EL-223 SIGNALS & SYSTEMS

LAB MANUAL

DEPARTMENT OF ELECTRICAL ENGINEERING,

FAST-NU, LAHORE

Created by: Ms. Bushra Rashid & Ms. Beenish Fatima

Date: 1st December, 2013

Last Updated by: Sara Kiran

Date: January, 2021

Approved by the HOD: Dr. Usman Shahid

Date: January, 2021

Table of Contents

List of Equipment

Sr. No.	Description
1	PC
2	MATLAB

MOTIVATION

The purpose of the lab sessions in signals and systems is to supplement and complement the learning of the theoretical aspects of the course. The course deals with the representation and mathematical modeling of signals and the same is done in the lab using the sophisticated user friendly MATLAB environment.

MATLAB is a high level programming environment which is easy to use and interpret for the humans but slow down the processing of the information within the computer systems. When faster simulation and expedited results are required, C codes are preferred to MATLAB programs.
MATLAB also has the facility of predefined functions and simplifies the computation of Fourier and Laplace Transforms. For really skillful programmers, the challenge is to write their own .m files that are capable of computing the necessary transforms.

The lab sessions of Signals and Systems attempts to provide students the confidence to utilize MATLAB for synthesizing signals, generating transforms and capturing figures to elucidate the conceptual difficulties in depicting signals and analyzing them.

EXPERIMENT # 1

INTRODUCTION TO MATLAB

OBJECTIVE

- Learn to use MATLAB as a calculator
- Get familiar with different functions of MATLAB

1.1 Introduction to MATLAB

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.

MATLAB v.7 interface is shown in Figure 1.1:

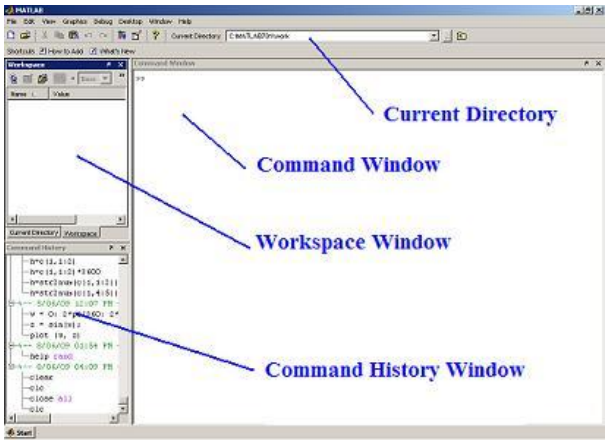


Figure 1.1: MATLAB v.7 interface

MATLAB v.7 interface is shown in Figure 1.2:

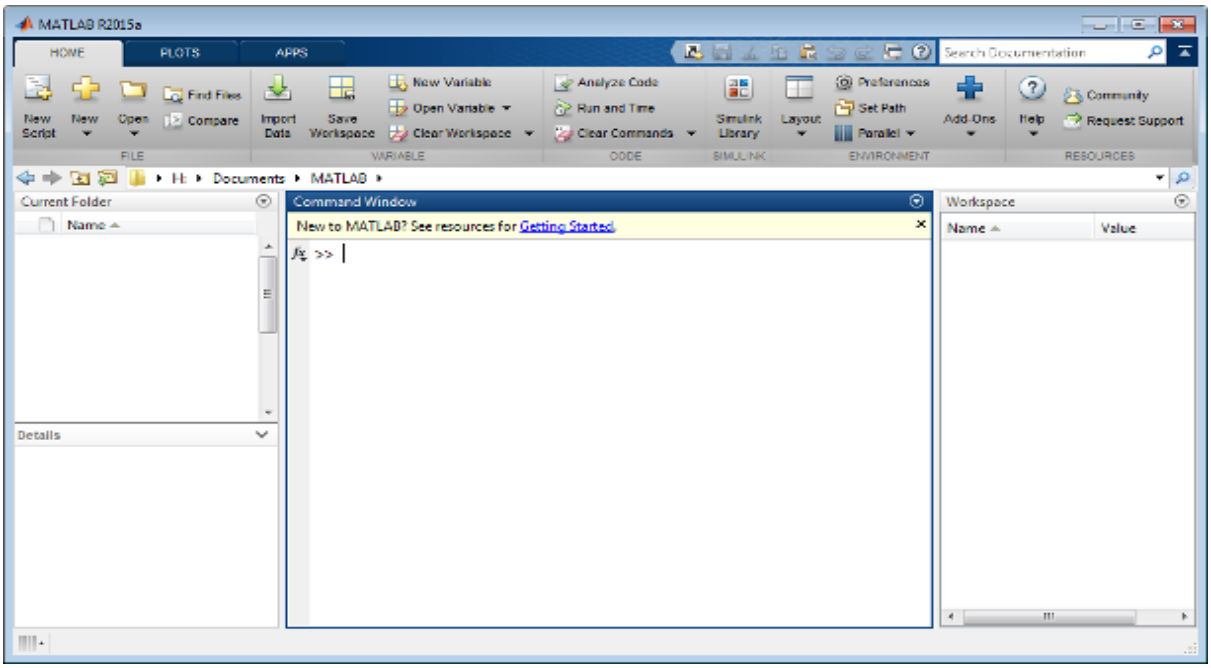


Figure 1.2: MATLAB v.7 interface

MATLAB Command Editor:

Because of Command line nature of MATLAB, editing and re-execution of previous commands is somewhat tricky. You can use shortcut arrow keys to repeat/edit a previous command or input data. Alternatively you can use command history window to call previous commands. However it is convenient for experienced computer language users to run commands in a batch-file mode. MATLAB provides command editor shown in Figure 1.3 for this purpose which is all like a language editor window.

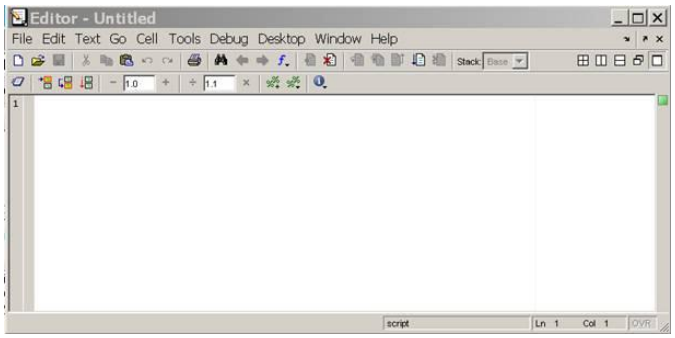


Figure 1.3: MATLAB command editor

All commands in command editor can be executed by pressing **F5** or by selecting **debug->Run** from menu. The set of commands can be saved with **.m** extension and can be executed later. Just follow the guideline for naming an m-file.

1.2 MATLAB as a Calculator

The basic arithmetic operators are **+ - * / ^** and these are used in conjunction with brackets: ( ). The symbol ^ is used to get exponents (powers):2^4=16.
You should type in commands shown following the prompt: >>.

```
>> 2 + 3/4*5
```

Is this calculation 2 + 3/(4*5) or 2 + (3/4)*5?
MATLAB works according to the priorities:

1. Quantities in brackets,
2. Powers 2 + 3^2 =>2 + 9 = 11,
3. * /, working left to right (3*4/5=12/5),
4. + -, working left to right (3+4-5=7-5),

Thus, the earlier calculation was for 2 + (3/4)*5 by priority 3.

1.3Numbers & Formats

MATLAB recognizes several different kinds of numbers.

Type	**Examples**
Integer	1362, 217897
Real	1.234, 10.76
Complex	3.21+4.3i (i =sqrt(-1))
Inf	Infinity (result of dividing by 0)
NaN	**N**ot **a N**umber, 0/0

The "**e**" notation is used for very large or very small numbers:
-1.3412e+03= $-1.3412 \times 10^{3} = -1341.2$
-1.3412e-01= $-1.3412 \times 10^{-1} = -0.13412$

All computations in MATLAB are done in double precision, which means about 15 significant figures. The format—how MATLAB prints numbers—is controlled by the "format" command. Type"helpformat"for full list.

1.4Variables

```
>> 3-2^4
>>ans*5
```

The result of the first calculation is labeled "ans" by MATLAB and is used in the second calculation where its value is changed. We can use our own names to store numbers:

```
>> x =3-2^4
```

```
>> y =x*5
```

Each variable must be assigned a value before it may be used on the right of an assignment statement.

1.4.1 Variable Names

Legal names consist of any combination of letters and digits, starting with a letter.

These are allowable:
NetCost, Left2Pay, x3, X3, z25c5
These are **not** allowable:
Net-Cost, 2pay, %x, @sign
Use names that reflect the values they represent.

Special names: you should avoid using eps =2.2204e-16 (The largest number such that 1 + eps is indistinguishable from 1) and pi =3.14159... = π.
If you wish to do arithmetic with complex numbers, both i and j have the value √-1 unless you change them.

```
>>i, j
>>i=3
>>i=j
```

1.5 General Information

MATLAB is case sensitive so 'a' and 'A' are two different names. Comment statements are preceded by a '%'.

1.6 Suppressing output

We often don't want to see the result of intermediate calculations---terminate assignment statement or expression with semi–colon (;)

```
>> x=-13;
```

The value of x is hidden.

```
y =5*x, z =x^2+y
```

Note also that we write different statements on one line, separated by commas. The above command will give two different variables when executed.

1.7Built–In Functions

1.7.1 Trigonometric Functions

Those known to MATLAB are sin, cos, tan and their arguments should be in radians. E.g. to work out the coordinates of a point on a circle of radius 5 centered at the origin and having an elevation $30^o = \pi/6$ radians:

```
>> x =5*cos (pi/6)

>> y = 5*sin (pi/6)
```

The inverse trig functions are called asin, acos, atan (as opposed to the usual arcsin or $\sin^{-1}$ etc.) The result is in radians.

```
>>acos(x/5)

>>asin(y/5)
```

cosd is cosine of argument in degrees. Y = cosd(X) is the cosine of the elements of X, expressed in degrees. For odd integers n, cosd(n*90) is exactly zero, whereas cos(n*pi/2) reflects the accuracy of the floating point value of pi.

```
>>cosd(90)
```

1.7.2 Other Elementary Functions

These include sqrt, exp, log, log10, abs, angle.

```
>> x =9;
>>sqrt(x),

>>exp(x),

>>log(sqrt(x))

>>log10(x^2+6)
```

exp(x) denotes the exponential function exp(x) $=e^x$and its inverse function is log (natural logarithm 'ln' in MATLAB).

1.8COMPLEX NUMBERS

The complex numbers consists of two separate parts: real part and an imaginary part. The basic imaginary unit is equal to the square root of -1. This is represented in MATLAB by either of two letters: i or j.
The following statement shows one way of creating a complex value in MATLAB. The variable z is assigned a complex number with a real part of 3 and an imaginary part of 4:

```
>>z=3+4j;
```

1.8.1 Operations on Complex Numbers

MATLAB provides different functions for operations on complex numbers. Some of them are listed here.

```
>>zr=real(z)

>>zi=imag(z)

>>abs(z)

>>angle(z)
```

Note that the angle returned by the angle(z) is in radians. The angle can be converted from radians to degrees using MATLAB's inbuilt function rad2deg().

```
>>conj(z)
```

1.9VECTORS

These come in two forms:**row vectors** and **column vectors**. We shall first describe the **row vectors**.

1.9.1Row Vectors

They are lists of numbers separated by either commas or spaces. The number of entries is known as the "length" of the vector and the entries are often referred to as "elements" or "components" of the vector. The entries must be enclosed in square brackets.

```
>> v = [1 3 sqrt(5)]

>>length(v)
```

Spaces can be vitally important:

```
>> v2 = [3+ 4 5]

>> v3 = [3   +4 5]
```

We can do certain arithmetic operations with vectors of the same length, such as v and v3 in the previous section.

>> v + v3

>> v4 =3*v

>> v5 =2*v -3*v3

A vector may be multiplied by a scalar (a number—see v4 above), or added/subtracted to another vector of the **same** length. The operations are carried out element wise.
We can build row vectors from existing ones:

>> w =[1 2 3], z = [8 9]
>>cd =[2*z,-w],
sort(cd)

Notice the last command sorted the elements of vector 'cd' into ascending order. We can also change the value of particular entries as shown below:

>>w(2)

>>w(2)=4

1.9.2 The Colon Notation

This is a shortcut for producing row vectors:
>> 1:4

>> 3:7

>> 1:-1

More generally a:b:cproduces a vector of entries starting with the value a, incrementing by the value b until it gets to c (it will not produce a value beyond c). This is why 1:-1 produced the empty vector [ ].

1.9.3 Extracting Bits of a Vector

>> r5 =[1:2:6, -1:-2:-7]
r5 =1 3 5 -1 -3 -5 -7
To get the 3rd to 6th entries:
>>r5(3:6)

To get alternate entries:

>>r5(1:2:7)

What does r5(6:-2:1) give?
See help colon for a fuller description.

1.9.4Column Vectors

These have similar constructs to row vectors. When defining them, entries are separated by a**';'** or '**newline/ enter**'.

>> c = [1; 3; sqrt(5)]

>> c2 =[3;4;5]

>> c3 =2*c - 3*c2

So column vectors may be added or subtracted provided that they have the same length.

1.9.5Transposing

We can convert a row vector into a column vector(and vice versa) by a process called transposing—denoted by **'**.

>> w, w', c, c'

If x is a complex vector, then x' gives the complex conjugate transpose of x:
>> x = [1+3i, 2-2i]
>> x'

To obtain the plain transpose of a complex number use **.'** as in
>> x**.'**

Exercise 1.1
Find the absolute value and phase of y = 2 cos (e^{1-4j})

Exercise 1.2
If $y = 5x^3 + 2x^2 + 9x + 16$, find y for x=2.

Exercise 1.3
If v= [2+ 3 4], v1= [2 +3 4]
Is v+ v1 possible? If not then mention the reason.

POST LAB QUESTIONS:

Q1. Solve the following expression by showing each step and storing in different variables.

7 + (-8) / 6 * (4-7/9)^(4-5/9*6)

Q2. In the following vector, print the data that is present on even indices.

V = [1:100]

Q3. Take two complex numbers and add, subtract, multiply and divide them.

EXPERIMENT # 2

VECTOR OPERATIONS AND MATRICES

OBJECTIVE

- Learn to plot elementary functions
- Implement arithmetic functions on vectors and matrices and observe their outputs

2.1 PLOTING OF ELEMENTARY FUNCTIONS

Suppose we wish to plot a graph of y = sin3πx for$0 \leq x \leq 1$. We do this by sampling the function at a sufficiently large number of points and then joining up the points (x, y) by straight lines. Suppose we take N+1 points equally spaced at distance h apart:

>> N =10; h =1/N; x =0: h: 1; defines the set of points x = 0, h, 2h, . . .,9h, 1.The corresponding y values are computed by

>> y =sin(3*pi*x); and finally, we can plot the points with

>>plot(x, y)

The result is shown in Figure 2.1, where it is clear that the value of N is too small.

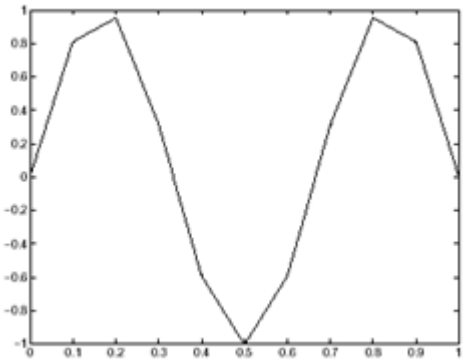


Figure 2.1: Graph of $y = \sin(3\pi x)$ for $0 \leq x \leq 1$ using h = 0.1

On changing the value of N to 100 the result is shown in Figure 2.2

>> N =100; h =1/N; x =0: h:1;

>> y =sin(3*pi*x); plot(x,y)

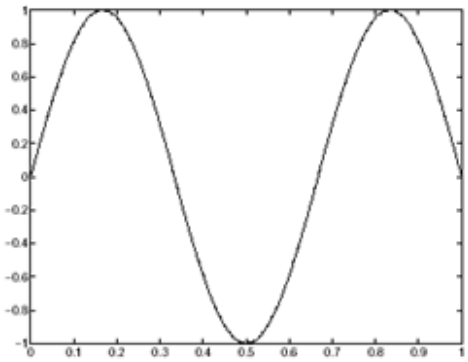


Figure 2.2: Graph of $y = \sin(3\pi x)$ for $0 \leq x \leq 1$ using h = 0.01

2.1.1 Plotting—Titles & Labels

To put a title and label the axes, we use
>>title('Graph of y =sin(3pi x)')
>>xlabel('x axis')
>>ylabel('y-axis')
The strings enclosed in single quotes, can be anything.

2.1.2 Grids

A dotted grid may be added by
>>grid
This can be removed using either **grid** again, or **grid off**.

2.1.3 Line Styles & Colors

The default is to plot solid lines. A solid black line is produced by:
>>plot(x, y, 'k-')

The third argument is a string whose first character specifies the color (optional) and the second the line style. The options for colors and styles are given in Table 2.1:

Colours		Line Styles	
y	yellow	.	point
m	magenta	o	circle
c	cyan	x	x-mark
r	red	+	plus
g	green	-	solid
b	blue	*	star
w	white	:	dotted
k	black	-.	dashdot
		--	dashed

Table 2.1: options for colors and styles

2.1.4 Multi–plots

Several graphs can be drawn on the same figure as:

>>plot(x,y, 'k-',x,cos(2*pi*x), 'g--')

A descriptive legend may be included with:

>>legend('Sin curve', 'Cos curve')

Whichwill give a list of line–styles, as they appeared in the plot command, followed by a brief description.

