1);a0=a/2;
s1=0;s2=0;
for k1=1:1000;
ak1=a*cos(k1*w0)/(2*(1+j*k1*w0));
x1=ak1.*exp(j*k1*w0*t);
```

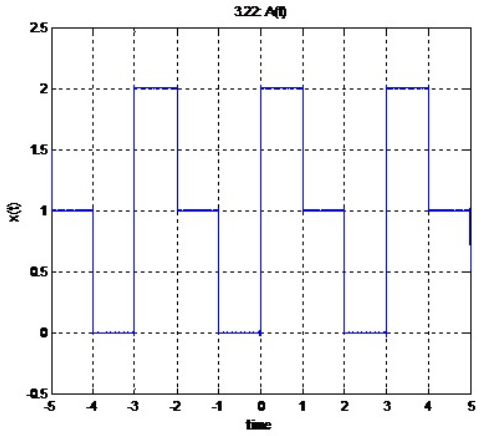


Figure 9.4: plot of Fourier of given function

```
s1=s1+x1;
end
for k2=-1000:-1;
ak2=a*cos(k2*w0)/(2*(1+j*k2*w0));
x2=ak2.*exp(j*k2*w0*t);
s2=s2+x2;
end
x=s1+s2+a0;
plot(t,x)
title('x(t)=exp(-t) for -1< t <1')
xlabel('time'),ylabel('x(t)')
```

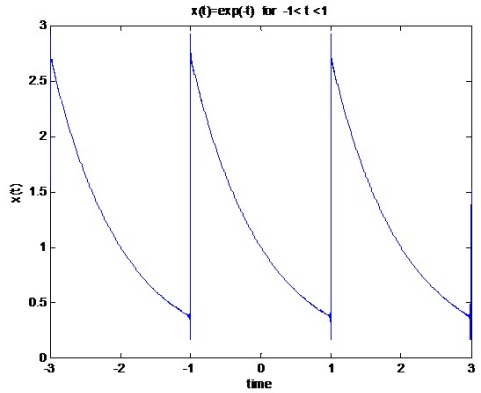


Figure 9.5: plot of Fourier of given function

Conclusion:
Summarize, in a paragraph or two, what you conclude from the results of your experiment and whether they are what you expected them to be. Compare the results with theoretical expectations and include percent error when appropriate. Don't use terms such as "fairly close" and "pretty good;" give explicit quantitative deviations from the expected result. Evaluate whether these deviations fall within your expected errors and state possible explanations for unusual deviations. Discuss and comment on the results and conclusions drawn, including the sources of the errors and the methods used for estimating them.

LAB NO. 10

10.1 To Calculate The Laplace And Fourier Of Different Signals

PERFORMANCE OBJECTIVES
Upon successful completion of this experiment student will be able to:
Calculate the Fourier of function
Calculate the Laplace of function

EQUIPMENT REQUIRED:
A computer running Matlab 7.0 or higher version

THEORETICAL BAKGROUND:
In mathematics, the Laplace transform is a widely used integral transform. Denoted L{f(t)}, it is a linear operator of a function f(t) with a real argument t (t âL'ě 0) that transforms it to a function F(s) with a complex argument s. This transformation is essentially bijective for the majority of practical uses; the respective pairs of f(t) and F(s) are matched in tables. The Laplace transform has the useful property that many relationships and operations over the originals f(t) correspond to simpler relationships and operations over the images F(s).[1] The Laplace transform has many important applications throughout the sciences. It is named for Pierre-Simon Laplace who introduced the transform in his work on probability theory.

