CHIRANJEEVI REDDY INSTITUTE OF ENGINEERING & TECHNOLOGY

(Approved by AICTE,New Delhi & Affiliated to JNTU,ANANTAPUR)
Susheela Nagar, Bellary Road, ANANTAPUR.

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING.

BASIC SIMULATION LAB
(9A04305)

(II B.Tech I Semester)

LAB-MANUAL

Head of the Department

S.Raghavendra Swami
M-Tech.,
Assistant professor in ECE.

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR

ELECTRONICS AND COMMUNICATION ENGINEERING

B.Tech II-I Sem. (E.C.E.) Lab **3** **C** **2**

(9A04305) BASIC SIMULATION LAB

List of Experiments:

1. Basic Operations on Matrices
2. Generation of Various Signals and Sequences (periodic & Aperiodic), Such as Unit Impluse, Units step, Square, Saw tooth, Triangular, Sinusoidal, Ramp, Sinc function
3. Operations on signals and Sequences such as Addition, Multiplication, Scaling, Shifting, Folding, Computation of energy and Average Power.
4. Finding Even and Odd Parts of Signal or Sequence and Real And Imaginary Parts of A Signal/Sequence
5. Convolution between signals and Sequences.
6. Auto Correlation and Cross Correlation between signals and Sequences.
7. Verification of Linearity and Time Invariance properties of a given continuous / Discrete system.
8. Computation of Unit Sample, Unit Step, Sinusoidal Responses of given LTI System and verifing its physical realizability and Stability properties.
9. Demonstration of Gibb's Phenomenon
10. finding the Fourier Transform of given signal and ploting its magnitude and phase spectrum.
11. Wave form synthesis using Laplace Transform
12. Locating Poles and Zeros, and ploting the pole zero maps in s-plane and Z-plane for a given Transfer Function
13. Generation of Gaussian Noise(Real & Complex), computastion of its mean, MeanSquare values and its Skew, Kurtosis and PSD, Probability Distribution function
14. Sampling Theorm verification
15. Removal of noise by autocorrelation/ cross correlation in a given signal corrupted by noise.
16. Impulse Response of Raised Cosine Filter
17. Verification of Wiener-Khinchin relations
18. Checking a Random Process for Stationsry in Wide Sense

Additional Experiments:

1. Finding the convolution of a finite length sequence with a long duration signal, using
   A) Overlap Add Method
   B) Overlap Save Method

2. MATLAB program to find frequency response of analog LP/HP/BP/BS filters

Design Experiments:

1. To design FIR filter (LP/HP) using windowing technique Using

   A) Kaiser window

   B) Rectangular window

   C) Triangular window

   D) Hamming window

Using Licensed MatLab of Version 7.0 and Above

1. BASIC OPERATIONS ON MATRICES

AIM:

To understand MATLAB basic features and built in functions available in MATLAB.

EQUIPMENT AND SOFTWARE:

Personal Computer loaded with MATLAB 7.1

OPERATING SYSTEM:

Windows XP

THEORY:

MATLAB, which stands for MATrix LABoratory, is a state-of-the-art mathematical software package, which is used extensively in both academia and industry. It is an interactive program for numerical computation and data visualization, which along with its programming capabilities provides a very useful tool for almost all areas of science and engineering. Unlike other mathematical packages, such as MAPLE or MATHEMATICA, MATLAB cannot perform symbolic manipulations without the use of additional Toolboxes. It remains however, one of the leading software packages for numerical computation.

As you might guess from its name, MATLAB deals mainly with matrices. A scalar is a 1-by-1 matrix and a row vector of length say 5, is a 1-by-5 matrix.. One of the many advantages of MATLAB is the natural notation used. It looks a lot like the notation that you encounter in a linear algebra. This makes the use of the program especially easy and it is what makes MATLAB a natural choice for numerical computations. The purpose of this experiment is to familiarize MATLAB, by introducing the basic features and commands of the program.

MATLAB is case-sensitive, which means that `a + B` is not the same as `a + b`. The MATLAB prompt (») in command window is where the commands are entered.

Matrices :

1. row matrix: Elements in a row are separated either by using white spaces or commas eg: a=[1 2 4 5]
2. column matrix: Elements which differ by a column are seperated by enter or semicolumn eg: b=[1; 2; 3]

Looking into matrix:

a(row,column) allows to look the particular element in the matrix “a”

Vectors:

d= [0:7]

”d” is a vector or row matrix with first element as 0 and last element as 7 and increment is by default 1.

The default increment can be changed (to 0.1) by using increment field in between as e= [0:0.1:7].

d(1:2) allows to look into vector with increment 1

e(1:2:4) look with increment

Operators:

1. + addition
2. - subtraction
3. * multiplication
4. ^ power
5. ' transpose
6. \ left division
7. / right division

Remember that the multiplication, power and division operators can be used in conjunction with a period to specify an element-wise operation.

Typical commands:

1. whos
2. who
3. clear
4. quit
5. save filename – filename.mat
6. load filename - retrive
7. dairy filename - b4 and after ascii text file
8. help

Built in Functions:

1. Scalar Functions:

Certain MATLAB functions are essentially used on scalars, but operate element-wise when applied to a matrix (or vector). They are summarized below.

1. sin - trigonometric sine
2. cos - trigonometric cosine
3. tan - trigonometric tangent
4. asin - trigonometric inverse sine (arcsine)
5. acos - trigonometric inverse cosine (arccosine)
6. atan - trigonometric inverse tangent (arctangent)
7. exp - exponential
8. log - natural logarithm
9. abs - absolute value
10. sqrt - square root
11. rem - remainder
12. round - round towards nearest integer
13. floor - round towards negative infinity
14. ceil - round towards positive infinity

2. Vector Functions:

Other MATLAB functions operate essentially on vectors returning a scalar value. Some of these functions are given below.

1. max largest component : get the row in which the maximum element lies
2. min smallest component
3. length length of a vector
4. sort sort in ascending order
5. sum sum of elements
6. prod product of elements
7. median median value
8. mean mean value std standard deviation

3. Matrix Functions:

Much of MATLAB‟ s power comes from its matrix functions. These can be further separated into two sub-categories.

The first one consists of convenient matrix building functions, some of which are given below.

1. eye - identity matrix
2. zeros - matrix of zeros
3. ones - matrix of ones
4. diag - extract diagonal of a matrix or create diagonal matrices
5. triu - upper triangular part of a matrix
6. tril - lower triangular part of a matrix
7. rand - randomly generated matrix

eg: diag([0.9092;0.5163;0.2661])
ans =

0.9092	0	0
0	0.5163	0
0	0	0.2661

commands in the second sub-category of matrix functions are

1. size size of a matrix
2. det determinant of a square matrix
3. inv inverse of a matrix
4. rank rank of a matrix
5. rref reduced row echelon form
6. eig eigenvalues and eigenvectors
7. poly characteristic polynomial
8. norm norm of matrix (1-norm, 2-norm, ∞ -norm)
9. cond condition number in the 2-norm
10. lu LU factorization
11. qr QR factorization
12. chol Cholesky decomposition
13. svd singular value decomposition

Operations on matrices:

1. Transpose (single quote „)

```
a= [ 1 2 3 ];
b=a‟ ;
```

2. extraction of submatrices

```
a=[1,2;3,4;5,6];
b=a(1:2,2:1;1:1;2:2);
c=a(:,2) ; second column of matrix “a” is sub- matrix “c”
d=a(1,:) ; first row of matrix “a” is sub-matrix “d”
```

3. Determinant and Inverse of a matrix

eg: A = [9,7,0;0,8,6;7,1,-6];

size(A), det(A), inv(A)

We can check our result by verifying that AA–1 = I and A–1A = I .
A*inv(A), inv(A)*A

The eigenvalues and eigenvectors of A (i.e. the numbers λ and vectors x that satisfy Ax = λx) can be obtained through the eig command.

eig(A)

produces a column vector with the eigenvalues

[X,D]=eig(A)

produces a diagonal matrix D with the eigen values on the main diagonal, and a full matrix X whose columns are the corresponding eigenvectors.