MATLAB fits the legend in a suitable position, so as not to conceal the graphs whenever possible. For further information do help plot etc.The result of the commands is shown in Figure 2.3:

>>plot(x,y, 'k-', x, cos(2*pi*x), 'g--')
>>legend('Sin curve', 'Cos curve')
>>title('Multi-plot ')
>>xlabel('x axis'), ylabel('y axis')
>>grid

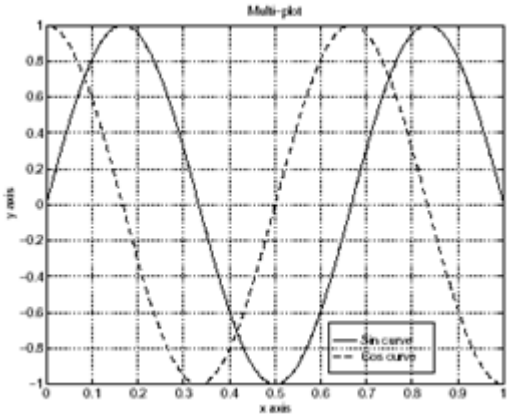


Figure 2.3: Graph of $y = \sin(3\pi x)$ and $y = \cos(3\pi x)$for $0 \leq x \leq 1$ using h = 0.01

2.1.5 Hold

A call to plot clears the graphics window before plotting the current graph. This is not convenient if we wish to add further graphics to the figure at some later stage. To stop the window being cleared:

>>plot(x,y,'k-'), hold
>>plot(x,y,'gx'), hold off

"hold on" holds the current picture; "hold off" releases it (but does not clear the window, which can be done with clf). "hold" on its own toggles the hold state.

2.1.6 Subplot

The graphics window may be split into an m × narray of smaller windows into which we may plotone or more graphs. The windows are counted 1 to mnrow–wise, starting from the top left. Bothhold and grid work on the current subplot.

```
>>subplot(221), plot(x,y)
>>xlabel('x'),ylabel('sin 3 pi x')
>>subplot(222), plot(x,cos(3*pi*x))
>>xlabel('x'),ylabel('cos 3 pi x')
>>subplot(223), plot(x,sin(6*pi*x))
>>xlabel('x'),ylabel('sin 6 pi x')
>>subplot(224), plot(x,cos(6*pi*x))
>>xlabel('x'),ylabel('cos 6 pi x')
```

subplot(221) (or subplot(2,2,1)) specifies that the window should be split into a 2 × 2 array and we select the first sub-window. The result of the commands is shown in Figure 2.4.

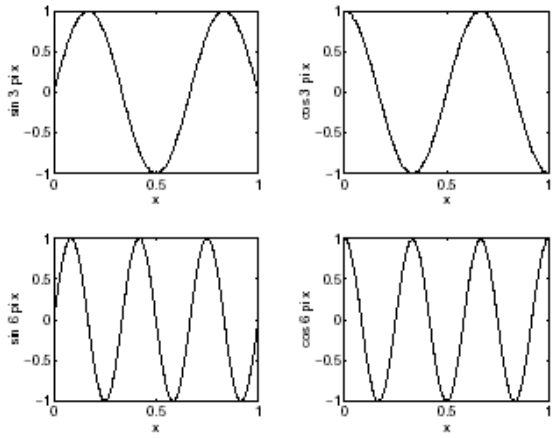


Figure 2.4: Result of the commands

2.1.7 Zooming

We often need to "zoom in" on some portion of a plot in order to see more detail. This is easily achieved using the command:

```
>> zoom
```

Left-Click on plot = Zoom in
Right-Click on plot = Zoom out
Selecting a specific portion of the plot using the mouse will zoom in to the selected portion of the plot.

2.1.8 Controlling Axes

Once a plot has been created in the graphics window you may wish to change the range of x and y values shown on the picture.

>>clf, N =100; h =1/N; x =0:h:1;
>> y =sin(3*pi*x); plot(x,y)
>>axis([-0.5 1.5 -1.2 1.2]), grid

The axis command has four parameters, the first two are the minimum and maximum values of x to use on the axis and the last two are the minimum and maximum values of y.
Note the square brackets. The result of these commands is shown in Figure 2.5. For more info, check out help axis.

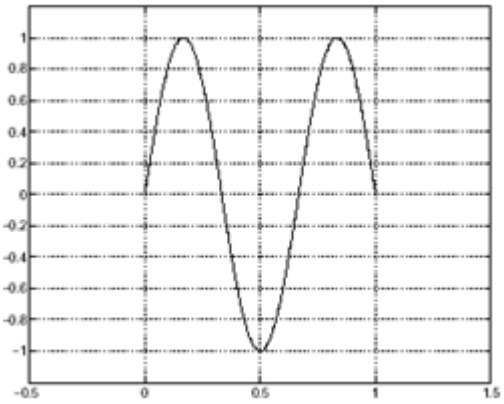


Figure 2.5: Result of these commands

Exercise 2.1.1: Define and plot the following:

$$x(t) = 2 \sin(2\pi t - \pi/2) \text{ for } 0 \le t \le 4$$

Exercise 2.1.2: Plot graphs of the functions
y = cos(x)
y = x

for$0 \le x \le 2$ in the same window. Use the zoom facility to determine the point of intersection of the two curves (and, hence, the root of x = cos(x)) to two significant figures.

2.2 PRODUCTS, DIVISION & POWER OF VECTORS

2.2.1 Scalar Product (*)

We shall describe two ways in which a meaning may be attributed to the product of two vectors. In both cases the vectors concerned must have the same length.
The first product is the standard scalar product. Suppose that u and v are two vectors of length n, **u**being a **row** vector and **v**a **column** vector:

$$\underline{u} = [u_1, \quad u_2, \quad u_3, \quad \dots, \quad u_n]$$

$$\underline{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_n \end{bmatrix}.$$

The scalar product is defined by multiplying the corresponding elements together and adding the results to give a single number (scalar).

$$\underline{uv} = \sum_{i=1}^{n} u_i v_i$$

For example, if $\underline{u} = [10 \quad -11 \quad 12]$, and $\underline{v} = \begin{bmatrix} 20 \\ -21 \\ -22 \end{bmatrix}$

Then $n = 3$ and

$\underline{uv} = 10 \times 20 + (-11) \times (-21) + (-12) \times 22 = 167.$
We can perform this product in MATLAB by

>> u =[10, -11, 12], v =[20; -21; -22]
>> prod =u*v % row times column vector

Suppose we also define a row vector **w** and a column vector **z** by
>> 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.

An error results because w is not a column vector.
Recall that transposing (with **'**) turns column vectors into row vectors and vice versa. So, to form the scalar product of two row vectors or two column vectors,

>> u*w' % u & w are row vectors

>> u*u' % u is a row vector

>> v'*z % v & z are column vectors

2.2.2Dot Product (.*)

The second way of forming the product of two vectors of the same length is known as the Dot product. It is not often used in Mathematics but frequently used in Signals & Systems. It involves vectors of the same type. If u and v are two vectors of the same type (both row vectors and both column vectors), the mathematical definition of this product, is the **vector** having the components:

$$\underline{u} \,.\, \underline{v} = [u_1 v_1, \; u_2 v_2, \quad \ldots, \quad u_n v_n]$$

.
The result is a vector of the same length and type as u and v. Thus, we simply multiply the

corresponding elements of two vectors. In MATLAB, the product is computed with the operator **.*** and, using the vectors u, v, w, z defined previously,
>>**u.*w**

>>**u.*v'**

>>**v.*z, u'.*v**

Example 2.1 *Tabulate the function y = x sin(πx)for x = 0, 0.25, . . .,1.*
It is easier to deal with column vectors so we first define a vector of x-values: (see Transposing)
>> **x =(0:0.25:1)';**

To evaluate y we have to multiply each element of the vector x by the corresponding element of the vector sin πx:
x× sin πx = x sin πx
0 × 0 = 0
0.2500 × 0.7071 = 0.1768
0.5000 × 1.0000 = 0.5000
0.7500 × 0.7071 = 0.5303
1.0000 × 0.0000 = 0.0000
To carry this out in MATLAB:

>> **y =x.*sin(pi*x)**

Note: **a)** the use of pi, **b)** x and sin(pi*x) are both column vectors (the sin function is applied to each element of the vector). Thus, the dot product of these is also a column vector.

2.2.3Dot Division of Arrays (./)

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 division—it is therefore only defined for vectors of the same size and type.
>>**a =1:5, b =6:10, a./b**

>>**a./a**

>>**c =-2:2, a./c**

The previous calculation required division by 0 --- notice the **Inf**, denoting infinity, in the answer.

>>**a.*b -24, ans./c**

2.2.4Dot Power of Arrays (.^)

To square each of the elements of a vector we could do **u.*u**. However, a neater way is to use the '**.^**' operator:

>>**u.^2**

>>**u.*u**

>>**u.^4**

>>**v.^2**

>> **u.*w.^(-2)**

Recall that powers (**.^** in this case) are done first, before any other arithmetic operation.

Exercise 2.2.1:

If x=[4 3 7 8 5 6 0] and y=[9 8 7 4 5 6 7], then find (x^2) * (y^3)

2.3 MATRICES--- 2-DArrays

Row and Column vectors are special cases of **matrices**. An m×n matrix is a rectangular array of numbers having m rows and n columns. It is usual in a mathematical setting to include the matrix in either round or square brackets—we shall use square ones. For example, when m =2, n= 3 we have a 2 × 3 matrix. To enter such a matrix into MATLAB we type it in row by row using the same syntax as for vectors:
>> A =[5 7 9;1 -3 -7]

Rows can be separated by a new line rather than a semicolon.
>> B = [-1 2 5
9 0 5]
>> C = [0, 1; 3, -2; 4, 2]
>> D =[1:5; 6:10; 11:2:20]
So A and B are 2 × 3 matrices, C is 3 × 2 and D is
3 × 5.
In this context, a row vector is a 1 × n matrix and a column vector an m ×1 matrix.

2.3.1Size of a matrix

We can get the size (dimensions) of a matrix with the command size
>>**size(A), size(x)**

>>**size(ans)**

So A is 2×3 and x is 3×1 (a column vector). The last command size(ans) shows that the value returned by size is itself a 1×2 matrix (a row vector). We can save the results for use in subsequent calculations.

>>**[r c] =size(A'), S =size(A')**

2.3.2Transpose of a matrix

Transposing a vector changes it from a row to a column vector and vice versa. The extension of this idea to matrices is that transposing interchanges rows with the corresponding columns: The 1st row becomes the 1st column, and so on.
>>**D, D'**

>>**size(D), size(D')**

2.3.3Special 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)

zeros(m,n) gives an m × n matrix of 0's,

>> Z =zeros(2,3), zeros(size(P'))

The second command illustrates how we can construct a matrix based on the size of an existing one. **Try ones(size(D))**

An n×nmatrix that has the same number of rowsand columns is called a **square** matrix.
A matrix is said to be **symmetric** if it is equal toit's transpose (i.e. it is unchanged by transposition):
>> S =[2 -1 0; -1 2 -1; 0 -1 2],

>> St =S'

>> S-St

2.3.4The Identity Matrix

The n × n **identity** matrix is a matrix of zeros except for having ones along its leading diagonal (top left to bottom right). This is called **eye(n)** in MATLAB (since mathematically it is usually denoted by I).
>> I =eye(3), x = [8; -4; 1], I*x

Notice that multiplying the 3 × 1 vector x by the 3 × 3 identity I has no effect (it is like multiplying a number by 1).

2.3.5Diagonal Matrices

A diagonal matrix is similar to the identity matrix except that its diagonal entries are not necessarily equal to 1.
D =
3 0 0
0 4 0
0 0 2
is a 3 × 3 diagonal matrix. To construct this in MATLAB, we could either type it in directly
>> D = [-3 0 0; 0 4 0; 0 0 2]

But this becomes impractical when the dimension is large (e.g. a 100 × 100 diagonal matrix). We then use the **diag** function. We first define a vector d, say, containing the values of the diagonal entries (in order) then diag(d) gives the required matrix.

>> d =[-3 4 2], D =diag(d)

On the other hand, if A is any matrix, the command diag(A) extracts its diagonal entries:
>> F = [0 1 8 7; 3 -2 -4 2; 4 2 1 1]
>>diag(F)
Notice that the matrix does not have to be square.

2.3.6Building Matrices

It is often convenient to build large matrices from smaller ones:

>> C=[0 1; 3 -2; 4 2]; x=[8;-4;1];
>> G =[C x]

>> A, B, H =[A; B]

so we have added an extra column (x) to C in order to form G and have stacked A and B on top of each other to form H.

>> J =[1:4; 5:8; 9:12; 20 0 5 4]

>> K =[diag(1:4) J; J' zeros(4,4)]

The command spy(K) will produce a graphical display of the location of the nonzero entries in K (it will also give a value for n—the number of nonzero entries):

>>spy(K), grid

2.3.7Extracting Bits of Matrices

We may extract sections from a matrix in much the same way as for a vector. Each element of a matrix is indexed according to which row and column it belongs to. The entry in the i^{th} row and j^{th} column is denoted mathematically by $A_{i,j}$and, in MATLAB, by A(i,j). So
>> J

>>J(1,1)

>>J(2,3)

>>J(4,3)

>>J(4,5)

>>J(4,1) =J(1,1) + 6

>>J(1,1) =J(1,1) - 3*J(1,2)

In the following examples we will extract
i) the 3rd column,
ii) the 2nd and 3rd columns,
iii) the 4th row, and
iv) the "central" 2 ×2 matrix

>>J(:,3) % 3rd column

>>J(:,2:3) % columns 2 to 3

>>J(4,:) % 4th row

>>J(2:3,2:3) % rows 2 to 3 & cols 2 to 3

Thus, '**:**' on its own refers to the entire column or row depending on whether it is the first or the second index.

2.3.8Dot product of matrices (.*)

The dot product works as for vectors: corresponding elements are multiplied together—so the matrices involved must have the same size.
>> A, B
A =
5 7 9
1 -3 -7
B =
-1 2 5
9 0 5

>> A.*B

>> A.*C

>> A.*C'

2.3.9Matrix–Matrix Products

To form the product of an m × n matrix A and a n×p matrix B, written as AB, we visualize the first matrix (A) as being composed of m row vectors of length n stacked on top of each other while the second (B) is visualized as being made up of p column vectors of length n:
(m× n) times (n ×p) =(m × p).
Check that you understand what is meant by working out the following examples by hand and comparing with the MATLAB answers.

>> A =[5 7 9; 1 -3 -7]
>> B =[0, 1; 3, -2; 4, 2]

>> C =A*B
>> D =B*A

>> E =B'*A'

We see that E = C' suggesting that (A*B)' = B'*A'

Exercise 2.3.1:

Let A=[1 2 3 4 5] , B=[4 3 2 8 9] and C=[5 4 3 2 1] then find (A x B') x C.

POST LAB QUESTIONS:

Q.1 Write MATLAB code to draw graphs of the following functions for x =0:0.1:10. Label your graphs properly.

i) $y = \sin(x)/x$

ii) $u= (1/(x-1)^2)+x$

iii) $v= (x^2+1)/(x^2-4)$

iv) $z = ((10-x)^{1/3}-1)/(4 - x^2)^{1/2}$

EXPERIMENT # 3

LOGICAL OPERATIONS &LOOPS

OBJECTIVE

- Understanding of Logical operations in MATLAB
- Understanding and implementation of loops in MATLAB
- Learn how to create M-Files

3.1 LOGICAL OPERATIONS

MATLAB represents true and false by means of the integers 0 and 1.
True =1, False =0

If at some point in a calculation a scalar x, say, has been assigned a value, we may make certain logical tests on it:
x = =2 Is x equal to 2?
x ~= 2 Is x **not** equal to 2?
x> 2 Is x greater than 2?
x< 2 Is x less than 2?
x >= 2 Is x greater than or equal to 2?
x <= 2 Is x less than or equal to 2?
Pay particular attention to the fact that the test for equality involves two equal signs = =.

>> x =pi
>>x ~=3, x ~=pi

When x is a vector or a matrix, these tests are performed element wise:
x =
-2.0000 3.1416 5.0000
-1.0000 0 1.0000

>>x == 0

>>x> 1, x >=-1

>> y =x>= -1, x > y

We may combine logical tests, as in
x =
-2.0000 3.1416 5.0000
-5.0000 -3.0000 -1.0000

>>x> 3 & x < 4

>>x> 3 | x = =-3

As one might expect, & represents **and,** the vertical bar | means **or**; also ~ means **not** as in ~= (not equal).

>>x> 3 | x ==-3 | x <=-5

One of the uses of logical tests is to "mask out" certain elements of a matrix.
>>x, L =x >= 0

>>pos =x.*L

So the matrix 'pos' contains just those elements of 'x' that are non–negative.

Half-Wave Rectification

>> x =0:0.05:6; y =sin(pi*x); Y =(y>= 0).*y;
>>plot(x,y,':',x,Y,'-')

The results is shown in Figure 3.1.

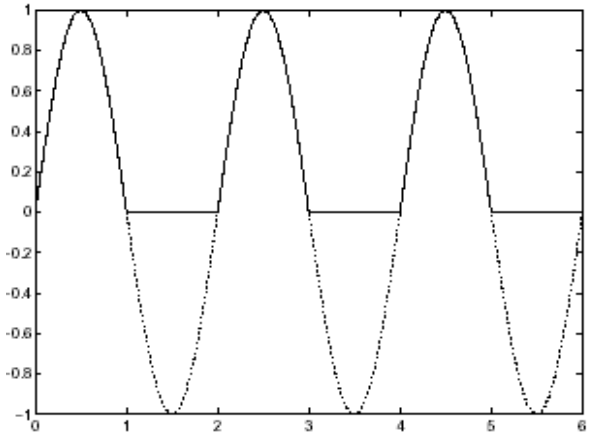


Figure 3.1: Results

Exercise 3.1

Write code to plot y = sin(πx) for y<= 0.

3.2 LOOPS

There are occasions that we want to repeat a segment of code a number of different times (such occasions are less frequent than other programming languages because of the**:** notation).

3.2.1'For' Loops

Example 3.2.1*Draw graphs of sin(nπx) on the interval*
-1 ≤ x ≤ 1 for n = 1, 2, . . .,8.