MATLAB CODES TO CALCULATE THE FOURIER OF FUNCTION

```
» syms t a;
» f=t2̂;
» F=fourier(f)
F =
-2*pi*dirac(2,w)
» f=1/(1+t2̂)
f =
1/(1+t2̂)
» F=fourier(f)
F =
pi*(exp(w)*heaviside(-w)+exp(-w)*heaviside(w))
» syms w;
» F=1/(1+w2̂)
F =
```

1/(1+w$\hat{2}$)
» f=ifourier(F)
f =
1/2*exp(-x)*heaviside(x)+1/2*exp(x)*heaviside(-x)
» subplot(211)
» ezplot(F)
» subplot(212)
» ezplot(f)

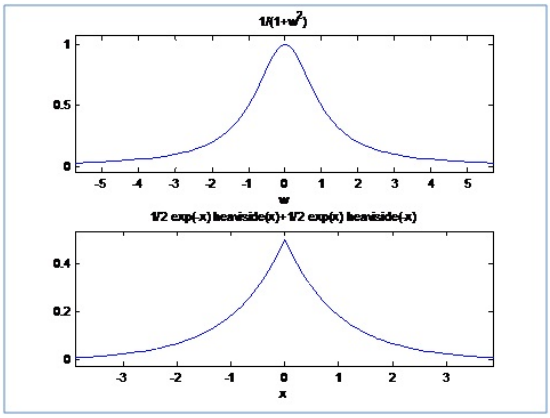


Figure 10.1: plot of Fourier of given function

» syms w
»F=2/(j*w)
F =
-2*i/w
» f=ifourier(F)
f =
2*heaviside(x)-1
» ezplot(f) MATLAB CODES TO CALCULATE THE LAPLACE OF FUNCTION »syms t

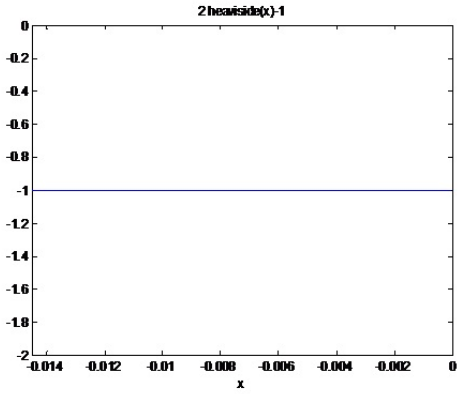


Figure 10.2: plot

» f=t;
» F=laplace(f)

F =
1/sˆ2
» subplot(211)
» ezplot(f)
» subplot(212)
» ezplot(F) » f=dirac(t);

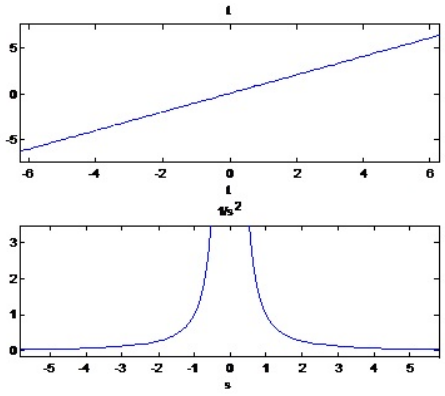


Figure 10.3: plot

» F1=fourier(f)
F1 =
1
» F2=laplace(f)
F2=
1
» f=heaviside(t);
» F=fourier(f)
F =
pi*dirac(w)-i/w
» F=laplace(f)
F =
1/s

Sr. No.	F(t)	F(jw)	F(s)
1	δ(t) =Dirac(t)	1	1
2	U(t) = heaviside(t)	pi*dirac(w)-i/w	1/s
3	t U(t) = t * heaviside(t)	i*(pi*dirac(1,w)*w^2+i)/w^2	1/s^2
4	1 / t	i*pi*(1-2*heaviside(w))	laplace(1/t,t,s)
5	t ² U(t)	-2*pi*dirac(2,w)	2/s^3

Figure 10.4: Table

Conclusion:
Summarize, in a paragraph or two, what you conclude from the results of your experiment and whether they are what you expected them to be. Compare the results with theoretical expectations and include percent error when appropriate. Don't use terms such as "fairly close" and "pretty good;" give explicit quantitative deviations from the expected result. Evaluate whether these deviations fall within your expected errors and state possible explanations for unusual deviations. Discuss and comment on the results and conclusions drawn, including the sources of the errors and the methods used for estimating them.

LAB NO. 11

11.1 To study the behavior of an Over-Damping RLC series Circuits using

THEORETICAL BACKGROUND:-
Inductors, capacitors and resistors are used. We now wish to determine the natural response of circuit model composed of ideal circuit elements connected in series.