RESULT:

MATLAB basic features and built in functions were practiced thoroughly.

2. GENERATION OF VARIOUS SIGNALS AND SEQUENCES

AIM:

To generate various periodic and aperiodic signals and sequences such as Unit Impulse, Unit step, Square, Saw Tooth, Triangular, Sinusoidal, Ramp, Sinc functions using MATLAB.

EQUIPMENT AND SOFTWARE:

Personal Computer loaded with MATLAB 7.1

OPERATING SYSTEM:

Windows XP

THEORY:

If *x* and *y* are two vectors of the same length then `plot(x,y)` plots *x* versus *y*.

For example, to obtain the graph of $y = \cos(x)$ from $-\pi$ to π, we can first define the vector *x* with components equally spaced numbers between $-\pi$ and π, with increment, say 0.01.

```
» x=-pi:0.01:pi;
```

We placed a semicolon at the end of the input line to avoid seeing the (long) output.
Note that the smallest the increment, the "smoother" the curve will be.
Next, we define the vector *y*

```
» y=cos(x);
```

(using a semicolon again) and we ask for the plot

```
» plot(x,y)
```

It is good practice to label the axis on a graph and if applicable indicate what each axis represents. This can be done with the `xlabel` and `ylabel` commands.

```
» xlabel('x')
» ylabel('y=cos(x)')
```

Inside parentheses, and enclosed within single quotes, we type the text that we wish to be displayed along the *x* and *y* axis, respectively. We could even put a title on top using

```
» title('Graph of cosine from -pi to pi')
```

Various line types, plot symbols and colors can be used. If these are not specified (as in the case above) MATLAB will assign (and cycle through) the default ones as given in the table below.

```
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
```

So, to obtain the same graph but in *green*, we type

```
» plot(x,y,'g')
```

where the third argument indicating the color, appears within single quotes. We could get a *dashed* line instead of a *solid* one by typing

```
» plot(x,y,'--')
```

or even a combination of line type and color, say a *blue dotted* line by typing

```
» plot(x,y,'b:')
```

Multiple curves can appear on the same graph. If for example we define another vector

```
» z = sin(x);
```

we can get both graphs on the same axis, distinguished by their line type, using

```
» plot(x,y,'r--',x,z,'b:')
```

When multiple curves appear on the same axis, it is a good idea to create a *legend* to label and distinguish them. The command `legend` does exactly this.

```
» legend('cos(x)','sin(x)')
```

The text that appears within single quotes as input to this command, represents the legend labels. We must be consistent with the ordering of the two curves, so since in the `plot` command we asked for *cosine* to be plotted before *sine*, we must do the same here.

At any point during a MATLAB session, you can obtain a hard copy of the current plot by either issuing the command `print` at the MATLAB prompt, or by using the command menus on the plot window. In addition, MATLAB plots can by *copied* and *pasted* (as pictures) in your favorite word processor (such as Microsoft Word). This can be achieved using the Edit menu on the figure window. Another nice feature that can be used in conjunction with `plot` is the command `grid`, which places grid lines to the current axis (just like you have on graphing paper). Type `help grid` for more information. Other commands for data visualization that exist in MATLAB include `subplot` create an array of (tiled) plots in the same window `loglog` plot using log-log scales `semilogx` plot using log scale on the *x*-axis `semilogy` plot using log scale on the *y*-axis

surf 3-D shaded surface graph
surfl 3-D shaded surface graph with lighting
mesh 3-D mesh surface

Common Sequences: Unit Impulse, Unit Step, and Unit Ramp

Common Periodic Waveforms

The toolbox provides functions for generating widely used periodic waveforms:

sawtooth generates a sawtooth wave with peaks at ±1 and a period of 2π. An optional width parameter specifies a fractional multiple of 2π at which the signal's maximum occurs.

square generates a square wave with a period of 2π. An optional parameter specifies duty cycle, the percent of the period for which the signal is positive.

Common Aperiodic Waveforms

The toolbox also provides functions for generating several widely used aperiodic waveforms:

gauspuls generates a Gaussian-modulated sinusoidal pulse with a specified time, center frequency, and fractional bandwidth. Optional parameters return in-phase and quadrature pulses, the RF signal envelope, and the cutoff time for the trailing pulse envelope.

chirp generates a linear, log, or quadratic swept-frequency cosine signal. An optional parameter specifies alternative sweep methods. An optional parameter phi allows initial phase to be specified in degrees.

The pulstran Function

The pulstran function generates pulse trains from either continuous or sampled prototype pulses. The following example generates a pulse train consisting of the sum of multiple delayed interpolations of a Gaussian pulse.

The Sinc Function

The sinc function computes the mathematical sinc function for an input vector or matrix x. Viewed as a function of time, or space, the sinc function is the inverse Fourier transform of the rectangular pulse in frequency centered at zero of width 2π and height 1. The following equation defines the sinc function:

$$\frac{\sin(\pi x)}{\pi x} = \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{j\omega x}\, d\omega.$$

The sinc function has a value of 1 whenx is equal to zero, and a value of

$$\frac{\sin(\pi x)}{\pi x}$$

for all other elements of x.

The Dirichlet Function

The toolbox function diric computes the Dirichlet function, sometimes called the periodic sinc or aliased sinc function, for an input vector or matrix x. The Dirichlet function is

$$\text{diric}(x) = \begin{cases} \dfrac{\sin(Nx/2)}{N\sin(x/2)} & x \neq 2\pi k, \quad k = 0, \pm 1, \pm 2, \pm 3, \ldots \\ (-1)^{k(N-1)} & x = 2\pi k, \quad k = 0, \pm 1, \pm 2, \pm 3, \ldots \end{cases}$$

where N is a user-specified positive integer. For N odd, the Dirichlet function has a period of 2π; for N even, its period is 4 π. The magnitude of this function is (1/N) times the magnitude of the discrete-time Fourier transform of the N-point rectangular window.

2.1 PROGRAM:

% To generate 1.5 s of a 50 Hz saw tooth wave with a sample rate of 10 kHz and plot 0.2 s of the %generated waveform

% To compute 2 s of a linear chirp signal with a sample rate of 1 kHz, that starts at DC and crosses % 150 Hz at 1 s, use

% The pulse train is defined to have a sample rate of 50 kHz, a pulse train length of 10 ms, and a pulse repetition rate of 1 kHz; `D` specifies the delay to each pulse repetition in column 1 and an optional attenuation for each repetition in column 2. The pulse train is constructed by passing the name of the `gauspuls` function to `pulstran`, along with additional parameters that specify a 10 kHz Gaussian pulse with 50% bandwidth:

% To plot the sinc function for a linearly spaced vector with values ranging from -5 to 5

%To plot the Dirichlet function over the range 0 to 4π for `N = 7` and `N = 8`, use

```
x = linspace(0,4*pi,300);
plot(x,diric(x,7)); axis tight;
plot(x,diric(x,8)); axis tight;
```

Output:

2.2 PROGRAM:

```
% impulse sequence GENERATION

% ramp sequence GENERATION

% exponential sequence GENERATION

% signal generation GENERATION

% SINUSOIDAL SIGNAL GENERATION
```

output:

RESULT: Various sequence were generated using MATLAB functions.