We could do this by giving 8 separate plot commands but it is much easier to use a loop. Thesimplest form would be

```
>> x =-1:.05:1;
>>for n =1:8
subplot(4,2,n), plot(x,sin(n*pi*x))
end
```

All the commands between the lines starting "**for**" and "**end**" are repeated with **n** being given the value 1 the first time through, 2 the second time, and so forth, until n = 8. The subplot constructs a 4 × 2 array of sub-windows and, on the nth time through the loop, a picture is drawn in the nth sub-window.

The commands

```
>> x =-1:.05:1;
>>for n =1:2:8
subplot(4,2,n), plot(x,sin(n*pi*x))
subplot(4,2,n+1), plot(x,cos(n*pi*x))
end
```

Draw sin nπx and cosnπx for n = 1, 3, 5, 7 alongside each other. We may use any legal variable name as the "loop counter" (n in the above examples) and it can be made to run through all of the values in a given vector (1:8 and 1:2:8 in the examples).
(Keep in mind that using a '**for'** loop in MATLAB is very much discouraged because they are computationally inefficient.)

Example 3.2.2 The Fibonacci sequence starts off with the numbers 0 and 1, then succeeding terms are the sum of its two immediate predecessors.

Mathematically,
f1 = 0, f2 = 1
and
fn= fn-1 + fn-2, n= 3, 4, 5, ...

Test the assertion that the ratio fn/fn-1 of two successive values approaches the golden ratio which is = (√5 + 1)/2= 1.6180.

```
>>F(1) =0; F(2) =1;
>>for i =3:20
F(i) =F(i-1) + F(i-2);
end
>>plot(1:19, F(2:20)./F(1:19),'o' )
>> hold on
>>plot(1:19, F(2:20)./F(1:19),'-' )
>>plot(0:20, (sqrt(5)+1)/2,'--')
```

Example 3.2.3 Produce a list of the values of the sums:

$$S_{20} = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \ ... + \frac{1}{20^2}$$

$$S_{21} = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \ ... + \frac{1}{20^2} + \frac{1}{21^2}$$

⋮

$$S_{100} = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \ ... + \frac{1}{20^2} + \frac{1}{21^2} + \ ... + \frac{1}{100^2}$$

There are a total of 81 sums. The first can be computed using **sum (1./(1:20).^2)**. A suitable piece of MATLAB code might be:

```
>> S =zeros(100,1);
>>S(20) =sum(1./(1:20).^2);
>>for n =21:100
>>S(n) =S(n-1) + 1/n^2;
>>end
>>[ (98:100)' S(98:100)]
```

Where a column vector S was created to hold the answers. The first sum was computed directly using the sum command then each succeeding sum was found by adding $1/n^2$ to its predecessor. The little table at the end shows the values of the last three sums.

Exercise 3.2

Repeat Example 3.2.3 to include 181 sums (i.e. the final sum should include the term $1/200^2$).

3.2.2 'While' Loops

There are some occasions when we want to repeat a section of MATLAB code until some logical condition is satisfied, but we cannot tell in advance how many times we have to go around the loop. This we can do with a while...end construct. The general form of a while statement is shown in Figure 3.2.

```
while a logical test
                Commands to be executed
                when the condition is true
end
```

Figure 3.2: While statement syntax

Example 3.2.4: What is the greatest value of n that can be used in the sum$1 + 2^2 + \ ... + n^2$to get a value of less than 100?

```
>> S =1; n = 1;
>> while S+ (n+1)^2 < 100
n =n+1; S = S + n^2;
end
```

>> [n, S]

The lines of code between while and end will only be executed if the condition **S+ (n+1)^2 < 100 is true**.

3.3M-FILES

M-files are macros of MATLAB command that are stored as ordinary text files with the extension 'm', that is filename.m. An M-file can be either a function with input and output variables or a list of commands (Script File).

For example, consider using MATLAB on a PC with a user-defined M-file stored in a directory called "\MATLAB7.0\work". Then to access that M-file, either change the working directory by typing "cd \matlab7.0\work" from within the MATLAB command window or by adding the directory to path which can be accomplished using the command "addpath c:\matlab\work" from within MATLAB.

As example of an M-file(File→New→M-File) that defines a function, create a file in your working directory named "yplusx.m" that contains the following commands:

```
function z = yplusx(y,x)
z=y+x;
```

The following commands typed from within MATLAB command window demonstrate how this M-file is used:

```
x=2;
y=3;
z=yplusx(y,x);
```
or simply,
```
z=yplusx(3,2);
```

MATLAB M-files are most efficient when written in a way that utilizes matrix or vector operations. Loops are available but should be used sparingly since they are computationally inefficient. An example of the use of the command 'for' is:

```
for k=1:10
        x(k)=cos(k);
end
```

This creates a 1x10 vector x containing the cosine of the positive integers from 1 to 10. This operation is performed more efficiently with the commands

```
k=1:10;
x=cos(k);
```

which utilizes a function of a vector instead of a for loop.

An**"if"** statement can be used to define conditional statements. An example is

```
if(a<=2)
        b=1;
elseif(a>=4)
b=2;
else
        b=3;
end
```

Exercise 3.3

Write MATLAB code to generate a unit step function using 'for' loop and 'if else' statement.

POST LAB QUESTIONS:

Q.1: Create a function in MATLAB to generate a full wave rectified sine wave of frequency 2kHz, amplitude ± 2V and duration of 1 millisecond?

Q.2: For a given matrix, write a generic MATLAB code to get the maximum of each row and afterwards set all the other elements of that row to zero**.**

For example,

	1	2	3	4		0	0	0	4
this matrix:	5	4	6	5	should become:	0	0	6	0
	7	9	8	3		0	9	0	0

EXPERIMENT # 4

GENERATION AND PLOTTING OF BASIC SIGNALS

OBJECTIVE

- Learn to generate and plot some basic signals

4.1UNIT STEP FUNCTION

```
function u = cfstep(t,to)
inc=0.01;
n=-t:inc:t;
u=[zeros(1,(t+to)/inc) 1 ones(1,(t-to)/inc)];
plot(n,u);
```

>>cfstep(5,0)
will generate a unit step function starting at to =0 ranging from $-5 < t \leq 5$.

Exercise 4.1

Modify the above code to generate a generic unit step function starting from $t = t_0$ and ranging from $t_1 < t \leq t_2$. There should be three input parameters i.e. t_1 = lower limit, t_2 = upper limit and t_0 = time where unit step function starts.

4.2SINUSOIDAL SIGNAL

$$y(t) = A\cos\left(\frac{2\pi f_o}{To} + \theta\right)$$
$$y(t) = A\cos(2\pi f_o t + \theta) = A\,cos(w_o t + \theta)$$

Where

A = amplitude of sinusoid

$w_o = fundamental\ frequency\ of\ sinusoid, radians\ per\ \sec(rad/s)$

$f_o = fundamental\ frequency\ of\ sinusoid, Hz$

$T_o = fundamental\ period\ of\ sinusoid$

$t = continuous\ time$

Example:
Suppose A=1, f=1Hz, t=0:0.01:1:
y(t)=cos(2πt)
y(t)=cos(2πt+π/2)
y(t)=cos(2πt-π/2)
y(t)=sin(2πt)

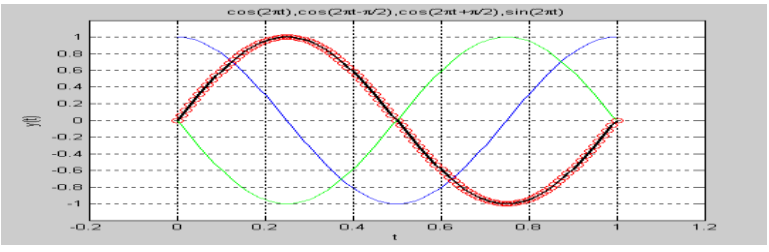


Figure 4.1: y(t)

4.3 PIECEWISE SIGNALS

A piecewise continuous function h(t) (figure 4.2) is defined below in MATLAB

```
>> t=[-1:0.01:2];
>> h=1*[t>=-1 & t<=0];
>> h=h+[-1 * (t>0 & t<=2)];
>>plot(t,h,'linewidth',2)
>>grid
>>axis([-2 3 -1.2 1.2])
```

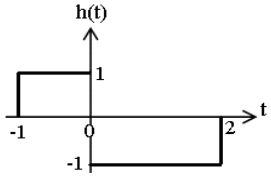


Figure 4.2: h(t)

Exercise 4.2

Define and plot the piecewise continuous function x(t) shown below in figure 4.3:

$$\mathbf{x(t)} = \begin{cases} e^{t} & -1 \le t \le 0 \\ e^{-t} & 0 < t \le 1 \end{cases}$$

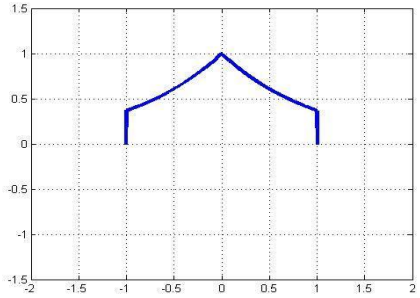


Figure 4.3: x(t)

Exercise 4.3

Define and plot the piecewise continuous function f(x) given below:

$$f(x) = \begin{cases} -\sin(x) & x < 0 \\ x^2 & 0 \le x \le 1 \\ 1/x & x > 1 \end{cases}$$

Exercise 4.4

A rectangular pulse is represented by a step function as shown below in Figure 4.4. Define and plot this rectangular pulse in MATLAB.

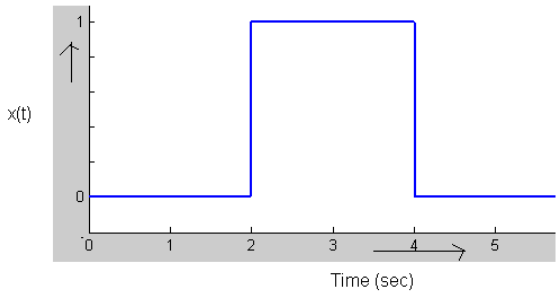


Figure 4.4: x(t)

4.4 EXPONENTIALLY VARYING SINUSOID

An exponentially varying sinusoid is defined below:

$$f(t) = 4e^{-2t}cos(6t - 60^o) \qquad 0 \le t \le 4$$

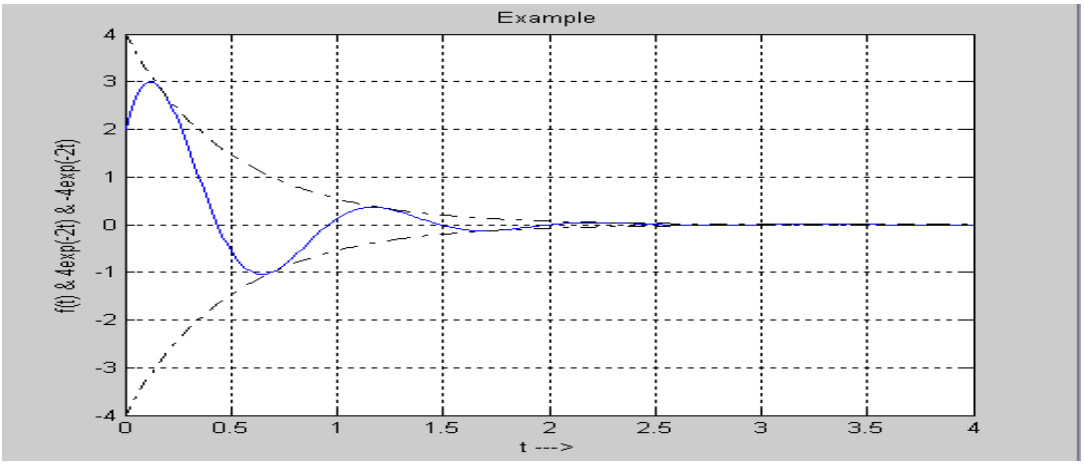


Figure 4.5: f(t)

Exercise 4.5

Write MATLAB code to define this exponentially varying sinusoid$\boldsymbol{f(t)}$ as given in Figure 4.5.Also, include in your code the upper and lower envelopes of$\boldsymbol{f(t)}$.

4.5 MULTIPLICATION OF TWO SIGNALS

$$Y(t) = \begin{cases} 2e^t & -5 \le t \le 0 \\ -t + 2 & 0 \le t \le 10 \end{cases}$$

$$W(t) = \cos(30t)$$

$$V(t) = Y(t) \times W(t)$$

Exercise 4.6

Write MATLAB code to define $\boldsymbol{Y(t)}$and the product$\boldsymbol{V(t) = Y(t) \times W(t)}$.
The result is shown in Figure 4.6.

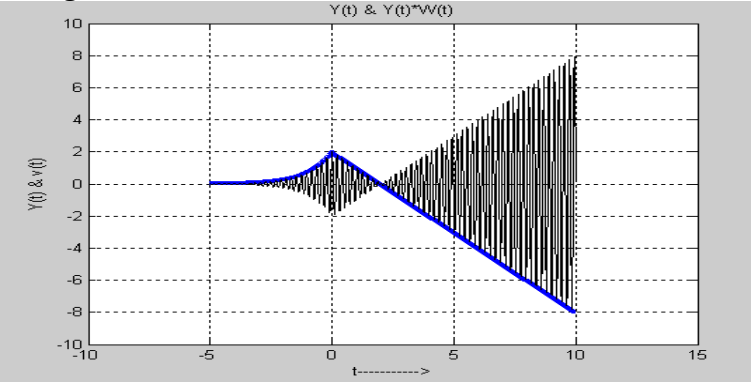


Figure 4.6: V(t)

POST LAB QUESTIONS:

Q.1 What will be the time period and number of cycles for the following periodic sinusoidal signals?

a) Sin(t/3)
b) Cos(6πt)
c) Sin(2t)

Q.2 Define and plot the signal defined below in MATLAB:

$$\boldsymbol{x(t)} = \begin{cases} \boldsymbol{0} & \boldsymbol{t < -4} \\ \boldsymbol{t + 2} & \boldsymbol{-4 \leq t \leq 3} \\ \boldsymbol{t - 2} & \boldsymbol{t > 3} \end{cases}$$

Briefly explain what care must be taken when defining a piecewise signal in MATLAB. What problem would appear in the plotted signal otherwise?

Q.3 Write a function in MATLAB that defines and plots the piecewise signal $\boldsymbol{x(t)}$ shown below in Figure 4.7.

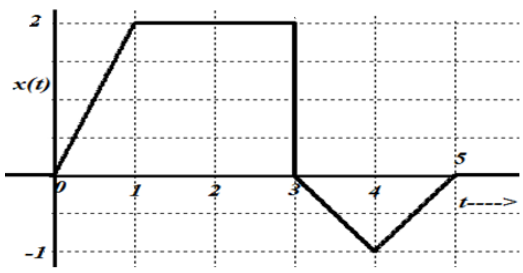


Figure 4.7: x(t)

EXPERIMENT # 5

SIGNAL OPERATIONS

OBJECTIVE

- Understand and perform different operations (time shifting, time inversion and time scaling) on signals

5.1 SIGNAL OPERATIONS

5.1.1 TIME SHIFTING

$$\Phi(t+T) = f(t)$$
$$\Phi(t) = f(t-T)$$

MATLAB function to implement time shifting is given below:

```
function [y,n] = c_shift (x,m,n0)

% implement y(n) = x(n-n0)
        % x = samples of original signal
   % m = index values of the signal
   % n0 = shift amount, may be positive or negative
   % [y,n] = sigshift(x,m,n0)

        n = m+n0;
        y = x;
```

Exercise 5.1:

Implement the function $f(t)$ given below:

$$f(t) = u(t-2)$$

Note that $u(t)$ is the unit step function.

Exercise 5.2:

If$x(t)$ is given by the following expression:

$$x(t) = cos(\pi t) \qquad -0.5 \leq t \leq 0.5$$

Shiftx(t) to the right by 0.5 units. Your result should look like Figure 5.1:

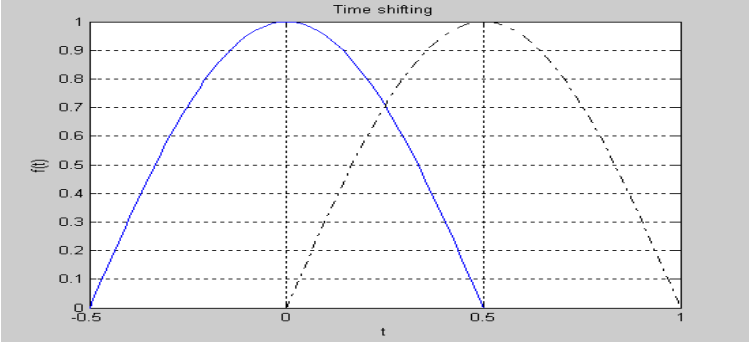


Figure 5.1: x(t) & x(t-0.5)

Exercise 5.3:

Shift the piecewise signal shown in the Figure 5.2 below, 2 units to the right.