The series RLC circuit is the dual of the parallel circuit and this single fact is sufficient to make its analysis a trivial affair. Figure show the circuit element connected in series. The equation is as follows, and we obtained the differential equation of RLC series circuit.
We get a brief resume of the Series circuit response which may be over-damped, critically

$$V_L + iR + V_C = 0$$

$$L\left(\frac{di}{dt}\right) + R \times i + \left(\frac{1}{C}\right) \int_{t0}^{t} i\,dt = 0$$

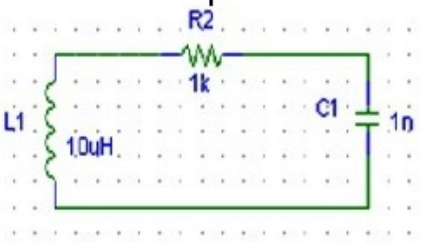


Fig 01.1: the RLC series circuit

Figure 11.1: Circuit Diagram

damped, under damped. We will use their formulas.
The over-damped response is MATLAB-CODE:

function project1

The over-damped response is

$$I(t) = A_1 e^{S_1 t} + A_2 e^{S_2 t}$$

$$S_{1,2} = -\frac{R}{2L} \pm \sqrt{(\frac{R}{2L})^2 - (\frac{1}{LC})^2}$$

And thus $\alpha = \frac{R}{2L}$, $\omega = \frac{1}{\sqrt{LC}}$

The critically-damped response is

$$I(t) = e^{-\alpha t}(A_1 t + A_2 t)$$

The under-damped response is

$$I(t) = e^{-\alpha t}(B1 \cos \omega_d t + B2 \sin \omega_d t)$$

where $\omega = \sqrt{(\omega d2 - \alpha 2)}$

In summary, we have

The parallel circuit	The series circuit
$\alpha = \frac{1}{2RC}$	$\alpha = \frac{R}{2L}$

Figure 11.2: .

CIRCUIT DIAGRAM:-

Circuit 01)

R=470 Ω, L=500 × 10^{-3} H, C=10 × 10^{-6} F

$\alpha = \frac{R}{2L} = 470 s^{-1}$ and $\omega_o = \frac{1}{\sqrt{LC}} = 44.2\ S^{-1}$

$V_L(t) = 524.9\, e^{-324.8t} - 994.9\, e^{-615.2t}$ V

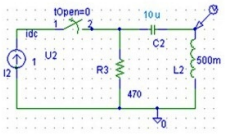


Fig 01.2: Series RLC with R=470 Ω

Circuit 02)

R=4.7 k Ω, L=500 × 10^{-3} H, C=10 × 10^{-6} F

$\alpha = \frac{R}{2L} = 4700 s^{-1}$ and $\omega_o = \frac{1}{\sqrt{LC}} = 447.2\ S^{-1}$

$V_L(t) = 10.71\, e^{-21.32t} - 4711\, e^{-9379t}$ V

over damped response with S_1 =-21.32 S^{-1} and S_2 =-9373 S^{-1} and peak inductor voltage magnitude is 4700

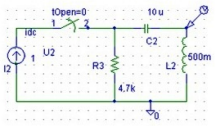


Fig 01.3: Series RLC with R=4700 Ω

Figure 11.3: .

```
    subplot(221)
t=0:0.001e-3:0.004;
y=524.9.*exp(-324.8.*t)-994.9.*exp(-615.2.*t);
plot(t,y)
xlabel('time');
ylabel('Voltage across Inductor)');
title('UNDER-DAMPED RESPONSE OF RLC SERIES CIRCUITS')
hold on

    subplot(222)
t=0:0.001e-3:0.004;
y=10.71.*exp(-21.32.*t) - 4711.*exp(-9379.*t);
plot(t,y)
xlabel('time');
ylabel('Voltage across Inductor)');
hold on
subplot(223)
t=0:0.001e-3:1e-4;
y=5.*exp(-10.*t)-470000.*exp(-939999.*t);
plot(t,y)
```

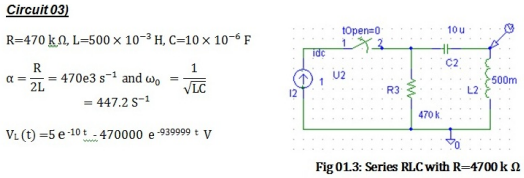


Figure 11.4: .