3. OPERATIONS ON SIGNALS AND SEQUENCES

AIM:

To performs functions on signals and sequences such as addition, multiplication, scaling, shifting, folding, computation of energy and average power.

EQUIPMENT AND SOFTWARE:

Personal Computer loaded with MATLAB 7.1

OPERATING SYSTEM:

Windows XP

THEORY:

Signal energy and power:

The energy of a signal is the area under the squared signal.

$$E_f = \int_{-\infty}^{\infty} (|f(t)|)^2 dt$$

Power is a time average of energy (energy per unit time). This is useful when the energy of the signal goes to infinity.

$$P_f = \lim_{T \to \infty} \frac{1}{T} \int_{-(\frac{T}{2})}^{\frac{T}{2}} (|f(t)|)^2 dt$$

Energy vs. Power

1. "Energy signals" have finite energy.
2. "Power signals" have finite and non-zero power.

3.1 Program: Addition, Multiplication, Scaling, Shifting, Folding

OUTPUT SCREEN:

3.2 Program: Computation of Energy

OUTPUT SCREEN:

3.3 Program: Power Calculation

```
%Power calculation of analog and discrete signal
```

OUTPUT SCREEN:

RESULT: various functions on signals and sequences such as addition, multiplication, scaling, shifting, folding, computation of energy and power were performed in MATLAB

4. FINDING EVEN AND ODD PARTS / REAL AND IMAGINARY PARTS OF A SIGNAL/SEQUENCE

AIM:

To Verify Auto and Cross Correlation of Sequences / Signals Using MATLAB

EQUIPMENT AND SOFTWARE:

Personal Computer loaded with MATLAB 7.1

OPERATING SYSTEM:

Windows XP

THEORY:

C is a vector whose elements are the coefficients of the polynomial P and d is the highest degree of the monomials of P.

If m is the length of the vector C, P represents the following Laurent polynomial:

P(z) = C(1)*z^d + C(2)*z^(d-1) + ... + C(m)*z^(d-m+1)

P = laurpoly(C,'dmin',d)

dmin specifies the lowest degree instead of the highest degree of monomials of P. The corresponding output P represents the following Laurent polynomial: P(z) = C(1)*z^(d+m-1) + ... + C(m-1)*z^(d+1) + C(m)*z^d

EVEN - Even part of a Laurent polynomial.
E = EVEN(P) returns the even part E of the Laurent polynomial P.
The polynomial E is such that:
E(z^2) = [P(z) + P(-z)]/2

ODD - Odd part of a Laurent polynomial.
O = ODD(P) returns the odd part O of the Laurent polynomial P.
The polynomial O is such that:
O(z^2) = [P(z) - P(-z)] / [2*z^(-1)]

4.1 PROGRAM: Plot of Real and imaginary part of a complex signal

OUTPUT:

4.2 PROGRAM: Even and odd part of a sequence expressed as Laurent polynomial.

OUTPUT:

4.3: PROGRAM: Even/ odd part of signal

OUTPUT:

4.4 PROGRAM: `Even and odd part of sequence`

OUTPUT:

4.5 PROGRAM: `Real and Imaginary part of signal:`

OUTPUT:

:

RESULT

MATLAB Functions to find Even and odd parts / real and imaginary parts of a signal/sequence were verified.

5. CONVOLUTION

AIM:

To perform linear and Circular Convolution of two Sequences/ Signals.

EQUIPMENT AND SOFTWARE:

Personal Computer loaded with MATLAB 7.1

OPERATING SYSTEM:

Windows XP

THEORY:

Linear Convolution:

Convolution is an integral concatenation of two signals. It is used for the determination of the output signal of a linear time-invariant system by convolving the input signal with the impulse response of the system.

Note that convolving two signals is equivalent to multiplying the Fourier transform of the two signals.

Mathematic Formula:
The linear convolution of two continuous time signals $x(t)$ and $h(t)$ is defined by

$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau)\, h(t-\tau)\, d\tau$$

For discrete time signals $x(n)$ and $h(n)$, the integration is replaced by a summation

$$y(n) = x(n) * h(n) = \sum_{k=-\infty}^{\infty} x(k)\, h(n-k)$$

Circular Convolution:

For the discrete case, multiplication in the frequency domain translates to *circular* convolution in the time domain

5.1 PROGRAM: Linear Convolution

OUTPUT SCREEN:

5.2 Program: Circular Convolution

OUTPUT SCREEN:

5.3 PROGRAM: Convolution between signals

OUTPUT SCREEN:

RESULT: Linear and Circular Convolution of Sequences/signals were verified using MATLAB

6. AUTO CORRELATION AND CROSS CORRELATION

AIM:

To Verify Auto and Cross Correlation of Sequences / Signals Using MATLAB

EQUIPMENT AND SOFTWARE:

Personal Computer loaded with MATLAB 7.1

OPERATING SYSTEM:

Windows XP

THEORY:

The autocorrelation function is defined by the following equation.

$$R(\tau) = \int_{-\infty}^{+\infty} f(t) f(t+\tau) dt$$

6.1 PROGRAM: Cross correlation of sequences:

output screen:

6.2 PROGRAM: Auto Correlation of a signal

output screen:

RESULT: Auto and Cross Correlation of Sequences/ Signal was verified using MATLAB

7. Linearity and Time Invariance

AIM:

To Verify the Linearity and Time Invariance Properties of a given system using MATLAB

EQUIPMENT AND SOFTWARE:

Personal Computer loaded with MATLAB 7.1

OPERATING SYSTEM:

Windows XP

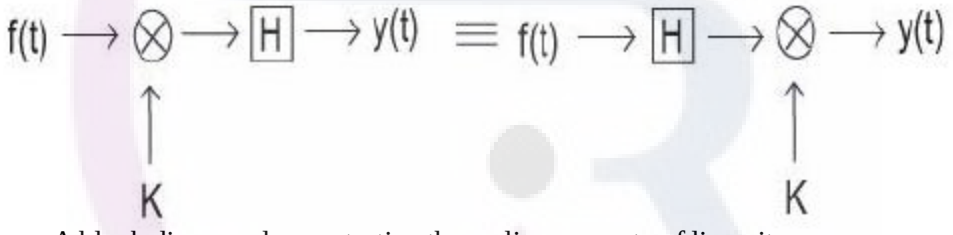
THEORY:

Linear and Non-linear Systems:

A *linear* system is any system that obeys the properties of scaling (homogeneity) and superposition, while a *nonlinear* system is any system that does not obey at least one of these.