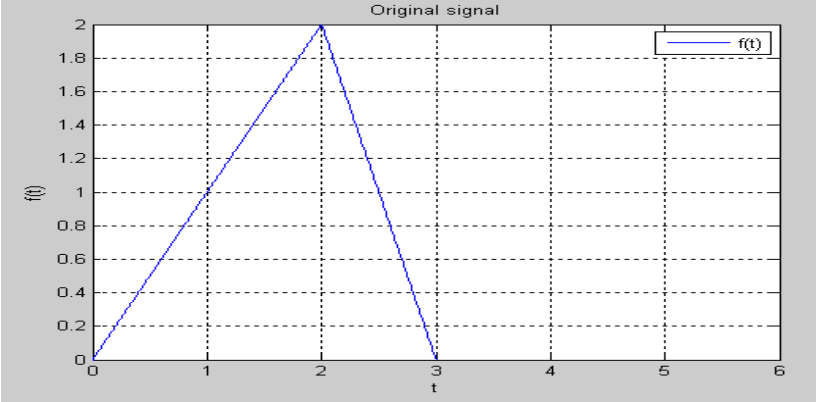


Figure 5.2 : Piecewise signal

5.1.2TIME SCALING

If $f(t)$ is compressed in time by a factor 'a' (a > 1), the resulting signal $\Phi(t)$ is given by

$$\boldsymbol{\Phi(t) = f(at)}$$

Similarly, $f(t)$ expanded in time by a factor 'a' (a > 1) is given by:

$$\boldsymbol{\Phi(t) = f(t/a)}$$

Exercise 5.4:

If $\boldsymbol{x = \cos(\pi t)}$, **for t=-0.5:0.01:0.5;**

Draw $\boldsymbol{\Phi(t) = x(2 * t)}$ **&** $\boldsymbol{\Phi(t) = x(t/2)}$

Your result should be look like as given in Figure 5.3:

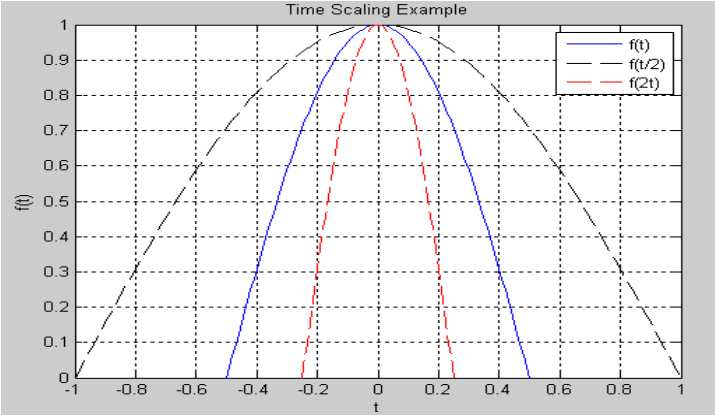


Figure 5.3: $\Phi(t)$& $x(t)$

5.1.3TIME INVERSION (FOLDING)

$$\Phi(-t) = f(t)$$
$$\Phi(t) = f(-t)$$

To time invert a signal, we replace t by –t.

Exercise 5.5:

For$f(t) = e^{t/2}$ $-5 \leq t \leq -1$.Draw$f(-t)$. Your result should be look like Figure 5.4 given below

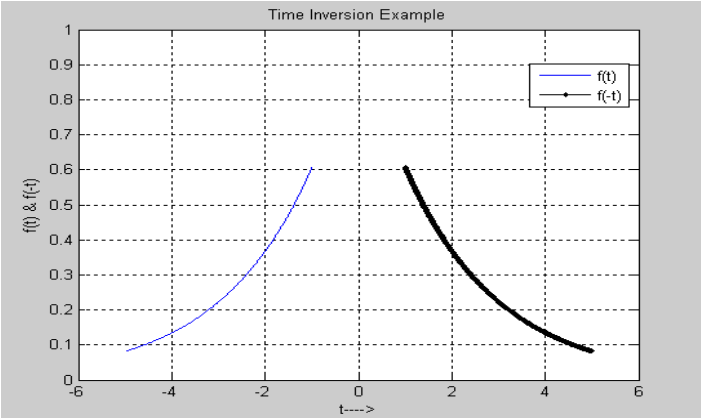


Figure 5.4: f(t) &f(-t)

5.2 COMBINED SIGNAL OPERATIONS

5.2.1 SHIFTING + SCALING

$f(at - b)$can be achieved by the following two procedures:

i) Time shift $f(t)$ by 'b' to obtain$f(t-b)$. Now scale the shifted signal $f(t-b)$by 'a' (i.e. replace 't' with 'at') to obtain $\boldsymbol{f(at-b)}$

ii) Time-scale $f(t)$ by 'a' to obtain$f(at)$. Now time-shift $f(at)$ by 'b/a' (i.e. replace t by t-b/a) to obtain:

$$\boldsymbol{f\left(a\left(t-\frac{b}{a}\right)\right)=\ f(at-b)}$$

Similarly $\boldsymbol{f\left(\frac{t}{a}-b\right)}$ can be achieved by the following two procedures:

i) Time shift $f(t)$ by 'b' to obtain$f(t-b)$. Now scale the shifted signal $f(t-b)$by 'a' (i.e. replace 't' with 't/a') to obtain $\boldsymbol{f\left(\frac{t}{a}-b\right)}$

ii) Time-scale $f(t)$ by 'a' to obtain$f(t/a)$. Now time-shift$f(t/a)$ by 'ab' (i.e. replace t by t-ab) to obtain

$$\boldsymbol{f\left(\frac{t}{a}-b\right)=f\left(\frac{t-ab}{a}\right)}$$

Exercise 5.6:

For the given function

$\boldsymbol{f(t)=\frac{3}{2}\,t \quad 0\le t\le 2,}$

Plot $\boldsymbol{f(5t-7)}$

5.2.2 INVERSION +SCALING

Combined operations of inversion and scaling,$\boldsymbol{f(-at)}$and$\boldsymbol{f(-t/a)}$, can be achieved by either first time scaling the function by 'a' and then performing time inversion (i.e. replace t by -t) or by first time inverting the signal and then time scaling by 'a'.

Exercise 5.7:

For the given function$\boldsymbol{x(t)}$ in Figure 5.5, plot$\boldsymbol{x(^{-t}/_{2})}$

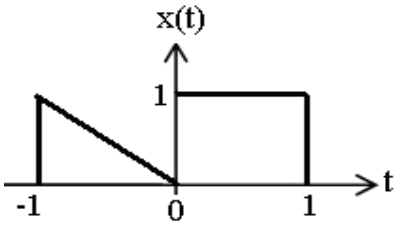


Figure 5.5: x(t)

5.2.3 SHIFTING + INVERSION

$\boldsymbol{f(a-t)}$can be achieved by the following two procedures:

i) Time invert $f(t)$to obtain$f(-t)$ and then shift the resulting signal $f(-t)$ by 'a' to obtain $f\big(-(t-a)\big) = f(a-t)$.

ii) Shift $f(t)$ by 'a' to obtain $f(t+a)$ and then time invert (replace t by -t) the resulting signal to obtain $f(-t+a) = f(a-t)$.

The same two procedures can be applied in turn to achieve$\boldsymbol{f(-t-a) = f(-(t+a))}$.

Exercise 5.8:

For $\boldsymbol{f(t) = \frac{3}{2}\,t \quad 0 \le t \le 2}$, Plot $\boldsymbol{f(-t-2)}$.

5.2.4 SHIFTING + INVERSION + SCALING

All the three transformations, time scaling, time inversion and time shifting can be applied simultaneously, for example,

$$\boldsymbol{f\left(\frac{-t-b}{a}\right)}$$

To understand the overall effect, it is usually best to break this transformation down into successive simple transformations,

$$\boldsymbol{f(t) \rightarrow f\left(\frac{t}{a}\right) \rightarrow f\left(\frac{-t}{a}\right) \rightarrow f\left(\frac{-(t+b)}{a}\right)}$$

Observe here that the order of transformations is important. If we change the order of time-scaling and time-shifting operations, we get

$$\boldsymbol{f(t) \rightarrow f(t-b) \rightarrow f(-t-b) \rightarrow f\left(\frac{-t}{a}-b\right) \neq f\left(\frac{-t-b}{a}\right)}$$

The result of this sequence of transformations is different from the preceding result. Similarly you should note the order of individual transformations for this combined transformation, $\boldsymbol{f(-at-b)}$.

Exercise 5.9:

For $\boldsymbol{f(t) = \frac{3}{2}\,t \quad 0 \le t \le 2}$, Plot $\boldsymbol{f\left(\frac{-t+2}{4}\right)}$

POST LAB QUESTIONS:

Q.1 Write MATLAB code to define the given function $x(t)$ in Figure 5.6, then use signal operations to plot $3x(^{-t}/_{2})$

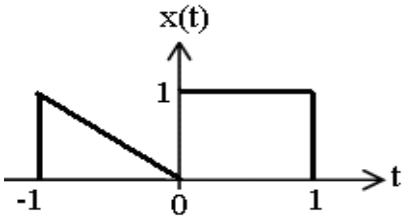


Figure 5.6: x(t)

Q.2 Using MATLAB, plot the function defined by

$$g(t) = \begin{cases} 0 & t < -2 \\ -4 - 2t & -2 < t < 0 \\ -4 + 3t & 0 < t < 4 \\ 16 - 2t & 4 < t < 8 \\ 0 & t > 8 \end{cases}$$

Then plot the transformed functions

$$3g(t+1)\,, \qquad \frac{1}{2}\,g(3t), \qquad -2g\left(\frac{t-1}{2}\right)$$

EXPERIMENT # 6

EVEN AND ODD COMPONENTS OF A SIGNAL

CONVOLUTION OF LTI SYSTEMS

OBJECTIVE

- Decomposition of a signal into its even and odd components
- Understanding and implementation of continuous time convolution

6.1 EVEN & ODD Components of a Signal

Define the signal x(t) shown below (Figure 6.1)in MATLAB.

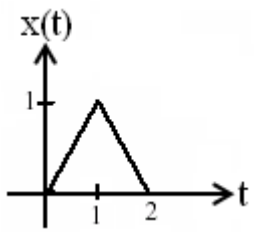


Figure 6.1: x(t)

Note: Since we require to flip the function in time domain and then add the two functions, so the time duration defined for x(t) should span the complete duration required to plot x(t) and x(-t). For example, in this case x(t) is non zero for $0 \leq t \leq 2$ and so x(-t) will be non-zero for $-2 \leq t \leq 0$. So the time duration you define in MATLAB should at least be $-2 \leq t \leq 2$ as shown below in Figure 6.2 to properly execute the rest of operations.

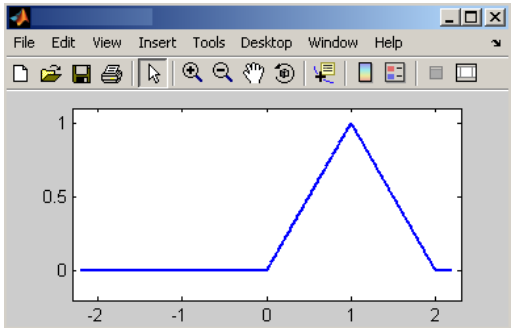


Figure 6.2: x(t)

i) Use the built-in MATLAB function *fliplr()* to define x(-t).

ii) Use the formula to determine the even component of a given function i.e. $x_e(t) = \left[\frac{x(t)+x(-t)}{2}\right]$ in MATLAB.

iii) Use the formula to determine odd component of a given function i.e. $x_o(t) = \left[\frac{x(t)-x(-t)}{2}\right]$ in MATLAB.

iv) Use *subplot()* function to plot the signals obtained in parts above.

Exercise6.1.1:

Determine and plot even and odd components of function f(t) shown below in Figure 6.3.

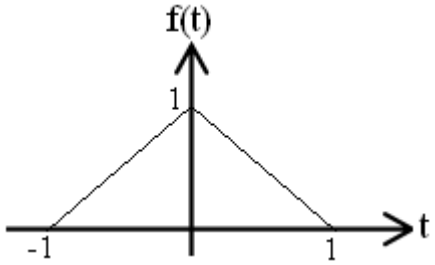


Figure 6.3: f(t)

Observe that the given function f(t) is an even function so

- It's odd component should be equal to zero
- f(-t) should look like f(t).

Exercise6.1.2:

Determine and plot even and odd components of function g(t) shown below in Figure 6.4.

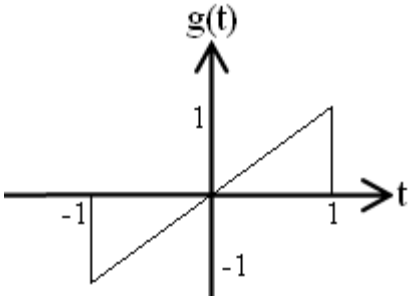


Figure 6.4: g(t)

Observe that the given function g(t) is an odd function so it's even component should be equal to zero.

6.2 Convolution of LTI system

Define the functions x(t) and h(t) given below (Figure 6.5) in MATLAB.

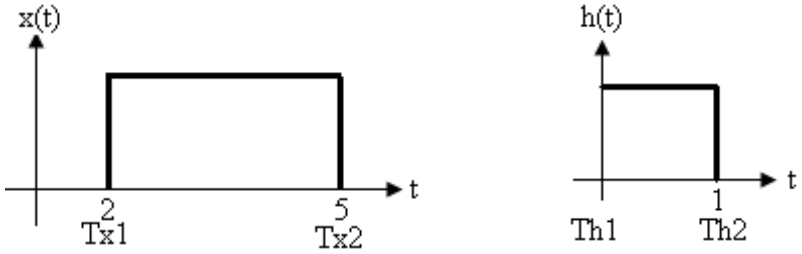


Figure 6.5 x(t) & h(t)

i) Use ***conv()*** function in MATLAB to convolve the two functions defined above.
ii) Plot *x(t), h(t)* and convolution result *y(t)*in MATLAB on a single figure.
iii) Note that the result of convolution spans from (Tx1+Th1) ≤ t ≤ (Tx2+Th2), so plot the graph of *y (t)* against this time.
iv) Your final result should look like the Figure 6.6 below:

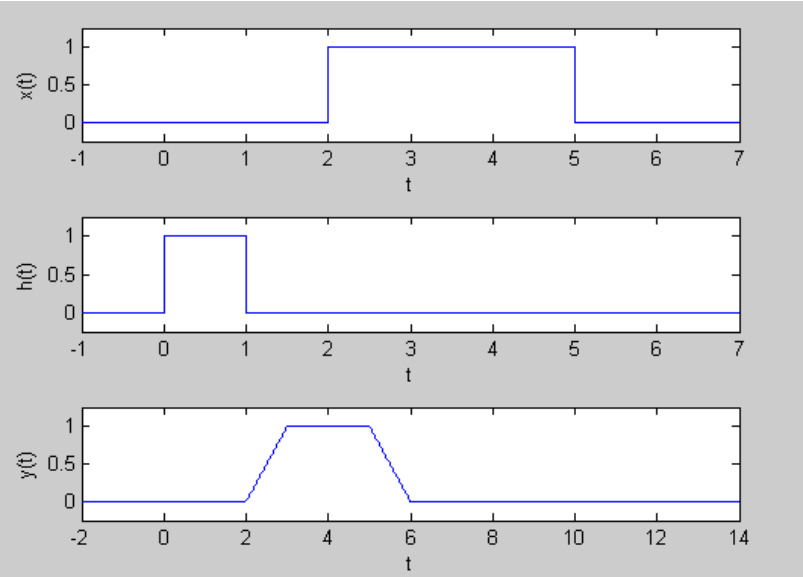


Figure 6.6: Final result

Exercise6.2.1:

For the two functions x(t) and h(t) shown below in Figure 6.7
i) Define the functions x(t) and h(t) in MATLAB.
ii) Use ***conv()*** function in MATLAB to convolve x(t) and h(t).
iii) Plot *x, h* and *y* on a single figure window using *subplot()* function.

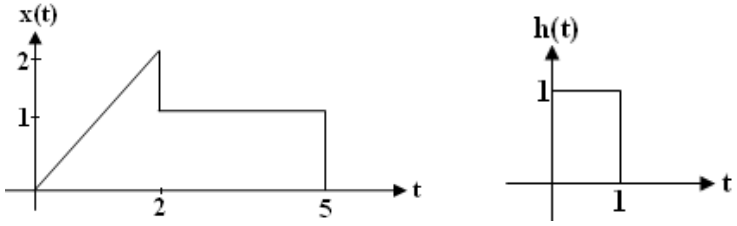


Figure 6.7: x(t) & h(t)

Exercise6.2.2:

For the two functions x(t) and h(t) shown below in Figure 6.8
i) Define the functions x(t) and h(t) in MATLAB.
ii) Use ***conv()*** function in MATLAB to convolve x(t) and h(t).
iii) Plot *x, h* and *y* on a single figure window using *subplot()* function.

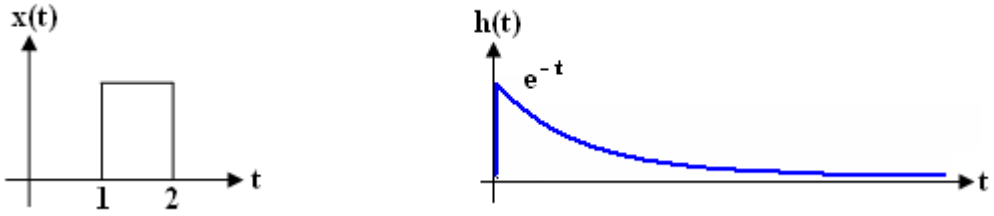


Figure 6.8: x(t) & h(t)

POST LAB QUESTIONS:

Q.1: Convolve the signals $x(t)$ and $h(t)$ given below in Figure 6.9 using Matlab.

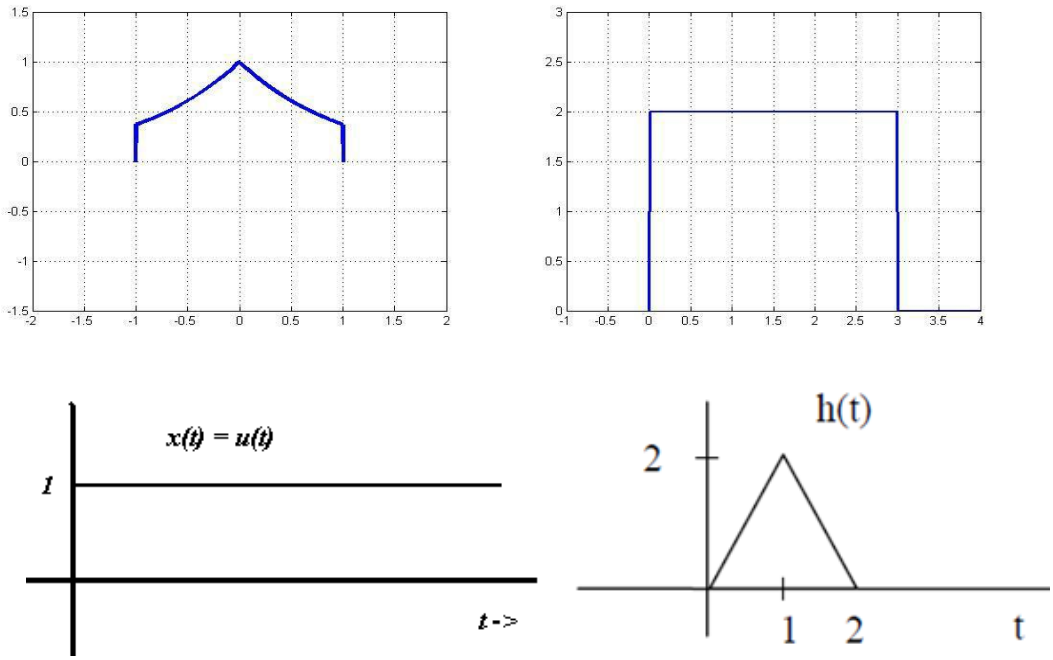


Figure 6.9: x(t) &h(t)

Q.2 Explain how the in-built function for convolution can be used to map continuous time convolution in MATLAB. What care must be taken while plotting the convolution result?