xlabel('time');
ylabel('Voltage across Inductor');

» project1
OUTPUT GRAPH:

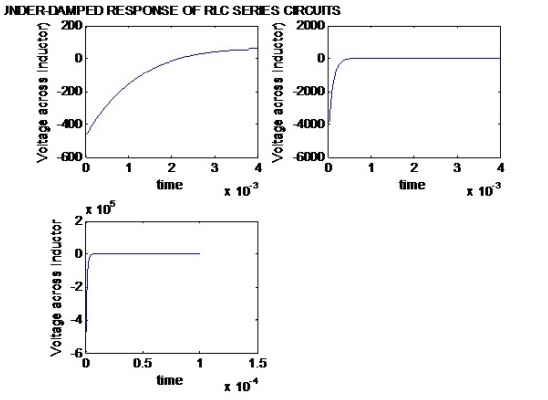


Figure 11.5: response of RLC series circuit

Conclusion:
Summarize, in a paragraph or two, what you conclude from the results of your experiment and whether they are what you expected them to be. Compare the results with theoretical expectations and include percent error when appropriate. Don't use terms such as "fairly close" and "pretty good;" give explicit quantitative deviations from the expected result. Evaluate whether these deviations fall within your expected errors and state possible explanations for unusual deviations. Discuss and comment on the results and conclusions drawn, including the sources of the errors and the methods used for estimating them.

LAB NO. 12

12.1 To study the behavior of an Over-Damping RLC series Circuits using MATLAB Control Structure

THEORETICAL BACKGROUND

Inductors, capacitors and resistors are used. We now wish to determine the natural response of circuit model composed of ideal circuit elements connected in series. The series RLC circuit is the dual of the parallel circuit and this single fact is sufficient to make its analysis a trivial affair. Figure 02 show the circuit element connected in series. The equation is as follows, and we obtained the differential equation of RLC series circuit.

We get a brief resume of the Series circuit response which may be over-damped, critically

$$V_L + iR + V_C = 0$$
$$L\left(\frac{di}{dt}\right) + R \times i + \left(\frac{1}{C}\right) \int_{t0}^{t} i\,dt = 0$$

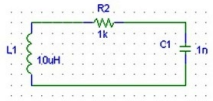


Fig 02.1: the RLC series circuit

Figure 12.1: response of RLC series circuit

damped, under damped. We will use their formulas.

In summary, we have

The over-damped response is

$$I(t) = A_1 e^{S_1 t} + A_2 e^{S_2 t}$$

$$S_{1,2} = -\frac{R}{2L} \pm \sqrt{\left(\frac{R}{2L}\right)^2 - \left(\frac{1}{LC}\right)^2}$$

And thus $\alpha = \frac{R}{2L}$, $\omega = \frac{1}{\sqrt{LC}}$

The critically-damped response is

$$I(t) = e^{-a t}(A_1 t + A_2 t)$$

The under-damped response is

$$I(t) = e^{-a t}(B_1 \cos \omega_d t + B_2 \sin \omega_d t)$$

where $\omega = \sqrt{(\omega d2 - \alpha 2)}$

Figure 12.2: response of RLC series circuit

MATLAB-CODE:

function project2

The parallel circuit	The series circuit
$\alpha = \frac{1}{2RC}$	$\alpha = \frac{R}{2L}$

CIRCUIT DIAGRAM:-

Hare there is circuit diagram of RLC series circuit. For different values of Resistance, Capacitor and Inductor given by the user we have different types of circuit response.