To show that a system *H* obeys the scaling property is to show that

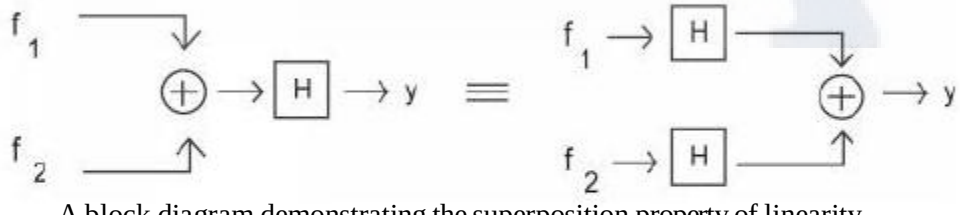
$H(kf(t)) = kH(f(t))$



A block diagram demonstrating the scaling property of linearity

To demonstrate that a system *H* obeys the superposition property of linearity is to show that

$H(f_1(t) + f_2(t)) = H(f_1(t)) + H(f_2(t))$



A block diagram demonstrating the superposition property of linearity

It is possible to check a system for linearity in a single (though larger) step. To do this, simply combine the first two steps to get

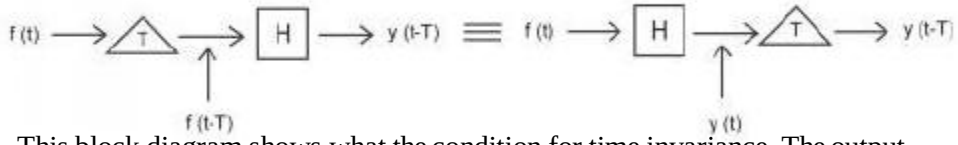
$H(k_1f_1(t) + k_2f_2(t)) = k_2H(f_1(t)) + k_2H(f_2(t))$

Time Invariant vs. Time Variant

A *time invariant* system is one that does not depend on when it occurs: the shape of the output does not change with a delay of the input.

That is to say that for a system H where $H(f(t)) = y(t)$, H is time invariant if for all T

$H(f(t-T)) = y(t-T)$



This block diagram shows what the condition for time invariance. The output is the same whether the delay is put on the input or the output.

When this property does not hold for a system, then it is said to be *time variant*, or time-varying.

7.1: Linearity check

PROGRAM:

```
% linearity property
```

Output:

7.2 Time invariance:

PROGRAM:

```
% Time variancy property
```

Output:

RESULT: Linearity and Time Invariance Properties of a given System was verified.

8. UNIT SAMPLE, UNIT STEP, SINUSOIDAL RESPONSES OF LTI SYSTEM TO VERIFY STABILITY

AIM:

To Compute Unit Sample, Unit Step, Sinusoidal Responses of given LTI system and to verify its Stability using MATLAB

EQUIPMENT AND SOFTWARE:

Personal Computer loaded with MATLAB 7.1

OPERATING SYSTEM:

Windows XP

THEORY:

An LTI system is *stable* if every bounded input produces a bounded output, i.e.,

$$\text{if } |x(t)| < B_x \text{ then } |y(t)| < B_y$$

is true for all . But as the output and input of an LTI is related by convolution, we have:

$$|y(t)| = \left|\int_{-\infty}^{\infty} h(\tau)x(t-\tau)d\tau\right| \le \int_{-\infty}^{\infty} |h(\tau)x(t-\tau)|d\tau$$

$$\le \int_{-\infty}^{\infty} |h(\tau)||x(t-\tau)|d\tau < B_x \int_{-\infty}^{\infty} |h(\tau)|d\tau < B_y$$

i.e., $$\int_{-\infty}^{\infty} |h(\tau)|d\tau < \infty$$

In other words, the LTI system is stable if its impulse response function $h(t)$ is absolutely integrable.

Program:

```
% the transfer function of system is defined by its num     & den

% To find the step response

% To find the impulse response

% To find sinusoidal response

 % simulate time response of LTI system where x is type of     input

 % alternate method to plot system response

 % to find the stability of the system
```

OUTPUT

RESULT:

Unit Sample, Unit Step, Sinusoidal Responses of given LTI system were obtained and its Stability was enquired using MATLAB.

9. DEMONSTRATION OF GIBB'S PHENOMENON

AIM:

To demonstrate Gibbs Phenomenon using MATLAB

EQUIPMENT AND SOFTWARE:

Personal Computer loaded with MATLAB 7.1

OPERATING SYSTEM:

Windows XP

THEORY:

Gibbs Phenomenon:

The peculiar manner in which the Fourier series of a piecewise continuously differentiable periodic function behaves at a jump discontinuity: the *n*th partial sum of the Fourier series has large oscillations near the jump, which might increase the maximum of the partial sum above that of the function itself. The overshoot does not die out as the frequency increases, but approaches a finite limit.

The Gibbs phenomenon involves both the fact that Fourier sums overshoot at a jump discontinuity, and that this overshoot does not die out as the frequency increases.

The best known version of the Gibbs phenomenon is the overshoot that arises when a discontinuous function is represented by a truncated set of Fourier expansion terms. The situation is similar if the truncated Fourier expansion is instead obtained by means of interpolation on an equispaced grid.

PROGRAM:

OUTPUT:

RESULT: Gibbs Phenomenon was verified using MATLAB

10. FOURIER TRANSFORM

AIM:

To obtain Fourier Transform and Inverse Fourier Transform of a given signal / sequence and to plot its Magnitude and Phase Spectra.

EQUIPMENT AND SOFTWARE:

Personal Computer loaded with MATLAB 7.1

OPERATING SYSTEM:

Windows XP

THEORY:

The fast Fourier transform (FFT) is an efficient algorithm for computing the DFT of a sequence; it is not a separate transform. It is particularly useful in areas such as signal and image processing, where its uses range from filtering, convolution, and frequency analysis to power spectrum estimation

For length N input vector x, the DFT is a length N vector X, with elements

$$X(k) = \sum_{n=1}^{N} x(n)*\exp(-j*2*pi*(k-1)*(n-1)/N),\ 1 <= k <= N.$$

The inverse DFT (computed by IFFT) is given by

$$x(n) = (1/N) \sum_{k=1}^{N} X(k)*\exp(j*2*pi*(k-1)*(n-1)/N),\ 1 <= n <= N.$$

10.1: Program: Fourier Transform and Inverse FT of a Signal

output:

10.2 PROGRAM : FFT AND IFFT OF A COMPLEX SIGNAL

output:

RESULT: The Magnitude and Phase Spectra of Fourier Transformed Signal was Plotted

11. LAPLACE TRANSFORM

AIM:

To obtain Laplace and Inverse Laplace Transforms of different functions.

EQUIPMENT AND SOFTWARE:

Personal Computer loaded with MATLAB 7.1

OPERATING SYSTEM:

Windows XP

THEORY:

The Laplace Transform of a function y(t) is defined by

$$Y(s) = L[y(t)](s) = \int_0^\infty e^{-st} y(t)\,dt$$

if the integral exists. The notation L[y(t)](s) means take the Laplace transform of y(t). The functions y(t) and Y(s) are partner functions. Note that Y(s) is indeed only a function of s since the *definite* integral is with respect to t.

Example :

Let y(t)=exp(t). We have

$$Y(s) = \int_0^\infty e^{-st} e^t\,dt = \int_0^\infty e^{-(s-1)t}\,dt = \frac{1}{s-1}$$

The integral converges if s>1. The functions exp(t) and 1/(s-1) are partner functions.