DESIGN EXPERIMENT # 7

SYSTEM RESPONSE OF CONTINUOUS-TIME LTI SYSTEM

7.1 PROBLEM STATEMENT

Design the systems given in Figure-7.1 and Figure-7.2 using MATLAB and then compare its output $y_1[t]$ and $y_2[t]$.

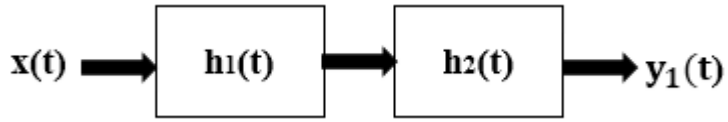


Figure 7.1: System

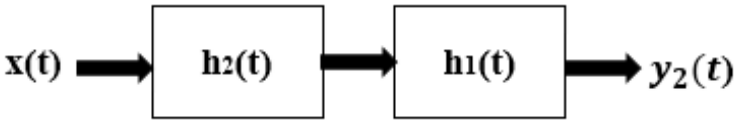


Figure 7.2: System

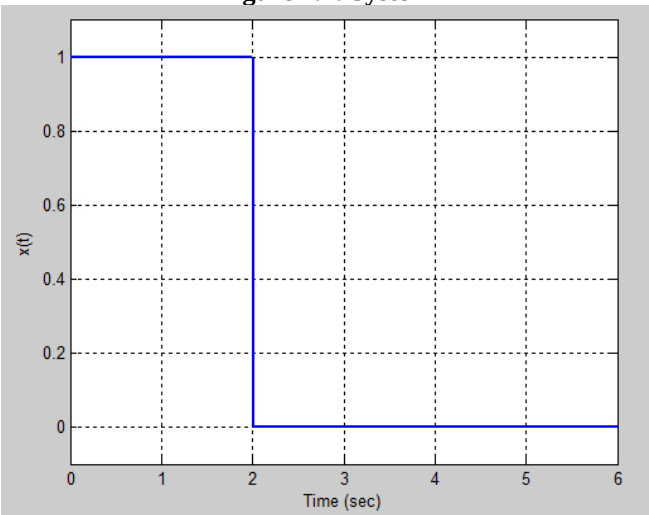


Figure 7.3: x(t)

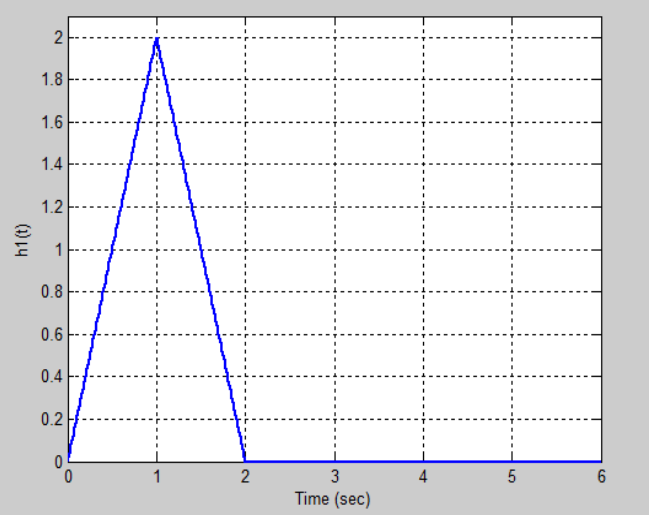


Figure 7.4: $h_1(t)$

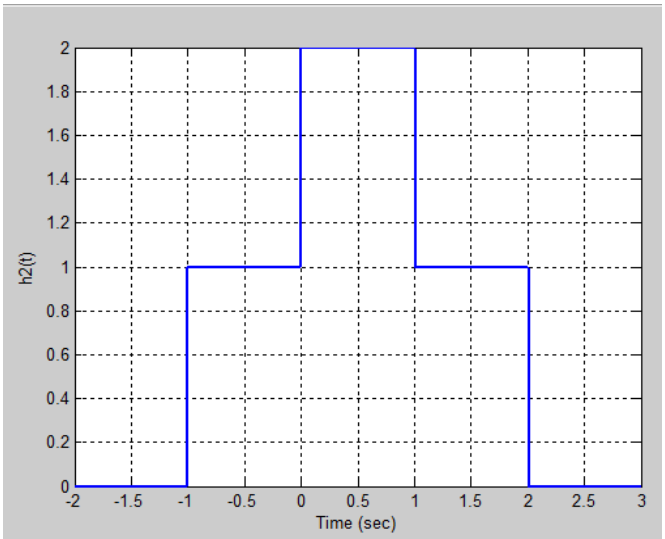


Figure 7.5: $h_2(t)$

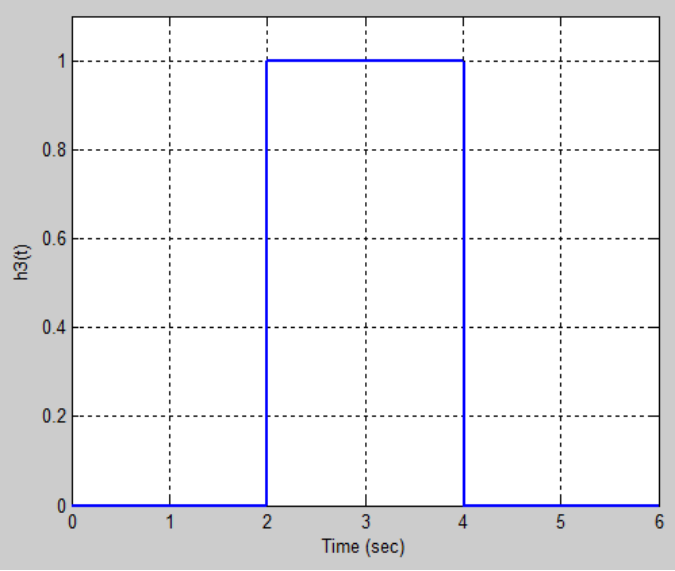


Figure 7.6: $h_3(t)$

- Write a MATLAB code for the signals x[t], $h_1[t]$ and $h_2[t]$. SLO-4.1
- Write down the system response equation of systems in terms of x[t], $h_1[t]$ and $h_2[t]$. SLO-4.2
- Write MATLAB code for the system in Figure-7.1 and Figure-7.2 using system response equation. SLO 4.3
- Using MATLAB, generate the plots of $y_1[t]$ and $y_2[t]$. SLO 4.4
- Are these plots of $y_1(t)$ and $y_2(t)$ are similar or different? State the reason for their similarity or difference. SLO 4.5
- What will be the time range of the output singals $y_1[t]$ and $y_2[t]$? SLO 4.6

7.2 PROBLEM STATEMENT:

Design the systems given in Figure-7.7 and Figure-7.8 using MATLAB.

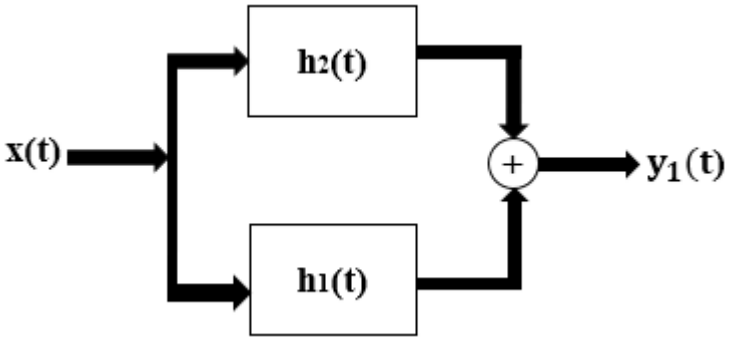


Figure 7.7: System

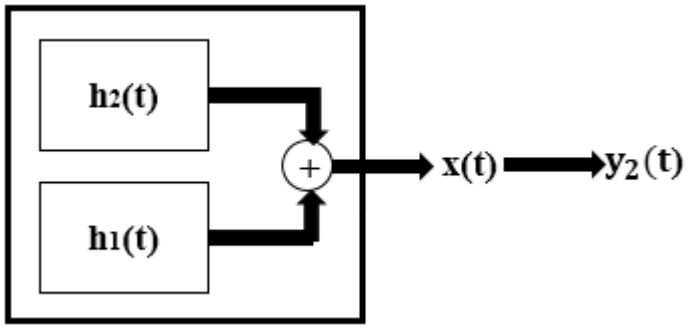


Figure 7.8: System

- Write a MATLAB code for the signals x[t], $h_1[t]$and$h_2[t]$. SLO-4.1
- Write down the system response equation of systems in terms of x[t], $h_1[t]$and$h_2[t]$. SLO-4.2
- Write MATLAB code for the system in Figure-7.7 and Figure-7.8 using system response equation. SLO 4.3
- Using MATLAB, generate the plots of $y_1[t]$ and $y_2[t]$. SLO 4.4
- Are these plots of $y_1(t)$ and $y_2(t)$ are similar or different? State the reason for their similarity or difference. SLO 4.5
- What will be the time range of the output singals $y_1[t]$ and $y_2[t]$? SLO 4.6

7.3 PROBLEM STATEMENT:

Design the systems given in Figure-7.9 and Figure-7.10 using MATLAB.

- Write a MATLAB code for the signals x[t], $h_1[t]$, $h_2[t]$ and $h_3[t]$. SLO-4.1
- Write down the system response equation of systems in terms of x[t], $h_1[t]$, $h_2[t]$ and $h_3[t]$. SLO-4.2
- Write MATLAB code for the system in Figure-7.9 and Figure-7.10 using system response equation. SLO 4.3
- Using MATLAB, generate the plots of $y_1[t]$ and $y_2[t]$. SLO 4.4
- Are these plots of $y_1(t)$ and $y_2(t)$ are similar or different? State the reason for their similarity or difference. SLO 4.5
- What will be the time range of the output singals $y_1[t]$ and $y_2[t]$? SLO 4.6

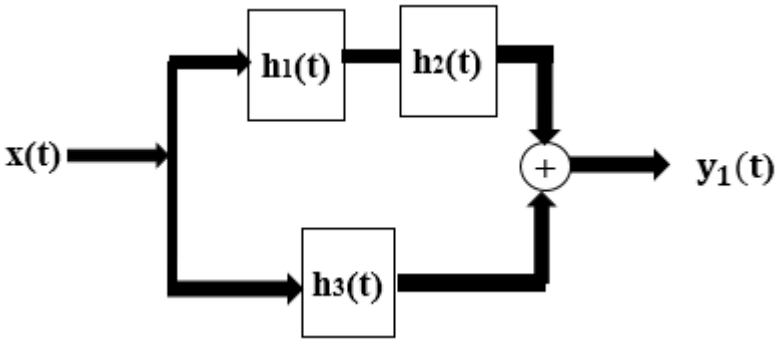


Figure 7.9: System

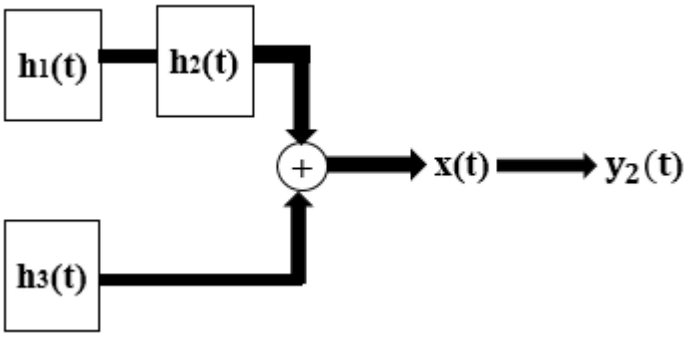


Figure 7.10: System

EXPERIMENT # 8

FOURIER ANALYSIS OF PERIODIC SIGNALS USING TRIGONOMETRIC FOURIER SERIES

OBJECTIVE

- Decomposition of a periodic signal into its Trigonometric Fourier series coefficients
- Plotting of magnitude and phase spectra using trigonometric coefficients

TRIGONOMETRIC FOURIER SERIES

A periodic signal $f(t)$with a period T_o can be expressed as a sum of a sinusoid of frequency f_o and its entire harmonics, as shown in equation below:

$$\boldsymbol{f(t) = a_o + \sum_{n=1}^{\infty} a_n \cos \omega_0 nt + b_n \sin \omega_0 nt}$$

$$\boldsymbol{= C_o + \sum_{n=1}^{\infty} C_n \cos(n\omega_o t + \theta_n)}$$

Where

$$\boldsymbol{C_o = a_o}$$

$$\boldsymbol{C_n = \sqrt{a_n^2 + b_n^2}}$$

$$\boldsymbol{\theta_n = \tan^{-1}\left(\frac{-b_n}{a_n}\right)}$$

Trigonometric Fourier series coefficients can be computed using following formulas.

$$\boldsymbol{a_o = \frac{1}{T_o}\int f(t)\ dt}$$

$$\boldsymbol{a_n = \frac{2}{T_o}\int f(t) \cos n\omega_o t\ dt}$$

$$\boldsymbol{b_n = \frac{2}{T_o}\int f(t) \sin n\omega_o t\ dt}$$

Exercise 8.1:
Find the Trigonometric Fourier series coefficients and plot the magnitude and phase spectra for the periodic signals shown in Figure 8.1:

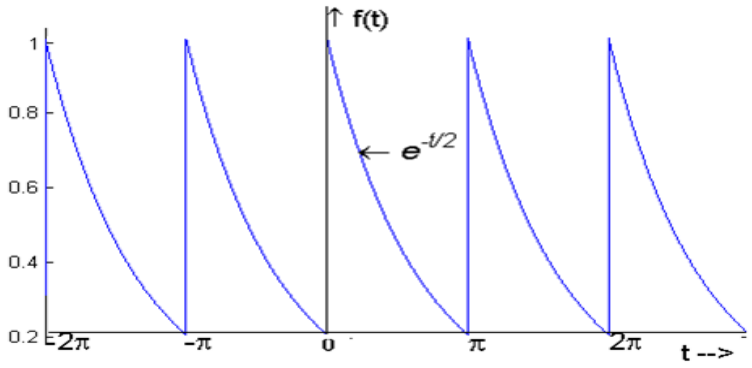


Figure 8.1: f(t)

Fourier series coefficients for the periodic signal shown above are:

$$a_0 = 0.504, \qquad a_n = 0.504\left(\frac{2}{1+16n^2}\right) b_n = 0.504\left(\frac{8n}{1+16n^2}\right)$$

The code for plotting the magnitude and phase spectra for the given co-efficients is given below using for loop:

```
n=1:7;
a0=0.504;
b0=0.504*(8*0/(1+16*0^2));          % b0=0;
Cn=a0;
theta0=atan(-b0/a0);
thetan=theta0;
den=(1+16*n.^2);
N=length(den);
for i=1:N
an(i)=0.504*2/den(i);
bn(i)=0.504*8*n(i)/den(i);
cn=sqrt(an(i)^2+bn(i)^2);
Cn=[Cncn];
theta=atan(-bn(i)/an(i));
thetan=[thetan theta];
end
n=0:7;
subplot(211),stem(n,Cn,'o'),grid,xlabel('n'),ylabel('C_n'),title(
'Question 1')
subplot(212),stem(n,thetan,'o'),grid,xlabel('n'),ylabel('\theta_n
(rad)')
```

Exercise8.2: If $f(t)$ is defined as below:

$$\boldsymbol{f(t)}= \begin{cases} \boldsymbol{2At} & \boldsymbol{|t| \leq 1/2} \\ \boldsymbol{2A(1-t)} & \boldsymbol{1/2 < t \leq 3/2} \end{cases} \text{Where } A = 1$$

Find the trigonometric Fourier series coefficients for the periodic signal given below in Figure 8.2 and use them to plot the magnitude and phase spectra in MATLAB using vector method.

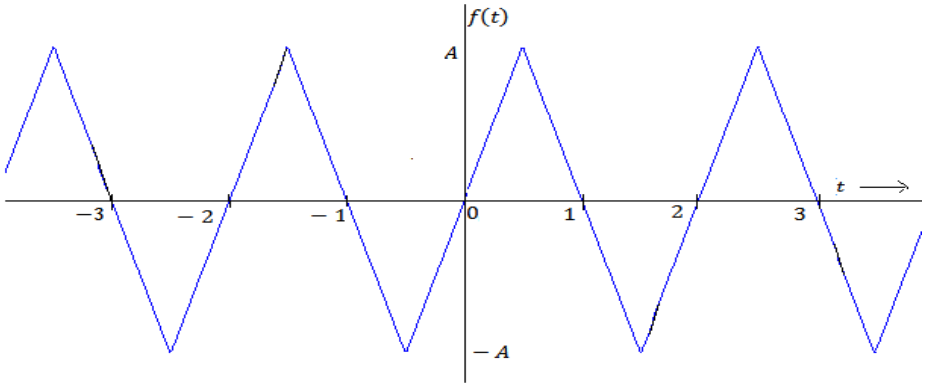


Figure 8.2: f(t)

Exercise 8.3:
For the periodic signal given below in Figure 8.3, mathematically compute the trigonometric Fourier series coefficients and plot the amplitude and phase spectra (in degrees).