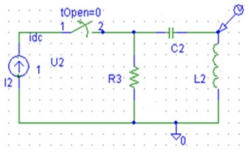


Fig 02 .2: the RLC series circuit

Figure 12.3: response of RLC series circuit

```
R=input('enter the Value of Resistance = ');
L=input('enter the Value of Inductor = ');
C=input('enter the Value of Capacitor = ');
a=R /(2*L);
w=1/((L*C)^0.5);
if (a>w)
disp(' RLC Circuit is Over-damped ');
Tmin=input(' Enter the starting time = ');
Tss=input(' Enter the Step Size on time interval = ');
Tmax=input(' Enter the ending time = ');
t=Tmin:Tss:Tmax;
s1=-a+(a^2-w^2)^0.5;
s2=-a-(a^2-w^2)^0.5;
y=10.71.*exp(s1.*t) + 22.*exp(s2.*t);
plot(t,y);
xlabel('time axis');
ylabel(' Response of Series RLC Over-damping ');
elseif (a==w)
disp(' Hare a=w');
disp(' RLC Circuit is not over-damped. It is Critical-Damped ');
else (a<w)
disp(' Hare a < w ');
disp(' RLC Circuit is not over-damped. It is Under-Damped ');
end
» project2
Enter the Value of Resistance = 470
Enter the Value of Inductor = 500e-3
Enter the Value of Capacitor = 10e-6
RLC Circuit is Over-damped
Enter the starting time = 00
Enter the Step Size on time interval = 0.0001
Enter the ending time = 10e-3
OUTPUT GRAPH:
```

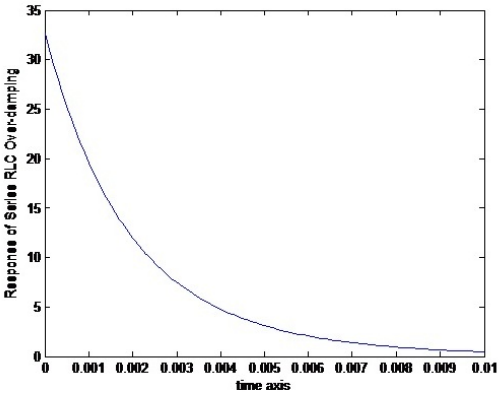


Figure 12.4: the RLC series circuit

Input values of electrical components;

Critical-damped

R=300Ω, L=22.5×10^{-3} H, C=1.0×10^{-6} F

Under-damping

R=10 Ω, L=45×10^{3} H, C=8.0 F

Figure 12.5: the RLC series circuit

Conclusion:

Summarize, in a paragraph or two, what you conclude from the results of your experiment and whether they are what you expected them to be. Compare the results with theoretical expectations and include percent error when appropriate. Don't use terms such as "fairly close" and "pretty good;" give explicit quantitative deviations from the expected result. Evaluate whether these deviations fall within your expected errors and state possible explanations for unusual deviations. Discuss and comment on the results and conclusions drawn, including the sources of the errors and the methods used for estimating them.

Chapter 13

LAB NO. 13

13.1 Simulation DC Motor Speed Control Methods Using MATLAB /Simulink

THEORETICAL BACKGROUND:-

This type of motor develops a very large amount of turning force, called torque, from a standstill. Because of this characteristic, the series dc motor can be used to operate small electric appliances, portable electric tools, cranes, winches, hoists, and the like.