Existence of the Laplace Transform

If y(t) is piecewise continuous for t>=0 and of exponential order, then the Laplace Transform exists for some values of s. A function y(t) is of exponential order c if there is exist constants M and T such that

$$|y(t)| \le Me^{ct} \quad t \ge T$$

All polynomials, simple exponentials (exp(at), where a is a constant), sine and cosine functions, and products of these functions are of exponential order. An example of a function not of exponential order is exp(t^2). This function grows too rapidly. The integral

$$\int_0^\infty e^{-st} e^{t^2}\,dt$$

does not converge for any value of s.

The inverse Laplace transform of the function Y(s) is the unique function y(t) that is continuous on [0,infty) and satisfies L[y(t)](s)=Y(s).

If all possible functions y(t) are discontinous one can select a piecewise continuous function to be the inverse transform.

We will use the notation

$$L^{-1}[Y(s)](t)$$

or Li[Y(s)](t) to denote the inverse Laplace transform of Y(s).

PROGRAM:

```
% finding laplace trasform


 %finding inverse laplace transform
```

OUTPUT:

RESULT : Laplace and inverse laplace transforms were verified using MATLAB functions.

12. POLES AND ZEROS OF A TRANSFER FUNCTION

AIM:

To Plot Pole-Zero map for a given Transfer Function in S- Plane and Z-Planes

EQUIPMENT AND SOFTWARE:

Personal Computer loaded with MATLAB 7.1

OPERATING SYSTEM:

Windows XP

THEORY:

A Transfer Function is the ratio of the output of a system to the input of a system, in the Laplace domain considering its initial conditions to be zero. If we have an input function of X(s), and an output function Y(s), we define the transfer function H(s) to be:

$$H(s) = \frac{Y(s)}{X(s)}$$

transfer function is the Laplace transform of a system‟ s impulse response.

Given a continuous-time transfer function in the Laplace domain, *H*(*s*) or a discrete-time one in the Z-domain, *H*(*z*), a zero is any value of *s* or *z* such that the transfer function is zero, and a pole is any value of *s* or *z* such that the transfer function is infinite.

***Zeros*:** 1. The value(s) for *z* where the *numerator* of the transfer function equals zero

2. The complex frequencies that make the overall gain of the filter transfer function zero.

Poles: 1. The value(s) for *z* where the *denominator* of the transfer function equals zero

2. The complex frequencies that make the overall gain of the filter transfer function infinite.

PROGRAM:

OUTPUT:

RESULT: Poles and Zeros of given transfer function are located and plotted in S- plane and Z- plane

13. GAUSSIAN NOISE

AIM:

To Generate Gaussian Noise and to Compute its Mean, M.S. Values, Skew, kurtosis, PSD and PDF

EQUIPMENT AND SOFTWARE:

Personal Computer loaded with MATLAB 7.1

OPERATING SYSTEM:

Windows XP

THEORY:

Gaussian noise is statistical noise that has a probability density function (abbreviated pdf) of the normal distribution (also known as Gaussian distribution). In other words, the values that the noise can take on are Gaussian-distributed. It is most commonly used as additive white noise to yield additive white Gaussian noise (AWGN).

Gaussian noise is properly defined as the noise with a Gaussian amplitude distribution. This says nothing of the correlation of the noise in time or of the spectral density of the noise.

Labeling Gaussian noise as 'white' describes the correlation of the noise. It is necessary to use the term "white Gaussian noise" to be correct. Gaussian noise is sometimes misunderstood to be white Gaussian noise, but this is not the case.

PROGRAM:

```
% Generate white gaussian noise


% to find mean of x


% to find standard deviation


 % to find skewness


 % to find Kurtosis


 % to find psd of x


 % to find probability distribution function
```

output:

RESULT: Additive White Gaussian noise was generated and its PSD and PDF were plotted and its Mean, Standrad Deviation, Kurtosis, Skew were Computed using MATLAB functions.

14. SAMPLING THEORM

AIM:

To Demonstrate Sampling Theorem and aliasing effect using MATLAB.

EQUIPMENT AND SOFTWARE:

Personal Computer loaded with MATLAB 7.1

OPERATING SYSTEM:

Windows XP

THEORY:

The theorem shows that a band limited analog signal that has been sampled can be perfectly reconstructed from an infinite sequence of samples if the sampling rate exceeds 2*B* samples per second, where *B* is the highest frequency in the original signal.

If a signal contains a component at exactly *B* hertz, then samples spaced at exactly 1/(2*B*) seconds do not completely determine the signal, Shannon's statement notwithstanding.

PROGRAM:

OUTPUT:

RESULT: Sampling theorem was verified and aliasing was identified when the sampling frequency is less than the nyquist frequency using MATLAB.

15. EXTRACTING PERIODIC SIGNAL FROM NOISE

AIM:

To generate a periodic sequence, corrupt it with zero mean White noise and extract the sequence using periodic circular cross correlation.

EQUIPMENT AND SOFTWARE:

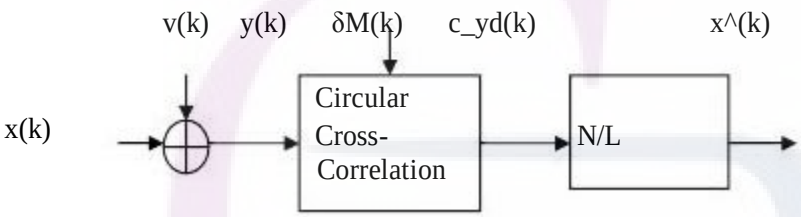
Personal Computer loaded with MATLAB 7.1

OPERATING SYSTEM:

Windows XP

THEORY:

Removal of noise by Filtering is not the solution because noise contains power at all frequencies. Hence we use correlation techniques.



PROGRAM:

```
% Extracting a periodic signal from noise

 % Construct signals x and y


 % periodic signal which include %two sin components


 % generation of N-point Periodic impulse train with period M…such that
 % M<<N


% Compute cross-correlation with periodic pulse train % fast cross correlation.


% Plot portions of x and rho
```

OUTPUT:

RESULT: using Cross- correlation the periodic signal from noise was estimated using MATLAB.

16. IMPULSE RESPONSE OF RAISED COSINE FILTER

AIM:

To Design an FIR Raised Cosine Filter and to plot its Magnitude, Phase and Impulse responses using MATLAB.

EQUIPMENT AND SOFTWARE:

Personal Computer loaded with MATLAB 7.1

OPERATING SYSTEM:

Windows XP

THEORY:

The ideal raised cosine filter frequency response consists of unity gain at low frequencies, a raised cosine function in the middle, and total attenuation at high frequencies.

The width of the middle frequencies are defined by the roll off factor constant Alpha, (0<Altpha<=1). In Filter Solutions, the pass band frequency is defined as the 50% signal attenuation point. The group delay must remain constant at least out to 15 to 20 dB of attenuation.

When the pass band frequency of a raised cosine filter is set to half the data rate, then the impulse response Nyquist's first criteria is satisfied in that the impulse response is zero for T = NTs, where N is an integer, and T is the data period.