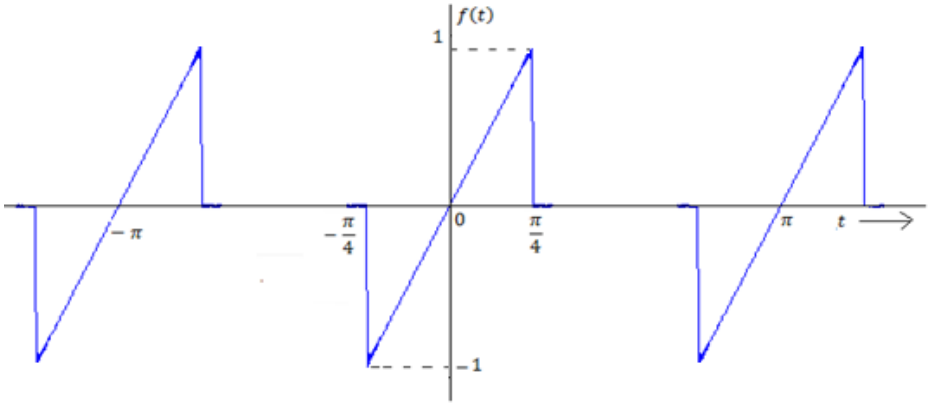


Figure 8.3: f(t)

POST LAB QUESTIONS:

Q.1 Write MATLAB code to plot the co-efficientsan, bn computed in lab with respect to 'n' for 'n' ranging between 0 – 20. Also include the figures in your report.

EXPERIMENT # 9

FOURIER ANALYSIS OF PERIODIC SIGNALSUSING EXPONENTIAL FOURIER SERIES

OBJECTIVE

- Decomposition of a periodic signal into its Exponential Fourier series coefficients
- Plotting of magnitude and phase spectra using exponential coefficients

EXPONENTIALFOURIER SERIES

$$f(t) = D_O + \sum_{n=1}^{\infty} D_n\, e^{jn\omega_o t} + D_{-n} e^{-jn\omega_o t}$$

$$f(t) = D_O + \sum_{n=-\infty(n\neq 0)}^{\infty} D_n\, e^{jn\omega_o t}$$

$$D_o = \frac{1}{T_o}\int f(t)\ dt$$

$$D_n = \frac{1}{T_o}\int f(t)\, e^{-jn\omega_o t}\ dt$$

Exercise 9.1: Write MATLAB code to plot the spectra of exponential Fourier series using following exponential Fourier series coefficients for $-10 \leq n \leq 10$.

$$D_o = 0.504$$

$$D_n = \frac{0.504}{1 + j4n}$$

Exercise 9.2: Write MATLAB code to plot the spectra of Exponential Fourier series using following Exponential Fourier series coefficient forgiven signal in Figure 9.1.

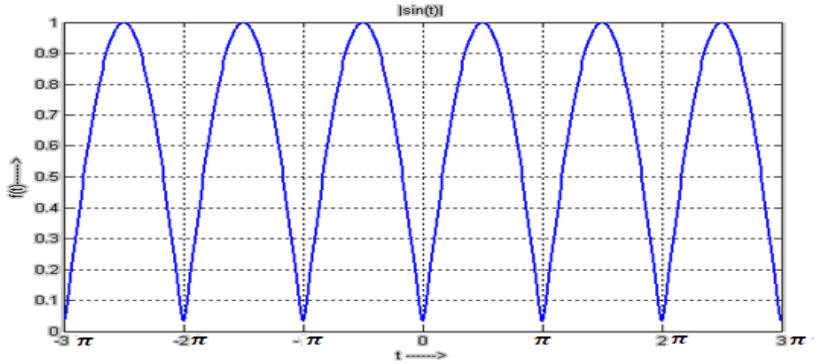


Figure 9.1: Signal

$$D_n = \frac{2}{\pi(1-4n^2)}$$

Exercise 9.3: Find the Exponential Fourier series coefficients for the given signal in Figure 9.2 and plot the spectra of Exponential Fourier series for $-10 \leq n \leq 10$.

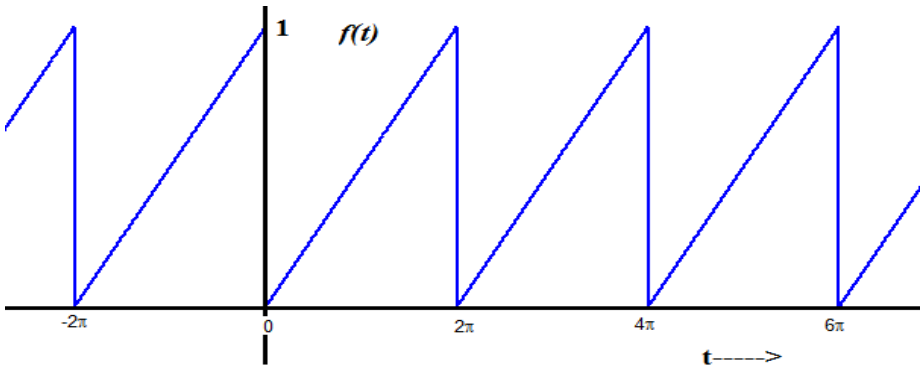


Figure 9.2: f(t)

POST LAB QUESTIONS:

Q.1 Write a MATLAB function, [a_k] = EFS(x,T,n), which gets the input function x(t) as a vector x, the period (T in sec), and number of coefficients (n) as the input arguments so as to calculate and return the exponential Fourier series coefficients in a vector a:

$$a_k = \frac{1}{T}\int_T x(t)e^{-jk\omega_0 t}dt$$

Here, 'a' is a vector of length 2n + 1, i.e. k varies from –n to n.

DESIGN EXPERIMENT # 10

FOURIER SYNTHESIS OF PERIODIC SIGNALS

OBJECTIVE

- Design an experiment to perform Fourier synthesis of a periodic signal given the Fourier coefficients. You will be evaluated based upon the rubrics of PLO-4.

FOURIER SYNTHESIS

A periodic signal,$\boldsymbol{f(t)}$, can be reconstructed from sine and cosine waves with frequencies that are multiples of the fundamental frequency,f_O. The $\boldsymbol{a_n}$ and $\boldsymbol{b_n}$ co-efficients hold the amplitudes of cosine and sine waves respectively. Therefore, any periodic signal is a sum of discrete frequency sinusoidal components. We can also represent $\boldsymbol{f(t)}$ using a sum of harmonically related complex exponential signals$\boldsymbol{e^{jn\omega_o t}}$. The component frequencies $\boldsymbol{n\omega_O}$ are integer multiples of the fundamental frequency. In general one needs an infinite series of such components.

10.1 PROBLEM STATEMENT:

For a signal $x(t)$with $\omega_0 = 2\pi$, the trigonometric Fourier coefficients are given below:

$$a_0 = \frac{1}{2}$$

$$a_n = 0$$

$$b_n = \frac{1}{2\pi}[1 - \cos(n\pi)]$$

- Synthesize x(t) using Fourier series co-efficients. SLO-4.1
- Find the unknowns for writing the function. SLO-4.2
  - Range of time
  - Input parameters of the function
- Write a MATLAB function for x(t). SLO 4.3
- Using MATLAB, generate the plots of x(t) for four different values of N, where N is the total number of non-zero harmonics. SLO 4.4
- Investigate and research about Gibbs phenomenon and explain it using the output of above function. SLO 4.5
- What is the effect of increasing N on the plots? SLO 4.6

10.2 PROBLEM STATEMENT:

The trigonometric Fourier series coefficients for the signal $f(t)$with $\omega_0 = \pi$ are given below:

$$a_0 = 0$$

$$a_n = 0$$

$$b_n = \frac{8}{n^2\pi^2}\left[\sin(\frac{n\pi}{2})\right]$$

- Synthesize x(t) using Fourier series co-efficients. SLO-4.1
- Find the unknowns for writing the function. SLO-4.2
    - Range of time
    - Input parameters of the function
- Write a MATLAB function for x(t). SLO 4.3
- Using MATLAB, generate the plots of x(t) for four different values of N, where N is the total number of non-zero harmonics. SLO 4.4
- Investigate and research about Gibbs phenomenon and explain it using the output of above function. SLO 4.5
- What is the effect of increasing N on the plots? SLO 4.6

POST LAB:

Question:

Repeat the lab work on the signal with following Fourier coefficients.

$$\omega_0 = 2$$

$$a_0 = 0$$

$$a_n = 0$$

$$b_n = \frac{4}{n^2\pi^2}\sin\left(\frac{n\pi}{2}\right) - \frac{2}{n\pi}\cos\left(\frac{n\pi}{2}\right)$$

EXPERIMENT # 11

FOURIER TRANSFORM &INVERSE FOURIER TRANSFORM

OBJECTIVE

- Learn to perform integration in MATLAB
- Learn to compute Fourier Transform and Inverse Fourier Transform

11.1SYMBOLIC EXPRESSION

The *sym* function can be used to create 'symbolic objects' in MATLAB. For example, typing >>x=sym('x') creates the symbolic variable with name x.

The *syms*function enables you to combine more than one such statement into a single statement. For example, typing *syms x* is equivalent to typing *x=sym('x')*, and typing *syms x y u v* creates the four variables x, y, u and v.

You can use sym function to create symbolic constants by single numerical value for the argument. For example, typing ***pi=sym('pi')***, ***fraction=sym('1/3')***and ***sqroot2=sym('sqrt(2)')*** create symbolic constants that avoid the floating point approximations inherent in the values of pi, 1/3 and sqrt(2). If you create a symbolic constant pi this way, it temporarily replaces the built-in numeric constant, and you no longer obtain a numeric value when you type its name.

The advantage of using symbolic constants is that they need not be evaluated (with the accompanying round-off error) until a numeric answer is required.

11.2INTEGRATION IN MATLAB

int(S)
int(S,a,b)

int(S) returns the indefinite integral of S with respect to its symbolic variable.

int(S,a,b) returns the definite integral from a to b of each element of S with respect to each element's default symbolicvariable a and b are symbolicor doublescalars.

Write following code in MATLAB and write your results in space provided

>>**syms x t n**
>>int(x)

>>int(cos(x))

>>int(sin(x))

>>int(exp(-2*t))

>>int(exp(-2*j*t))

>>int(t*exp(-t))

>>int(cos(x),0,2*pi)

>>int(sin(x),0,2*pi)

>>int(exp(-t/2)*cos(n*t),0,pi)

>> a0=int(exp(-t/2),0,pi)

11.3FOURIER TRANSFORM IN MATLAB

$$\boldsymbol{f(t) \overset{F}{\Longleftrightarrow} F(\omega)}$$

$F = fourier(f)$is the Fourier transform of the symbolic scalar f with default independent variable$t$. The default return is a function of ω. The Fourier transform is applied to a function of t and returns a function ofω.

Write following code in MATLAB and observe the output.
syms t
fourier(cos(t))

Explain what is dirac() function in MATLAB.

Find the Fourier transform of **δ(t)** in MATLAB.

Unit step Function is defined as **heaviside(t)** in MATLAB. Find Fourier transform of $\boldsymbol{u(t)}$in MATLAB.

Define $\boldsymbol{rect\left(\frac{t}{4}\right)}$in MATLAB as a sum of shifted heaviside() functions. Find its Fourier transform.

Define$\boldsymbol{e^{-2t}u(t)}$in MATLAB. Find its Fourier transform.

fourier() function takes in a symbolic variable and cannot plot the function as long as it's variable is a symbol. So after you have derived Fourier transform of a function, define 't' for some fixed value e.g. t=[-10:0.01:10] and w=[-3π:0.1:3π], and rewrite the equations to plot the graphs.

Example 11.1

```
syms t
x=heaviside(t+2)-heaviside(t-2);
f=fourier(x)
t=[-10:0.01:10];
w=[-3*pi:.1:3*pi];
x=heaviside(t+2)-heaviside(t-2);
f=2./w.*sin(2.*w);
subplot(3,1,1)
plot(t,x)
subplot(3,1,2)
plot(w,abs(f))
subplot(3,1,3)
plot(w,angle(f))
```

The final graph should look like Figure 11.1:

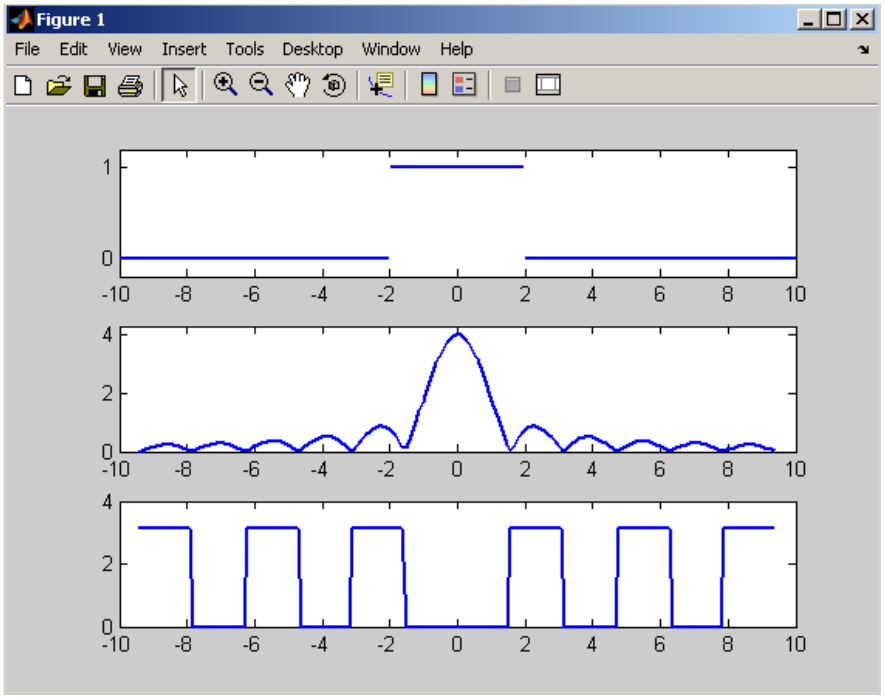


Figure 11.1: Final graph

Note that actually the graph should be like Figure 11.2

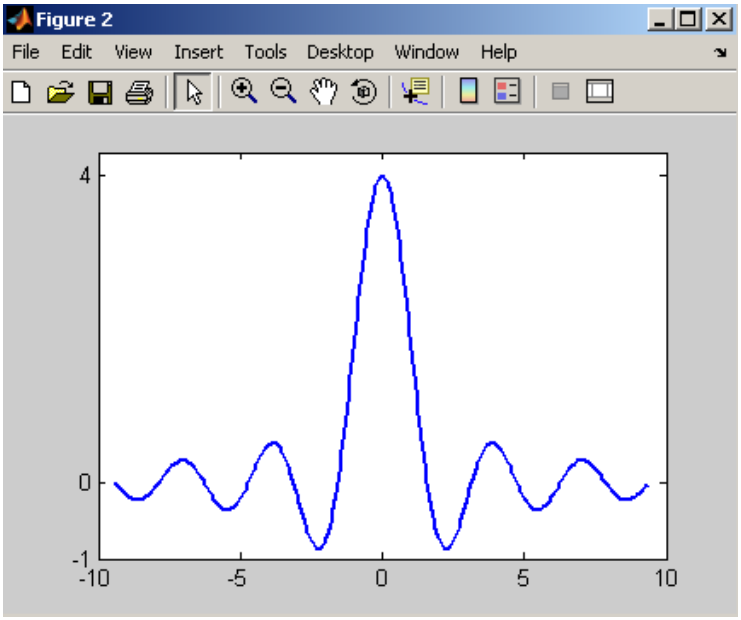


Figure 11.2: sinc function

But when you plot the absolute value of the transform, then it takes all the positive values in place of negative values and gives a phase of pi in the phase graph.

Exercise 10.1: Find and plot the Fourier transform of $e^{-2t}u(t)$.

Your final result should look like the Figure 11.3 below:

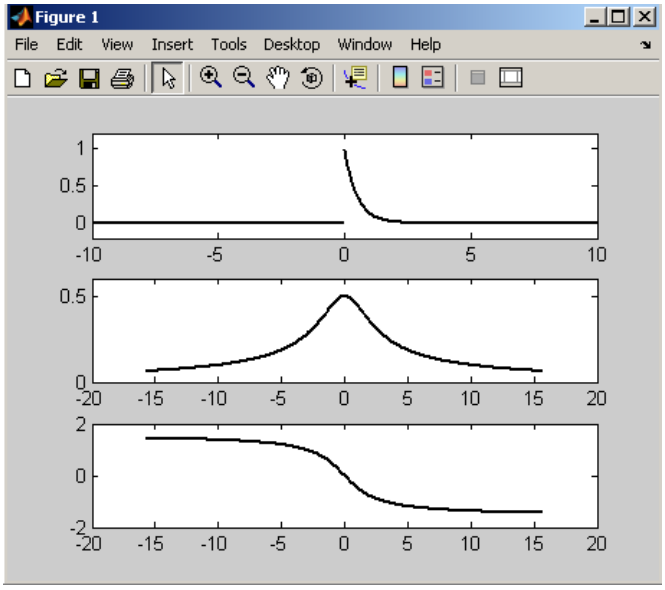


Figure 11.3: final result

10.4 INVERSE FOURIER TRANSFORM

If you have obtained the frequency domain expression $\boldsymbol{F(\omega)}$, then you can find the corresponding time domain expression by finding its inverse Fourier transform. In MATLAB, this is obtained by the function $\boldsymbol{ifourier}( )$.

$\boldsymbol{f = ifourier(F)}$ is the inverse Fourier transform of the scalar symbolic object F with default independent variable ω. The default return is a function of x. The inverse Fourier transform is applied to a function of ω and returns a function of t.