The purpose of a motor speed controller is to take a signal representing the demanded speed, and to drive a motor at that speed. The controller may or may not actually measure the speed of the motor. Computer modeling and simulation tools have been extensively used to support and enhance electric machinery Figure 1-Series-wound dc motor ourses. MATLAB with its toolboxes such as Simulink and SimPowerSystems is one of the most popular software ackages used by educators and students to enhance concept about transient and steady-state characteristics of electric machines. The speed of a DC motor can be varied by controlling the field flux, the armature resistance or the terminal voltage applied to the armature circuit. The three most common speed control methods are

1. Flux Control Method
2. Variable Resistance in Series with motor

In this section, Simulink models of these two methods for DC motor drives for dynamic analysis are presented. In the method 02 Resistance control method, a series resistance is inserted in the field circuit of the motor in order to change the flux by controlling the field current. It is theoretically expected that an increase in the resistance will result in decrease in the no-load speed of the motor and in the slope of the torque-speed curve. Figure 1 shows the Simulink implementation of the field resistance control method. A DC motor block of SimPowerSystems toolbox is used. The DC motor block implements a separately excited DC motor. An access is provided to the field connections (F+, F-) so that the motor model can be used as a Series-connected. The field circuit is represented by an RL circuit (Rf and Lf in series) and is connected between the ports (F+, F-). The armature circuit consists of an inductor La and resistor Ra in series with an electromotive force EA and is connected between the ports (A+, A-). The load torque is specified by the input port TL. The electrical and mechanical parameters of the motor could be specified using its dialog box. Observe that 240 V DC source is applied to the armature and field circuits. An external resistance Rf1 is inserted in series with the field circuit to realize the Voltage resistance speed control. The output port (port m) allows for the measurement of several variables, such as rotor speed, armature and field currents, and electromechanical torque developed by the motor. Through the scope and display block, the waveform and steady-state value of the rotor speed can be easily measured

in radian per second (rad/s). In the armature voltage control method, the voltage applied

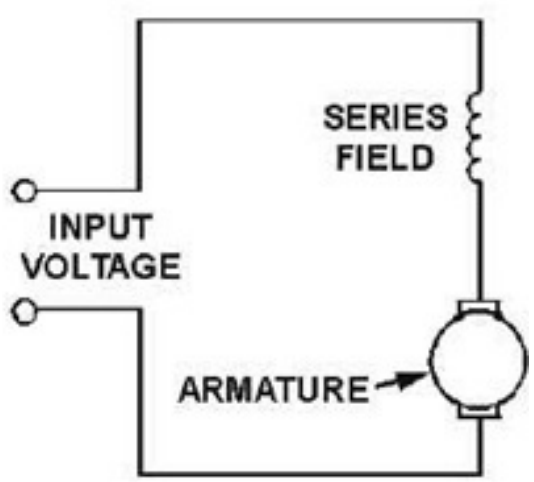


Figure 13.1: Series-wound dc motor

to the armature circuit, Va is varied without changing the voltage applied to the field circuit of the motor. Therefore, the motor must be separately excited to use armature voltage control. When the armature voltage is increased, the no-load speed of the motor increases while the slope of the torque-speed curve remains unchanged since the flux is kept constant. Figure 2 shows the Simulink realization of the armature voltage speed control method. This simulation model is similar to that of the field resistance control method shown in Figure 1. The main difference is that the armature and field circuit are supplied from two different DC sources to have a separately excited connection. Moreover, the external resistance Rf1 in Figure 2 is removed in this model.

INTRODUCTION TO SIMULINK:

In the last few years, SIMULINK has become the most widely used software package in academia and industry for modeling and simulating dynamic systems. SIMULINK is a software package for modeling, simulating, and analyzing dynamic systems. It supports linear and nonlinear systems, modeled in continuous time, sampled time, or a hybrid of the two. Systems can also be multi-rate, i.e., have different parts that are sampled or updated at different rates. SIMULINK encourages the user to try things out. User can easily build models from scratch, or take an existing model and modify it. Simulations are interactive, so user can change parameters on the spot and immediately see what happens. And because MATLAB and SIMULINK are incorporated together; user can simulate, analyze, and revise the models in either environment at any point. SIMULINK is so practical that thousands of engineers around the world are using it to model and solve real problems. Knowledge of this software will serve the user well throughout his/her professional career.

To produce a good design, there needs to be some amount of modeling or simulations done to avoid aimless trial and error techniques with the actual equipment (the DC motor). There was other specification that was required for the modeling of the DC motor and it was measured using test equipments in the laboratory and thru mathematical calculations.