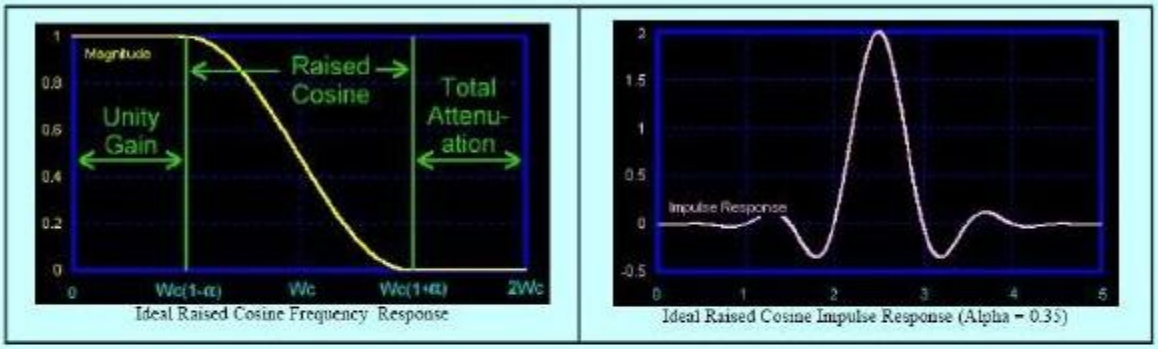
Filter Solutions provides analog, IIR and FIR raised cosine filters. FIR filters are the most accurate and are best to use. However, if it is not possible to use an FIR filter, analog filters may approximate the raised cosine response. The higher the order of the filter, the greater the raised cosine approximation. High order raised cosine filters also produce longer time delays. The lower alpha values use less bandwidth, however, they also produce more ISI due to element value errors and design imperfections.

Mathematically, the frequency response may be written as:

$$F(\omega)=\begin{cases}1 & \text{For } \omega < \omega_c(1-\alpha)\\ 0 & \text{For } \omega > \omega_c(1+\alpha)\\ \dfrac{1+COS\left(\dfrac{\pi(\omega-\omega_c(1-\alpha))}{2\alpha\omega_c}\right)}{2} & \text{For } \omega_c(1-\alpha) < \omega < \omega_c(1+\alpha)\end{cases}$$

Where ω_c is half the data rate in r/s

The ideal raised cosine filter frequency response is shown below:



PROGRAM:

OUTPUT:

RESULT:

The Impulse Response of a Raised Cosine Filter was plotted using fvtool in MATLAB.

17. WIENER-KHINCHIN THEORM

AIM:

To calculate auto-correlation function using Wiener- Khinchin Theorm in MATLAB.

EQUIPMENT AND SOFTWARE:

Personal Computer loaded with MATLAB 7.1

OPERATING SYSTEM:

Windows XP

THEORY:

The **Wiener–Khinchin theorem** (also known as the **Wiener–Khintchine theorem** and sometimes as the **Wiener–Khinchin–Einstein theorem** or the **Khinchin–Kolmogorov theorem**) states that the power spectral density of a wide-sense-stationary random process is the Fourier transform of the corresponding autocorrelation function.

Continuous case:

$$S_{xx}(f) = \int_{-\infty}^{\infty} r_{xx}(\tau) e^{-j2\pi f\tau}\, d\tau$$

where

$$r_{xx}(\tau) = \mathrm{E}\big[\, x(t) x^*(t-\tau) \,\big]$$

is the autocorrelation function defined in terms of statistical expectation, and where $S_{xx}(f)$ is the power spectral density of the function $x(t)$. Note that the autocorrelation function is defined in terms of the expected value of a product, and that the Fourier transform of $x(t)$ does not exist in general, because stationary random functions are not square integrable.

The asterisk denotes complex conjugate, and can be omitted if the random process is real-valued.

Discrete case:

$$S_{xx}(f) = \sum_{k=-\infty}^{\infty} r_{xx}[k] e^{-j2\pi kf}$$

where

$$r_{xx}[k] = \mathrm{E}\big[\, x[n] x^*[n-k] \,\big]$$

and where $S_{xx}(f)$ is the power spectral density of the function with discrete values $x[n]$. Being a sampled and discrete-time sequence, the spectral density is periodic in the frequency domain.

17.1 PROGRAM:

OUTPUT:

17.2 PROGRAM:

OUTPUT:

RESULT: Calculation of Auto-correlation function using Weiner-Khinchin theorm was verified using MATLAB.

18. WSS OF A RANDOM PROCESS

AIM:

To generate a Random process and to check for its Wide Sense Stationary using MATLAB

EQUIPMENT AND SOFTWARE:

Personal Computer loaded with MATLAB 7.1

OPERATING SYSTEM:

Windows XP

THEORY:

A random process is *wide sense stationary* (WSS) if the mean function and autocorrelation function are invariant to a time shift. In particular, this implies that

$$\mu_X(t) = \mu_X = \text{constant},$$

$$R_{XX}(t, t+\tau) = R_{XX}(\tau) \quad \text{(function only of } \tau\text{)}.$$

All strict sense stationary random processes are also WSS, provided that the mean and autocorrelation function exist. The converse is not true. A WSS process does not necessarily need to be stationary in the strict sense. We refer to a process that is not WSS as non - stationary.

Many of the processes we deal with are WSS and hence have a constant mean function and an autocorrelation function that depends only on a single time variable. Hence, in the remainder of the text, when a process is known to be WSS or if we are assuming it to be WSS, then we will represent its autocorrelation function by $R_{XX}(\tau)$. If a process is non - stationary or if we do not know if the process is WSS, then we will explicitly write the autocorrelation function as a function of two variables, $R_{XX}(t, t + \tau)$.

PROGRAM:

Manual Calculation

1) Consider the random process $X(t)=A\cos(\omega_0 t+\Theta)$ Where "Θ" is real-valued random variable and is uniformly distributed over $[0, \pi]$. Check if the process is wide sense stationary.
2) Consider the random process $X(t)=A\cos(\omega_0 t+\Theta)$ Where "Θ" is real-valued random variable and is uniformly distributed over $[0, 2\pi]$. Check if the process is wide sense stationary.

OUTPUT:

RESULT:

Thus the given random processes are identified whether they are WSS or not through calculating the mean and autocorrelation functions manually and verified these results through MATLAB simulation.

ADDITIONAL EXPERIMENTS

EXP :1(A)	Finding the Convolution of a finite length Sequence with a long duration signal, using Overlap-save method	

AIM: To write a matlab program for finding the convolution of a Finite length sequence using Overlap-add & Overlap-Save methods.

EQUIPMENTS:

PC with Matlab Software

Theory:

Let the length of the input sequence is L_S and the length of the Impulse response is M. In this method input sequence is divided into blocks of data of size $N=L+M-1$. Each block consists of last $(M-1)$ data points of previous block followed by L new data points to form a data sequence of length $N=L+M-1$. For first block of data first $M-1$ points are set of zeros. Thus the blocks of data sequence are

$x_1(n)=\{0,0,0. \ . \ . \ 0,x(0),x(1), \ . \ . \ . \ x(L-1)\}$

$x_2(n)=\{x(L-M+1). \ . \ . \ x(L-1) \ \ ,x(L),x(L+1), \ . \ . \ . \ x(2L-1)\}$

$x_3(n)=\{x(2L-M+1). \ . \ . \ x(2L-1) \ \ ,x(2L),x(2L+1), \ . \ . \ . \ x(3L-1)\}$

and so on

Now the impulse response of the filter is increased in length by appending $L-1$ zeros and an N-point circular convolution of $xi(n)$ and $h(n)$ is computed.

i.e., $y_i(n) = x_i(n) \ \textcircled{N} \ h(n)$

In $y_i(n)$, the first $(M-1)$ points will not agree with the linear convolution of $x_i(n)$ and $h(n)$ because of alising, while the remaining points are indentical to the linear convolution. Hence we disacrd first $M-1$ points of the filtered section $x_i(n) \ \textcircled{N} \ h(n)$. The remaining points from seccessive sections are then abutted to construct the final output.