$\boldsymbol{f = ifourier(F, \mathbf{v}, \mathbf{u})}$ takes F to be a function of v and f to be a function of u instead of the default ω and t, respectively.

```
>>syms w t

>>g= exp(-abs(w))

>>ifourier(g,w,t)
```

returns

```
1/(pi*(t^2 + 1))
```

POST LAB QUESTIONS:

Parseval's relation states that the energy of a signal can be computed by integrating the squared magnitude of the signal either over the time domain or frequency domain. If x(t) and X(f) are a Fourier transform pair, then

$$\text{Energy} = \int_{-\infty}^{\infty} |x(t)|^2 \, dt = \int_{-\infty}^{\infty} |X(f)|^2 \, df$$

Verify the above mentioned Parseval's relation for question given in Exercise 11.1.

EXPERIMENT # 12

LINEARITY AND SCALING PROPERTY OF FOURIER TRANSFROM

OBJECTIVE

- Understanding of linearity and scaling property of Fourier Transform

12.1LINEARITY PROPERTY

The Fourier transform is linear; that is, if

$$\boldsymbol{x_1(t) \overset{F}{\Longleftrightarrow} X_1(\omega)}$$

and

$$\boldsymbol{x_2(t) \overset{F}{\Longleftrightarrow} X_2(\omega)}$$

then

$$\boldsymbol{\alpha x_1(t) + \beta x_2(t) \overset{F}{\Longleftrightarrow} \alpha X_1(\omega) + \beta X_2(\omega)}$$

Define $x_1(t)=\text{rect}(\frac{t}{4})$as a sum of shifted heaviside() functions and $x_2(t)= e^{-2t}u(t)$ in MATLAB. Apply linearity property on these two signals. Suppose $\alpha = 2, \beta = 3$

$$2x_1(t) + 3x_2(t) \overset{F}{\Longleftrightarrow} 2X_1(\omega) + 3X_2(\omega)$$

After you have derived Fourier transform of functions, define 't' for some fixed value e.g. t =[-5:0.01:5] and w=[-2π:0.1:2π], and rewrite the equations to plot the graphs shown in Figure 12.1.

```
syms t
x1=heaviside(t+2)-heaviside(t-2);
x2=exp(-t*2)*heaviside(t);
x=2*x1+3*x2;

X1 =fourier(x1);
X2 =fourier(x2);
X =fourier(x);

t=[-5:0.01:5];
w=[-2*pi:.1:2*pi];
x1=heaviside(t+2)-heaviside(t-2);
x2=exp(-t*2).*heaviside(t);
x=2*x1+3*x2;
X1 =2./w.*sin(2*w);

X2 =1./(2+i*w);
X =4./w.*sin(2*w)+3./(2+i*w);
```

```
subplot(311),plot(t,x,'linewidth',2),grid,title('plot of x(t)=2*x_1 (t)+ 3*x_2
(t)'subplot(312),plot(w,(2*X1+3*X2),'linewidth',2),grid,title('plot of 2*X_1
(w)+ 3*X_2 (w)'
subplot(313),plot(w,(X),'linewidth',2),grid,title('plot of X(w)')
```

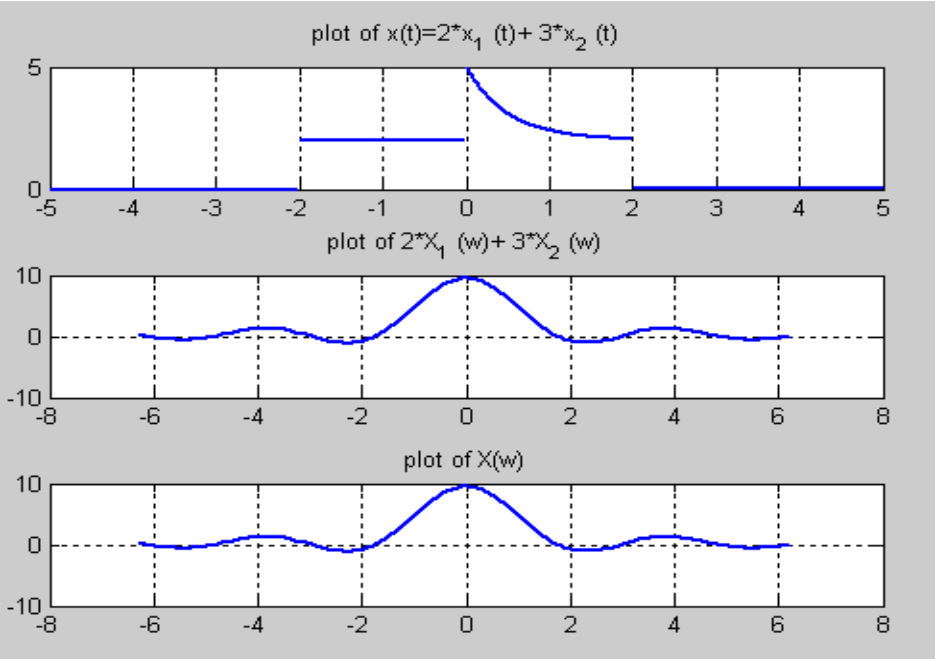


Figure 12.1: Final result

Also, write MATLAB code to compare the magnitude and phase spectra of the right side of the equation i.e. $\boldsymbol{2X_1(\omega) + 3X_2(\omega)}$ to the Fourier transform of the left side i.e. $\boldsymbol{X = F(\,2x_1(t) + 3x_2(t)\,)}$.

The output should be as given in Figure 12.2:

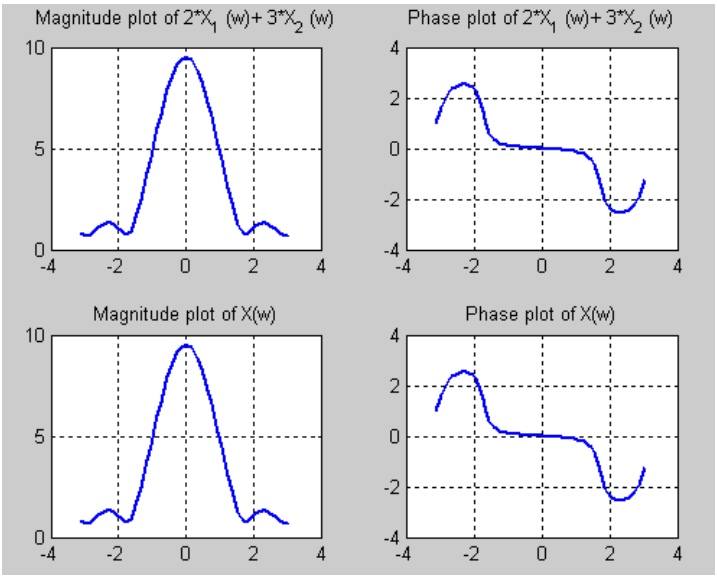


Figure 12.2: Final result

11.2THE SCALING PROPERTY

The scaling property states that time compression of a signal results in its spectral expansion, and time expansion of the signal results in its spectral compression.

$$\boldsymbol{x(at) \overset{F}{\Longleftrightarrow} \frac{1}{|a|} \times X\left(\frac{\omega}{a}\right)}$$

The function $x(at)$represents the function $x(t)$compressed by factor "a". Similarly, a function $X\left(\frac{\omega}{a}\right)$ represents the function $X(\omega)$ expanded in frequency by same factor a.

Prove the Time Scaling property for given signal $x(t)$:

$$x(t) = rect\left(\frac{t}{4}\right), \qquad a = 2$$

Output of your code should be as follows Figure 12.3 :

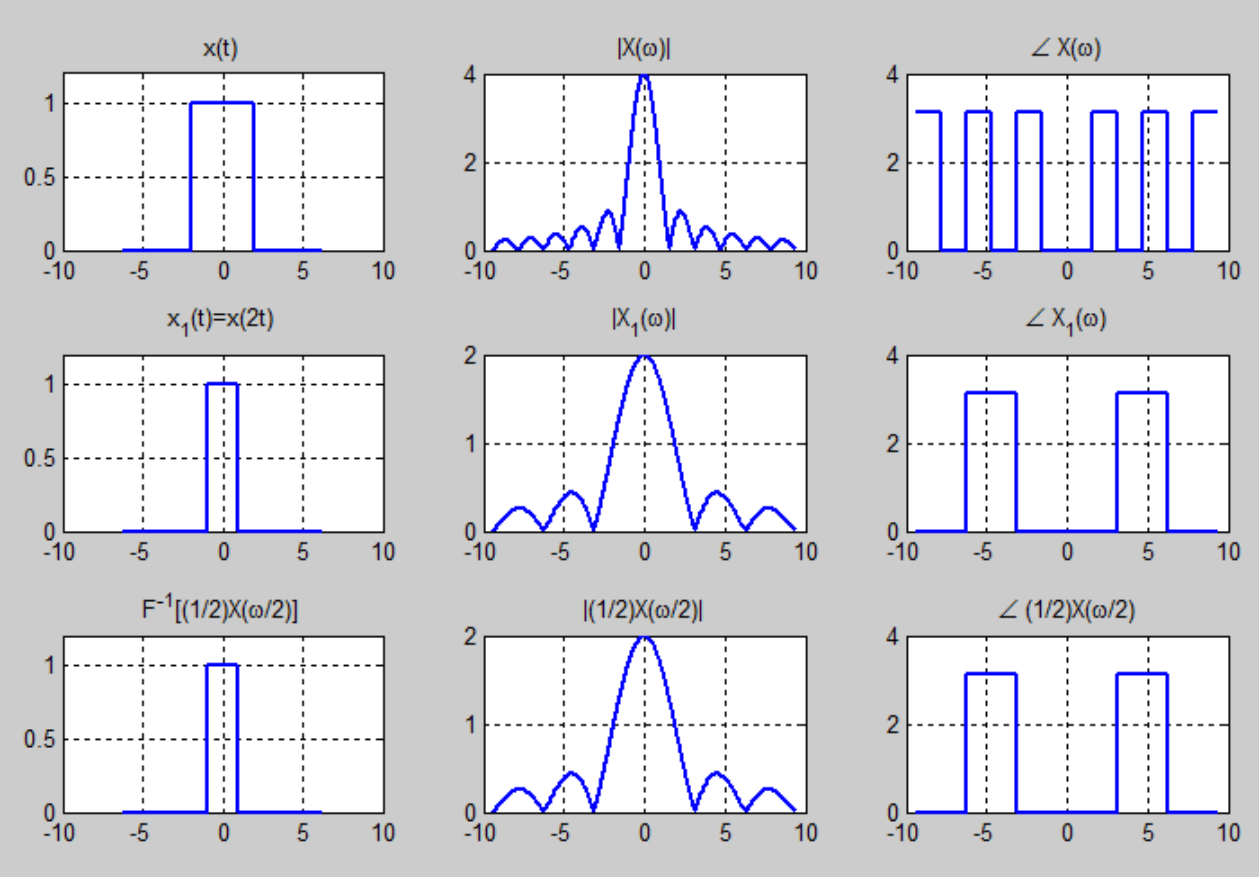


Figure 12.3: Output

POST LAB QUESTIONS:

Q1. Prove the linearity property for the following two functions:

$x_1(t) = \text{sinc}(5t)$ and $x_2(t) = u(t+2)$ with $\alpha = 4$ and $\beta = 3$.

Q2. Prove the scaling property for the following function:

$x(t) = \sin(\pi t)$and use a = 5.

EXPERIMENT # 13

SHIFTING PROPERTIES OF FOURIER TRANSFORM

OBJECTIVE

- Understanding of shifting properties of Fourier Transform

12.1 TIME SHIFTING PROPERTY

$$x(t - t_o) \overset{F}{\Longleftrightarrow} X(\omega)e^{-j\omega t_o}$$

Time shifting property states that delaying a signal by t_0 seconds does not change its amplitude spectrum. The phase spectrum, however, is changed by $-\omega t_0$.

Prove the time shifting property for given $x(t)$

$$x(t) = rect\left(\frac{t}{4}\right), t_o = 2$$

Output of your code should be as following Figure 13.1.

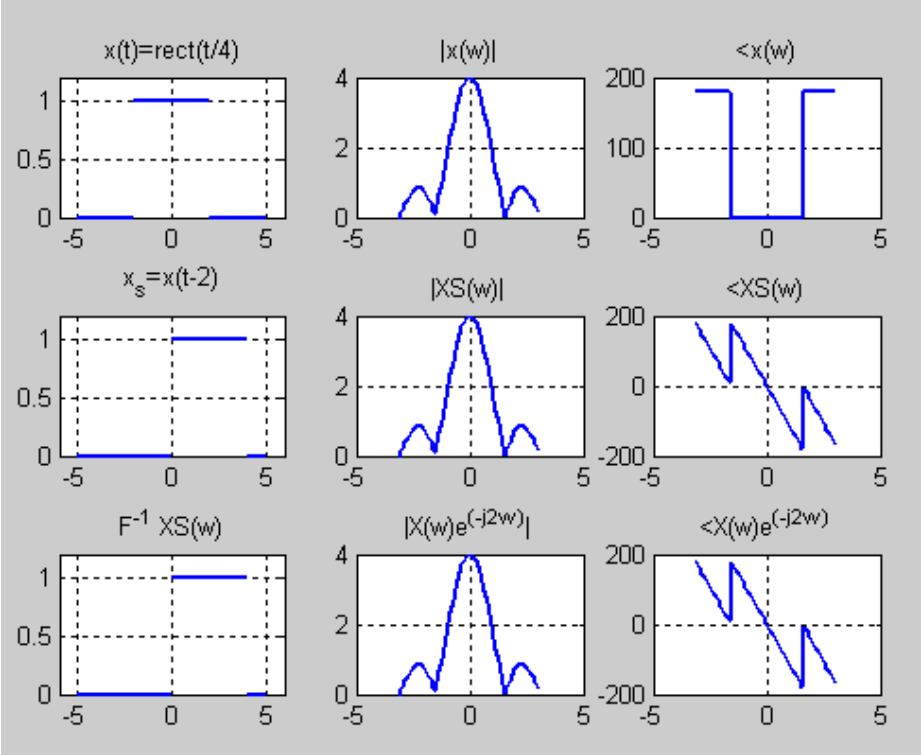


Figure 13.1: Output of code

12.2FREQUENCY SHIFTING PTOPERTY

$$x(t)e^{j\omega_o t} \overset{F}{\Longleftrightarrow} X(\omega - \omega_o)$$

This property states that's that multiplication of a signal by a factor$e^{j\omega_o t}$shifts the spectrum of that signal byω_o.

Prove the Frequency shifting property for given signal:

$$x(t) = rect\left(\frac{t}{4}\right), \omega_o = 2$$

Output of your code should be look like Figure 13.2:

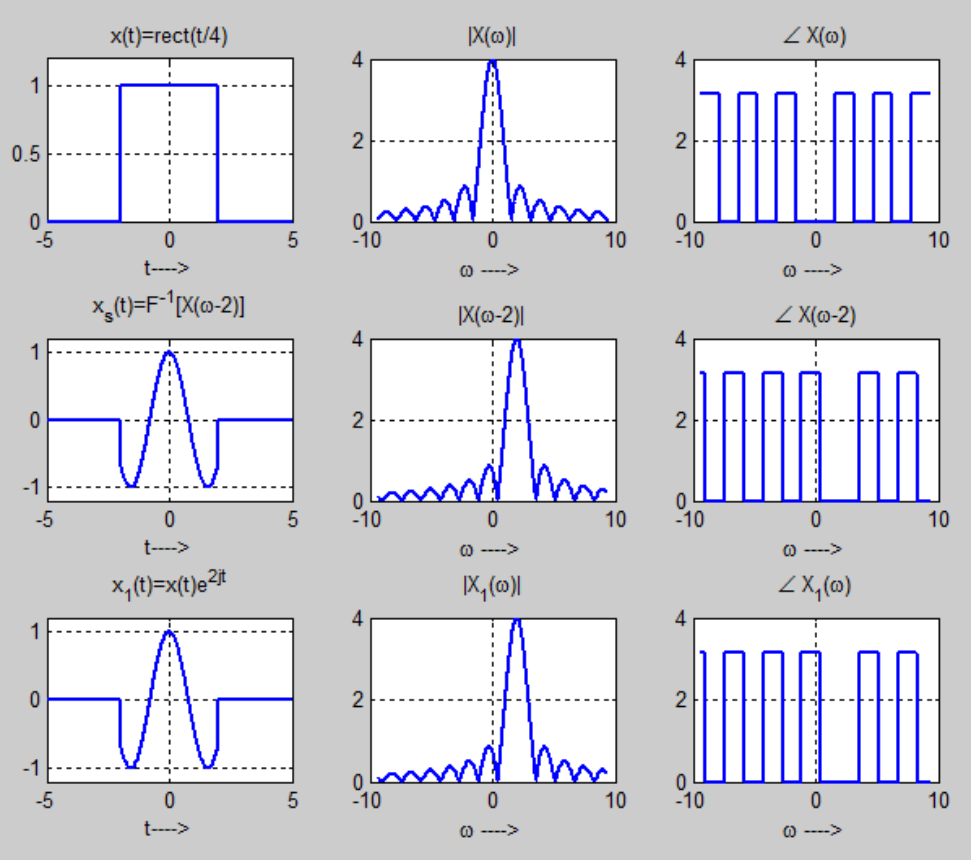


Figure 13.2: Output of code

POST LAB QUESTIONS:

Q.1 The signal in the Figure 13.3 below is a modulated signal with carrier $\cos(10t)$.

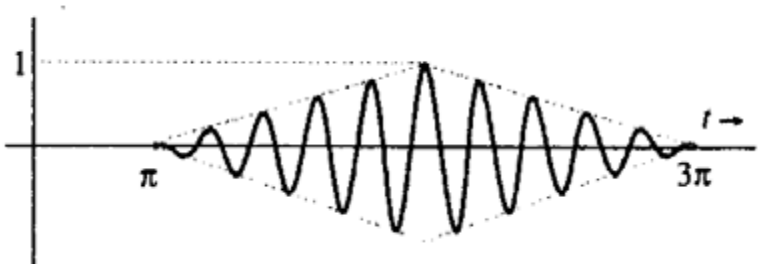


Figure 13.3: Modulated signal

Write MATLAB code to find the Fourier transform of this signal using appropriate properties of the Fourier Transform. Sketch the resulting amplitude and phase spectra.