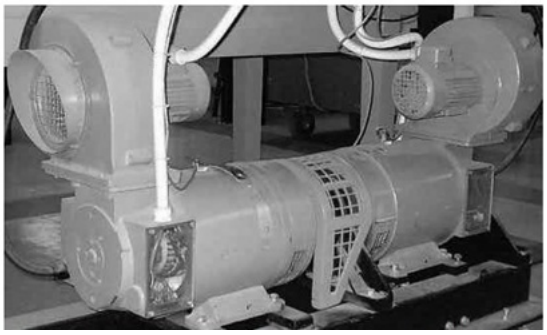


Figure 13.2: Actual DC motor in the laboratory

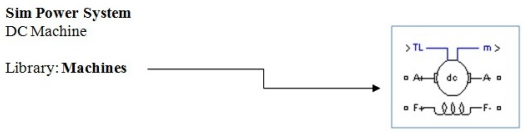


Figure 13.3: .

This block implements a separately excited DC machine. An access is provided to the field terminals (F+, F-) so that the machine model can be used as a shunt-connected or a series-connected DC machine. The torque applied to the shaft is provided at the Simulink input TL. The armature circuit (A+, A-) consists of an inductor La and resistor Ra in series with a counter-electromotive force (CEMF) E. The armature circuit is connected between the A+ and A- ports of the DC Machine block. It is represented by a series Ra La branch in series with a Controlled Voltage Source and a Current Measurement block.
The mechanical part is represented by Simulink blocks that implement the equation
Where Te is electromechanical torque, J = inertia, Bm = viscous friction coefficient, and Tf

$$J\frac{d\omega}{dt} = T_e - \ sgn(\omega) - B_m\omega - \ T_f$$

Figure 13.4: Equation

= Coulomb friction torque.The field circuit is represented by an RL circuit. It is connected between the F+ and F- ports of the DC Machine block. The mechanical part computes the speed of the DC machine from the net torque applied to the rotor. The speed is used to implement the CEMF voltage E of the armature circuit.
On double click on Machine we get the Dialog Box and Parameters. Figure shows the Implementation of Variable Resistance in Series method. Electromechanical torque
The Powergui block provides useful graphical user interface (GUI) tools for the analysis of SimPowerSystems models. Copy the Powergui block into the top level of your model and double-click the block to open the interface. What the Powergui Block Does the Powergui block allows you to choose one of three methods to solve your circuit: Continuous method, which uses a variable step solver from Simulink Discretization of the electrical system for a solution at fixed time steps Phasor solution method

Figure 13.5: Equation

IMMPLEMENTATION ON SIMULINK:
Simulation and Results

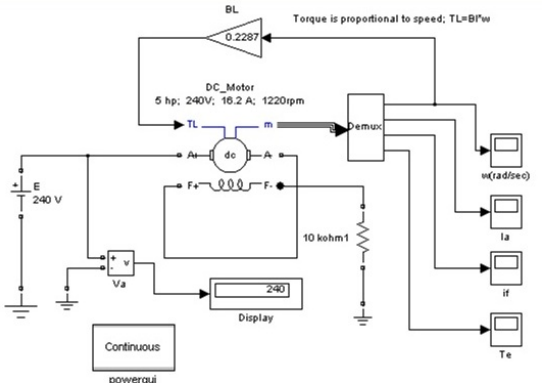
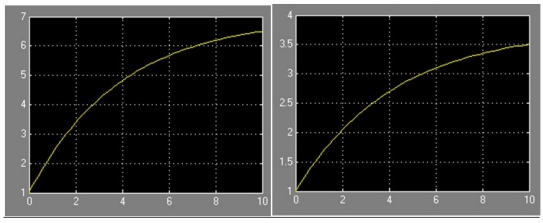


Figure 13.6: Simulink implementation of Variable Resistance in Series method



Scope result of angular speed ω ($\frac{rad}{sec}$) in DC motor (1) R= 2 , (2) R=100 Ω

Figure 13.7: Simulink implementation of Variable Resistance in Series method

Conclusion:
Summarize, in a paragraph or two, what you conclude from the results of your experiment and whether they are what you expected them to be. Compare the results with theoretical expectations and include percent error when appropriate. Don't use terms such as "fairly close" and "pretty good;" give explicit quantitative deviations from the expected result. Evaluate whether these deviations fall within your expected errors and state possible explanations for unusual deviations. Discuss and comment on the results and conclusions drawn, including the sources of the errors and the methods used for estimating them.