For Example: Let the total length of the sequence L_s=15 and the length of the impulse response is 3. Let the length of each of block is 5.

Now the input sequence can be divided into blocks as

$x_1(n)=\{0,0,x(0),x(1),x(2)\}$

$x_2(n)=\{x(1),x(2), x(3),x(4),x(5)\}$

$x_3(n)=\{x(4),x(5), x(6),x(7),x(8)\}$

$x_4(n)=\{x(7),x(8), x(9),x(10),x(11)\}$

$x_5(n)=\{x(10),x(11), x(12),x(13),x(14)\}$

$x_6(n)=\{x(13),x(14), 0,0,0\}$

Now we perform 5 point circular convolution of $x_i(n)$ and $h(n)$ by appending two zeros to the sequence $h(n)$. In the output block $y_i(n)$, first $M-1$ points are corrupted and must be discarded.

$$y_1(n)= x_1(n) \textcircled{N} h(n)=\{y_1(0), y_1(1)), y_1(2), y_1(3)), y_1(5)\}$$

$$y_2(n)= x_2(n) \textcircled{N} h(n)=\{y_2(0), y_2(1)), y_2(2), y_2(3)), y_2(5)\}$$

$$y_3(n)= x_3(n) \textcircled{N} h(n)=\{y_3(0), y_3(1)), y_3(2), y_3(3)), y_3(5)\}$$

$$y_4(n)= x_4(n) \textcircled{N} h(n)=\{y_4(0), y_4(1)), y_4(2), y_4(3)), y_4(5)\}$$

$$y_5(n)= x_5(n) \textcircled{N} h(n)=\{y_5(0), y_5(1)), y_5(2), y_5(3)), y_5(5)\}$$

$$y_6(n)= x_6(n) \textcircled{N} h(n)=\{y_6(0), y_6(1)), y_6(2), y_6(3)), y_6(5)\}$$

The ouput blocks are abutted togather to get

$$y(n)= \{ y_1(2), y_1(3), y_1(4), y_2(1), y_2(2), y_2(3), y_2(4) , y_3(2), y_3(3), y_3(4) , y_4(2),$$
$$y_4(3), y_4(4) , y_5(2), y_5(3), y_5(4) , y_6(2), y_6(3), y_6(4)\}$$

PROGRAM:

```
close all;
clc;
clear all;
x=[2,-2,1,0,1,3,2,0,1,1];
h=[1,1,1];
m=length(h);
ls =length(x);
l=3;
N=l+m-1;
f=0;
r=ceil(ls/l);
d=zeros(r,N);
for p=1:r
    for q=1:m-1
        if(p==1)
            d(p,q)=0;
      else
       d(p,q)=d(p-1,q+l);
        end
    end
    for q=m:N
       if(f*l+1+q-m<=ls)
          d(p,q)=x(f*l+1+q-m);
       end
    end
    f=f+1;
end
Intr=zeros(r,N);
for p=1:r
    Intr(p,:)=circularcon(d(p,:),h);
end
y=zeros(1,m+ls-1);
g=0;
for p=1:r
```

```
      for q=m:N
         y(q-(m-1)+g*l)=Intr(p,q);
      end
      g=g+1;
   end
   stem(y)
```

Output:

Overlap save method: y(n)={2,0,1,-1,2,4,6,5,3,2,2,1}

RESULT:

Matlab program for finding the convolution of long duration sequences using overlap save and overlap save method by dividing input sequences (long Sequence) into different blocks and adding them to get result.

EXP :01(B)	Finding the Convolution of a finite length Sequence with a long duration signal, using Overlap-Add Method	

AIM: To write a matlab program for finding the convolution of a Finite length sequence using Overlap-add & Overlap-Save methods.

EQUIPMENTS:

PC with Matlab Software

THEORY:

Let the length of the input sequence is L_S and the length of the impulse response is M. In this method the input sequence is divided into blocks of data of size L and $M-1$ zeros are appended to it make the data size of $L+M-1$.

Thus the data blocks may be represented as

$x_1(n)=\{x(0),x(1), \ldots x(L-1),0,0. \ldots\}$

$x_2(n)=\{x(L),x(L+1). \ldots x(2L-1),0,0 \ldots\}$

$x_3(n)=\{x(2L), x(2L+1). \ldots x(3L-1),0,0, \ldots\}$

Now $L-1$ zeros are added to the impulse response $h(n)$ and N-point Circular convolution is performed. Since each data block is terminated with $M-1$ zeros, the last $M-1$ points from each output block must be overlapped and added to the first $M-1$ points of the succeeding block. Hence this method is called overlap-add method.

Let the output blocks are of the form

$y_1(n)=\{ y_1(0), y_1(1). \ldots y_1(L-1), y_1(L). \ldots y_1(N-1)\}$

$y_2(n)=\{ y_2(0), y_2(1). \ldots y_2(L-1), y_2(L). \ldots y_2(N-1)\}$

$y_3(n)=\{ y_3(0), y_3(1). \ldots y_3(L-1), y_3(L). \ldots y_3(N-1)\}$

The output sequence is

$y(n)= \{ y_1(0), y_1(1). \ldots y_1(L-1), y_2(L)+ y_2(0) \ldots$

$\ldots y_1(N-1)+ y_2(M-1) , y_2(M) \ldots y_2(L)+ y_3(0),$

$y_2(L+1)+y_3(1) \ldots y_3(N-1)\}$

PROGRAM:

```
close all;
clc;
clear all;
x=[3,-1,0,1,3,2,0,1,2,1];
h=[1,1,1];
m=length(h);
ls=length(x);
l=3;
N=l+m-1;
r=ceil(ls/l);
d=zeros(r,N);
f=0;
for p=1:r
    for q=1:l
    if(f*l+q<=ls)
        d(p,q)=x(f*l+q);
    end  end
    f=f+1;
    end
temp=zeros(r,N);
for p=1:r
    temp(p,:)=circularcon(d(p,:),h);
end
y=zeros(1,ls+m-1);
y=temp(1,:);
f=1;
for p=2:r
for q=1:m-1
    y(f*l+q)=y(f*l+q)+temp(p,q);
end
for q=1:l
    y(f*l+m-1+q)=temp(p,q+m-1);
end
f=f+1;
end
```

Output:

Overlap add method: y(n)={2,0,1,-1,2,4,6,5,3,2,2,1}

RESULT:

Matlab program for finding the convolution of long duration sequences using overlap save and overlap save method by dividing input sequences (long Sequence) into different blocks and adding them to get result.

EXP :02(A)	FREQUENCY RESPONSE OF ANALOG LPF (USING ELLIP TYPE FILTER)	

AIM: To write a matlab program for finding the frequency response of analog LPF using ellip type.

EQUIPMENTS:

PC with Matlab Software

THEORY:

A Finite Impulse Response (FIR) filter is a discrete linear time-invariant system whose output is based on the weighted summation of a finite number of past inputs.An FIR transversal filter structure can be obtained directly from the equation for discrete time convolution.

$$y(n) = \sum_{k=0}^{n}(x(k)h(n-k)) \quad ; 0 < n < N - 1$$

In this equation, x(k) and y(n) represent the input to and output from the filter at time n. h(n-k) is the transversal filter coefficients at time n. These coefficients are generated by using FDS (Filter Design Software or Digital filter design package). FIR – filter is a finite impulse response filter. Order of the filter should be specified.