Q.2 Using the frequency shifting property, write MATLAB code to find the inverse Fourier transform of the spectra shown in the Figure 13.4 below. Plot the resulting time domain signal.

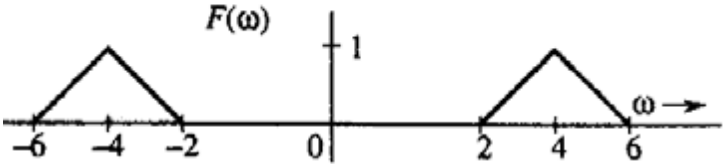


Figure 13.4: inverse Fourier transform

EXPERIMENT # 14

DUALITY AND DIFFERENTIATION PROPERTY OF FOURIER TRANSFORM

OBJECTIVE

- Understanding of duality and differentiation property of Fourier Transform

14.1DUALITY PROPERTY

This property stats that if

$$x(t) \overset{F}{\Longleftrightarrow} X(\omega)$$

Then $$X(t) \overset{F}{\Longleftrightarrow} 2\pi\, x(-\omega)$$

$$rect\left(\frac{t}{\tau}\right) \overset{F}{\Longleftrightarrow} \tau\, sinc\left(\frac{\omega\tau}{2}\right)$$

Prove the Duality property. Given that

a) $x(t) = rect\left(\frac{t}{4}\right)$

$$\tau\, sinc\left(\frac{t\,\tau}{2}\right) \overset{F}{\Longleftrightarrow} 2\pi\, rect\left(\frac{-\omega}{\tau}\right) = 2\pi\, rect\left(\frac{\omega}{\tau}\right)$$

In the above equation we used the fact that $rect(-x) = rect(x)$ because rect is an even function.Output of your code should be as following Figure 14.1

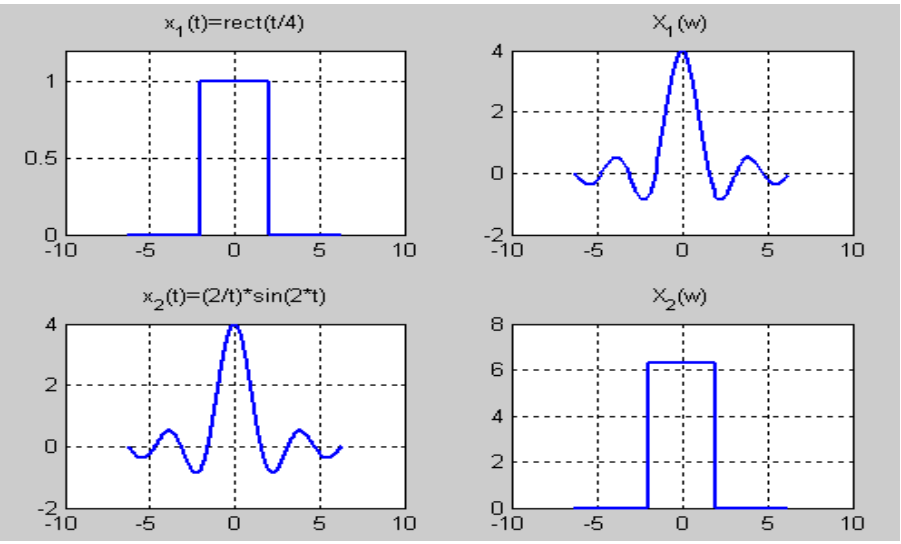


Figure 14.1: Output

b) $\boldsymbol{x(t) = e^{-2t}u(t)}$

Output of your code should be as following Figure 14.2:

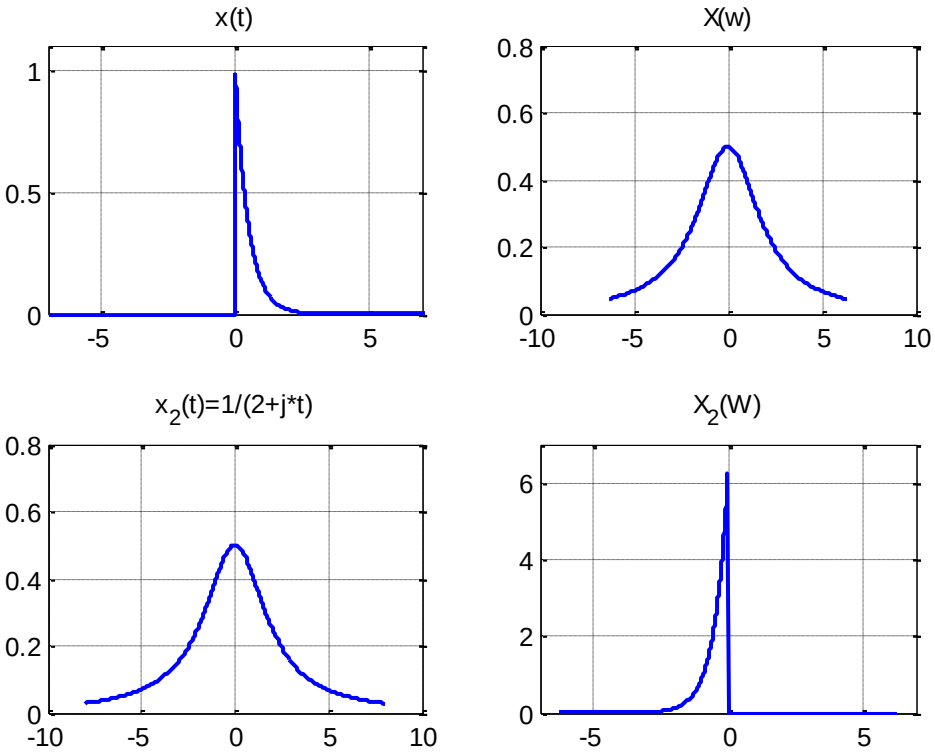


Figure 14.2: Output

14.2 TIME DIFFERENTIATION PROPERTY

If

$$\boldsymbol{x(t) \overset{F}{\Longleftrightarrow} X(\omega)}$$

then

$$\boldsymbol{\frac{d}{dt}\big(x(t)\big) \overset{F}{\Longleftrightarrow} j\omega X(\omega)}$$

This property states that taking derivative of a signal multiplies the spectrum of that signal by $\boldsymbol{j\omega}$.

Prove the Time differentiation property for given x(t)

$$x(t) = 2 * tri\left(\frac{t}{4}\right)$$

Output of your code should be as following Figure 14.3:

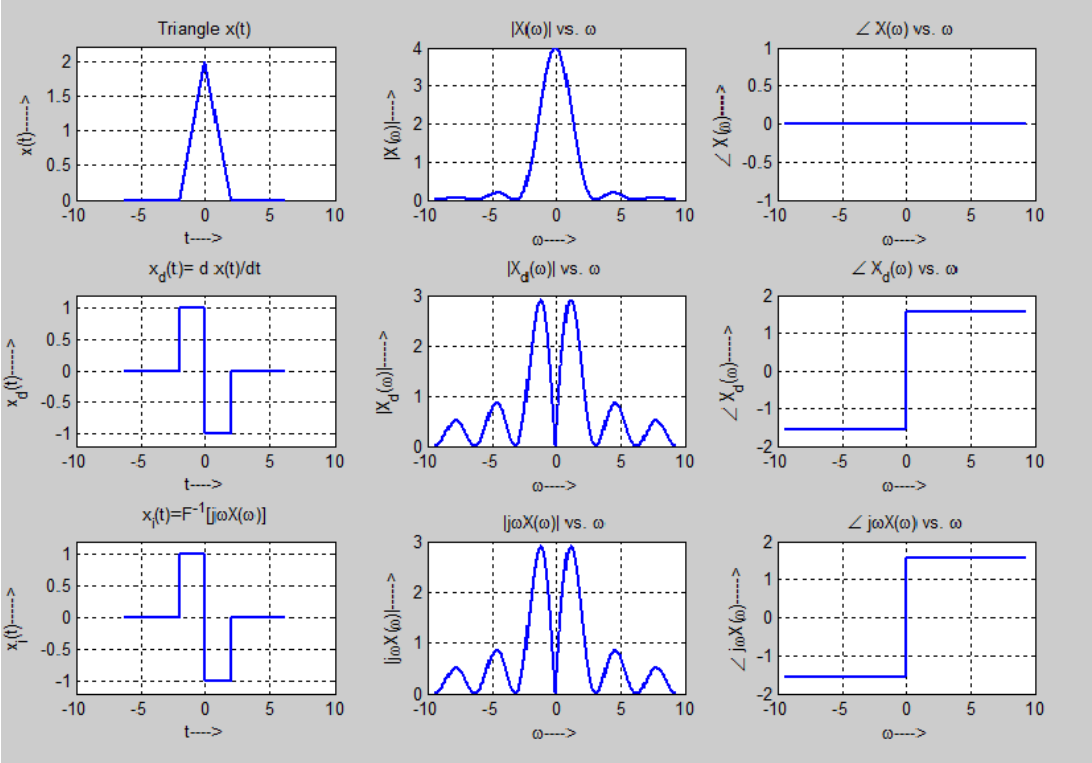


Figure 14.3: Output of code

POST LAB QUESTIONS:

Q1. Using duality property, determine the Fourier Transform of:

i. $x(t) = u(t-5) * e^{-5t}$
ii. g(t) = rect((t-5)/8 τ)

Appendix A: Lab Evaluation Criteria

Labs with projects

1. Experiments and their report 50%
    a. Experiment 60%
    b. Lab report 40%
2. Quizzes (3-4) 15%
3. Final evaluation 35-%
    a. ProjectImplementation 60%
    b. Project report and quiz 40%

Labs without projects

1. Experiments and their report 50%
    a. Experiment 60%
    b. Lab report 40%
2. Quizzes (3-4) 20%
3. Final Evaluation 30%
    i. Experiment. 60%
    ii. Lab report, pre and post experiment quiz 40%

Notice:

Copying and plagiarism of lab reports is a serious academic misconduct. First instance of copying may entail ZERO in that experiment. Second instance of copying may be reported to DC. This may result in awarding FAIL in the lab course.

Appendix B: Safety around Electricity

In all the Electrical Engineering (EE) labs, with an aim to prevent any unforeseen accidents during conduct of lab experiments, following preventive measures and safe practices shall be adopted:

- Remember that the voltage of the electricity and the available electrical current in EE labs has enough power to cause death/injury by electrocution. It is around 50V/10 mA that the "cannot let go" level is reached. "The key to survival is to decrease our exposure to energized circuits."
- If a person touches an energized bare wire or faulty equipment while grounded, electricity will instantly pass through the body to the ground, causing a harmful, potentially fatal, shock.
- Each circuit must be protected by a fuse or circuit breaker that will blow or "trip" when its safe carrying capacity is surpassed. If a fuse blows or circuit breaker trips repeatedly while in normal use (not overloaded), check for shorts and other faults in the line or devices. Do not resume use until the trouble is fixed.
- It is hazardous to overload electrical circuits by using extension cords and multi-plug outlets. Use extension cords only when necessary and make sure they are heavy enough for the job. Avoid creating an "octopus" by inserting several plugs into a multi-plug outlet connected to a single wall outlet. Extension cords should ONLY be used on a temporary basis in situations where fixed wiring is not feasible.
- Dimmed lights, reduced output from heaters and poor monitor pictures are all symptoms of an overloaded circuit. Keep the total load at any one time safely below maximum capacity.
- If wires are exposed, they may cause a shock to a person who comes into contact with them. Cords should not be hung on nails, run over or wrapped around objects, knotted or twisted. This may break the wire or insulation. Short circuits are usually caused by bare wires touching due to breakdown of insulation. Electrical tape or any other kind of tape is not adequate for insulation!
- Electrical cords should be examined visually before use for external defects such as: Fraying (worn out) and exposed wiring, loose parts, deformed or missing parts, damage to outer jacket or insulation, evidence of internal damage such as pinched or crushed outer jacket. If any defects are found the electric cords should be removed from service immediately.
- Pull the plug not the cord. Pulling the cord could break a wire, causing a short circuit.
- Plug your heavy current consuming or any other large appliances into an outlet that is not shared with other appliances. Do not tamper with fuses as this is a potential fire hazard. Do not overload circuits as this may cause the wires to heat and ignite insulation or other combustibles.
- Keep lab equipment properly cleaned and maintained.
- Ensure lamps are free from contact with flammable material. Always use lights bulbs with the recommended wattage for your lamp and equipment.
- Be aware of the odor of burning plastic or wire.

- ALWAYS follow the manufacturer recommendations when using or installing new lab equipment. Wiring installations should always be made by a licensed electrician or other qualified person. All electrical lab equipment should have the label of a testing laboratory.
- Be aware of missing ground prong and outlet cover, pinched wires, damaged casings on electrical outlets.
- Inform Lab engineer / Lab assistant of any failure of safety preventive measures and safe practices as soon you notice it. Be alert and proceed with caution at all times in the laboratory.
- Conduct yourself in a responsible manner at all times in the EE Labs.
- Follow all written and verbal instructions carefully. If you do not understand a direction or part of a procedure, ASK YOUR LAB ENGINEER / LAB ASSISTANT BEFORE PROCEEDING WITH THE ACTIVITY.
- Never work alone in the laboratory. No student may work in EE Labs without the presence of the Lab engineer / Lab assistant.
- Perform only those experiments authorized by your teacher. Carefully follow all instructions, both written and oral. Unauthorized experiments are not allowed.
- Be prepared for your work in the EE Labs. Read all procedures thoroughly before entering the laboratory. Never fool around in the laboratory. Horseplay, practical jokes, and pranks are dangerous and prohibited.
- Always work in a well-ventilated area.
- Observe good housekeeping practices. Work areas should be kept clean and tidy at all times.
- Experiments must be personally monitored at all times. Do not wander around the room, distract other students, startle other students or interfere with the laboratory experiments of others.
- Dress properly during a laboratory activity. Long hair, dangling jewelry, and loose or baggy clothing are a hazard in the laboratory. Long hair must be tied back, and dangling jewelry and baggy clothing must be secured. Shoes must completely cover the foot.
- Know the locations and operating procedures of all safety equipment including fire extinguisher. Know what to do if there is a fire during a lab period; “Turn off equipment, if possible and exit EE lab immediately.”

Appendix C:
Guidelines on Preparing Lab Reports

Each student will maintain a lab notebook for each lab course. He will write a report for each experiment he performs in his notebook. A format has been developed for writing these lab reports.

Programming Stream Lab Report Format

For programming streams, the format of the report will be as given below:

1. **Introduction**: Introduce the new constructs/ commands being used, and their significance.
2. **Objective**: What are the learning goals of the experiment?
3. **Design**: If applicable, draw the flow chart for the program. How do the new constructs facilitate achievement of the objectives; if possible, a comparison in terms of efficacy and computational tractability with the alternate constructs?
4. **Issues**: The bugs encountered and the way they were removed.
5. **Conclusions**: What conclusions can be drawn from experiment?
6. **Application**: Suggest a real world application where this exercise may apply.
7. Answers to post lab questions (if any).

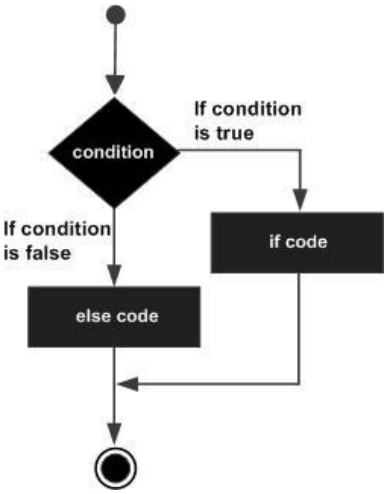
Sample Lab Report for Programming Labs

Introduction

The ability to control the flow of the program, letting it make decisions on what code to execute, is important to the programmer. The if-else statement allows the programmer to control if a program enters a section of code or not based on whether a given condition is true or false. If-else statements control *conditional branching*.

```
if ( expression )
  statement1
else
  statement2
```

If the value of *expression* is nonzero, *statement1* is executed. If the optional **else** is present, *statement2* is executed if the value of *expression* is zero. In this lab, we use this construct to select an action based upon the user's input, or a predefined parameter.



Objective:

To use if-else statements for facilitation of programming objectives: A palindrome is a number or a text phrase that reads the same backward as forward. For example, each of the following five-digit integers is a palindrome: 12321, 55555, 45554 and 11611. We have written a C++ program that reads in a five-digit integer and determines whether it is a palindrome.

Design:

The objective was achieved with the following code:

```
#include<iostream>

usingnamespacestd;
intmain()
{
    inti,temp,d,revrs=0;

    cout<<"enter the number to check :";
        cin>>i;
    temp=i;
    while(temp>0)
    {
        d=temp%10;
        temp/=10;
        revrs=revrs*10+d;

        }
    if(revrs==i)
    cout<<i<<" is palindorme";
    else
    cout<<i<<" is not palindrome";

    }
}
```

Screen shots of the output for various inputs are shown in Figure 1:

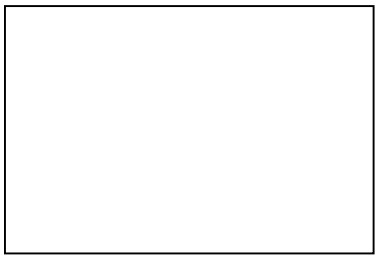


Fig.1. Screen shot of the output

The conditional statement made this implementation possible; without conditional branching, it is not possible to achieve this objective.

Issues:

Encountered bugs and issues; how were they identified and resolved.

Conclusions:

The output indicates correct execution of the code.

Applications:

If-else statements are a basic construct for programming to handle decisions.