Chapter 14

LAB NO. 14

14.1 To study z-transform practically using MATLAB

THEORETICAL BACKGROUND:- In mathematics and signal processing, the Z-transform converts a discrete time-domain signal, which is a sequence of real or complex numbers, into a complex frequency-domain representation. The Z-transform, like many other integral transforms, can be defined as either a one-sided or two-sided transform. Bilateral Z-transform
The bilateral or two-sided Z-transform of a discrete-time signal x[n] is the function X(z) defined as: Unilateral Z-transform

$$X(z) = \mathcal{Z}\{x[n]\} = \sum_{n=-\infty}^{\infty} x[n]z^{-n}$$

Figure 14.1: Equation

Alternatively, in cases where x[n] is defined only for n âL'ě 0, the single-sided or unilateral Z-transform is defined as: The z-transform is the discrete-time counter-part of the Laplace

$$X(z) = \mathcal{Z}\{x[n]\} = \sum_{n=0}^{\infty} x[n]z^{-n}$$

Figure 14.2: Equation

transform and a generalization of the Fourier transform of a sampled signal. The z-transform allows insight into the transient behavior, the steady state behavior, and the stability of discrete-time systems. A working knowledge of the z-transform is essential to the study of digital filters and systems Frequency Response:
The Freqz function computes and display the frequency response of given Z- Transform of the function
freqz(b,a,npt,Fs)
b= Coeff. Of Numerator; a= Coeff. Of Denominator; Fs= Sampling Frequency Npt= no. of free points between and Fs/2

MATLAB Code

```
syms z n
ans=ztrans(1/4^n)
return
ans =
4*z/ (4*z-1)
```

Inverse Z-Transform

$X(n) = Z^{-1}[X(Z)]$

$X(Z) = 3*Z / (Z+1)$

MATLAB Code

```
syms Z n
ans=iztrans(3*Z/(Z+1))
return
ans =3*(-1) ^n
```

Stability of a system

A system is said to be stable if the poles lie inside of a unit circle on the z-plane.

Pole Zero Diagrams for a Function in Z Domain

Z plane command computes and display the pole-zero diagram of Z function.

zplane(b,a)

To display the pole value, use **root(a)**

To display the zero value, use **root(b)**

$X(Z) = [Z^{-2} + Z^{-4}] / [1-2Z^{-1}+3Z^{-2}]$

MATLAB Code

```
b= [0 1 1]
a= [1 -2 +3]
roots(a) = 1.0000 + 1.4142i
           1.0000 - 1.4142i
roots(b) = -1
zplane(b,a)
```

Figure 14.3: Code

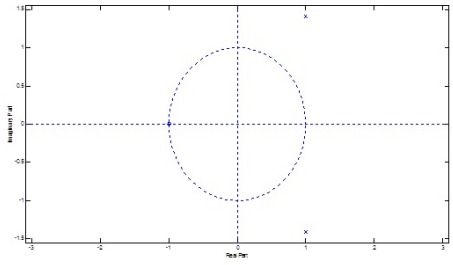


Figure 14.4: Graph

$X(Z) = [2+ 5Z^{-1}+9Z^{-2}+5Z^{-3}+3Z^{-4}] / [5+ 45Z^{-1}+2Z^{-2}+Z^{-3}+Z^{-4}]$

MATLAB Code

```
b= [2 5 9 5 3]
a= [5 45 2 1 1]
freqz(b,a)
```

Figure 14.5: Code

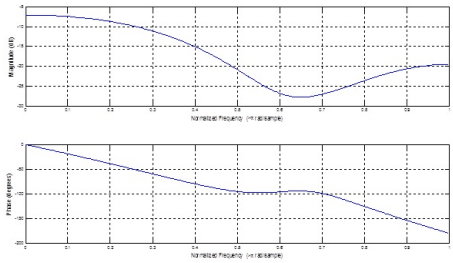


Figure 14.6: Graph