Infinite response is truncated to get finite impulse response. placing a window of finite length does this.

Types of windows available are Rectangular, Barlett, Hamming, Hanning, Blackmann window etc. This FIR filter is an all zero filter.

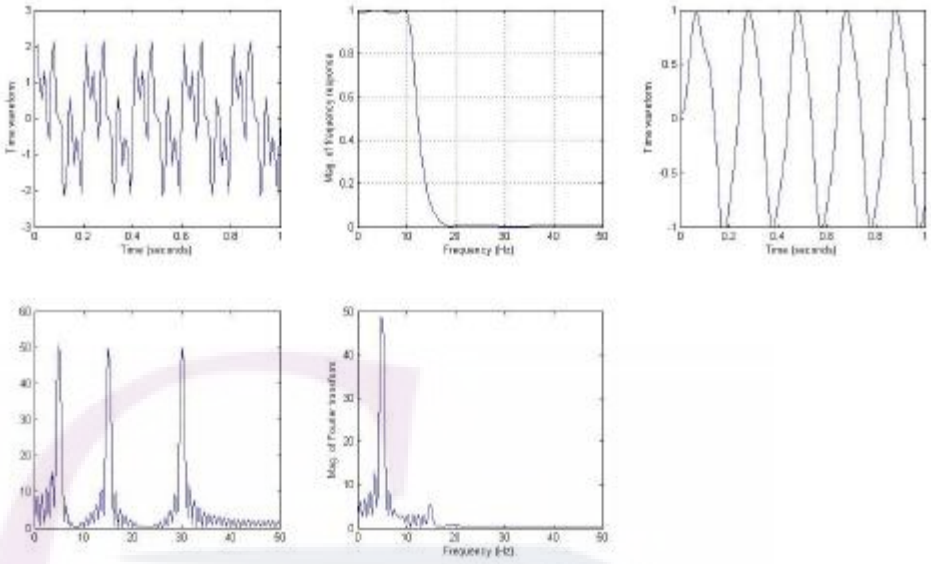
PROGRAM:

```
close all;
clc;
clear all;

Fs = 100;

t = (1:100)/Fs;

s1 = sin(2*pi*t*5); s2=sin(2*pi*t*15); s3=sin(2*pi*t*30);
s = s1+s2+s3;

subplot(2,3,1);plot(t,s);
xlabel('Time (seconds)');
ylabel('Time waveform');

[b,a] = ellip(4,0.1,40,[10]*2/Fs,'low');
[H,w] = freqz(b,a,512);

subplot(2,3,2);
plot(w*Fs/(2*pi),abs(H));
xlabel('Frequency (Hz)'); ylabel('Mag. of frequency response');
grid;

sf = filter(b,a,s);

subplot(2,3,3);
plot(t,sf);
xlabel('Time (seconds)');
ylabel('Time waveform');
axis([0 1 -1 1]);

S = fft(s,512);
SF = fft(sf,512);
w = (0:255)/256*(Fs/2);

subplot(2,3,4);
plot(w,abs(S(1:256)));

subplot(2,3,5);
plot(w,abs(SF(1:256)));
xlabel('Frequency (Hz)'); ylabel('Mag. of Fourier transform');
```

RESULT:

Matlab program for finding the frequency response of analog LPF using ellip type is written.

Waveforms:


Time waveform
Time (seconds)
Mag. of Frequency response
Frequency (Hz)
Time waveform
Time (seconds)
Mag. of Fourier transform
Frequency (Hz)

EXP :02(B)	FREQUENCY RESPONSE OF ANALOG HPF(USING BUTTERWORTH FILTER)	

AIM: To write a matlab program for finding the frequency response of analog HPF using Butterworth type.

EQUIPMENTS:

PC with Matlab Software

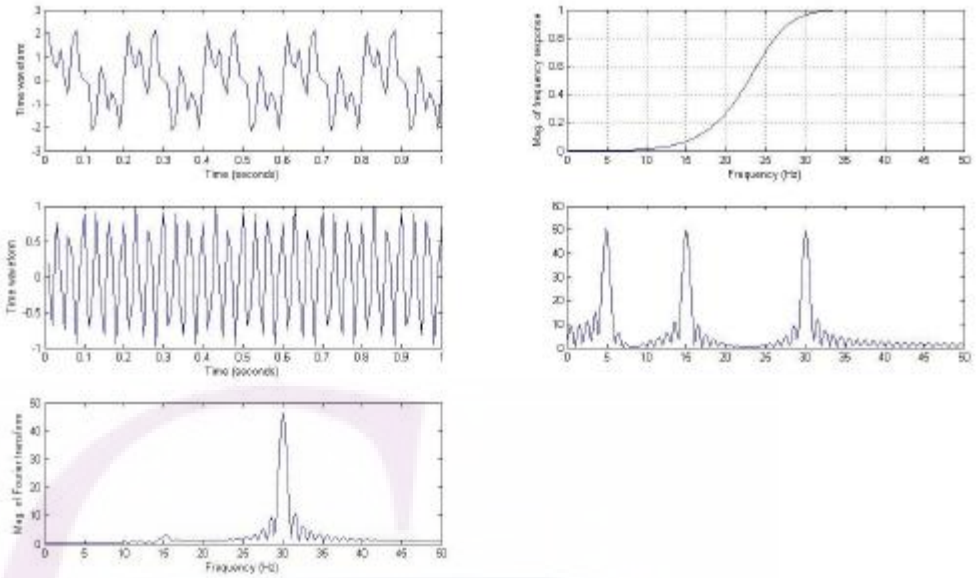
PROGRAM:

```
close all;
clc;
clear all;

Fs = 100;
t = (1:100)/Fs;
s1 = sin(2*pi*t*5); s2=sin(2*pi*t*15); s3=sin(2*pi*t*30);
s = s1+s2+s3;

subplot(3,2,1);plot(t,s);
xlabel('Time (seconds)');
ylabel('Time waveform');

[b,a] = butter(4,[25]*2/Fs,'high');
[H,w] = freqz(b,a,512);

subplot(3,2,2);
plot(w*Fs/(2*pi),abs(H));
xlabel('Frequency (Hz)'); ylabel('Mag. of frequency response');
grid;

sf = filter(b,a,s);

subplot(3,2,3);
plot(t,sf);
xlabel('Time (seconds)');
ylabel('Time waveform');
axis([0 1 -1 1]);

S = fft(s,512);
SF = fft(sf,512);
w = (0:255)/256*(Fs/2);

subplot(3,2,4);
plot(w,abs(S(1:256)));

subplot(3,2,5);
plot(w,abs(SF(1:256)));
xlabel('Frequency (Hz)'); ylabel('Mag. of Fourier transform');
```

RESULT:

Matlab program for finding the frequency response of analog HPF using Butterworth type is written.

Waveforms:


Time waveform
Time (seconds)
Mag. of frequency response
Frequency (Hz)
Time waveform
Time (seconds)
Mag. of Fourier transform
Frequency (Hz)

EXP :02(c)	FREQUENCY RESPONSE OF ANALOG BPF(USING CHEBY1 FILTER)	

AIM: To write Matlab program for finding the frequency response of analog BPF using cheby-1 type.

EQUIPMENTS:

PC with Matlab Software

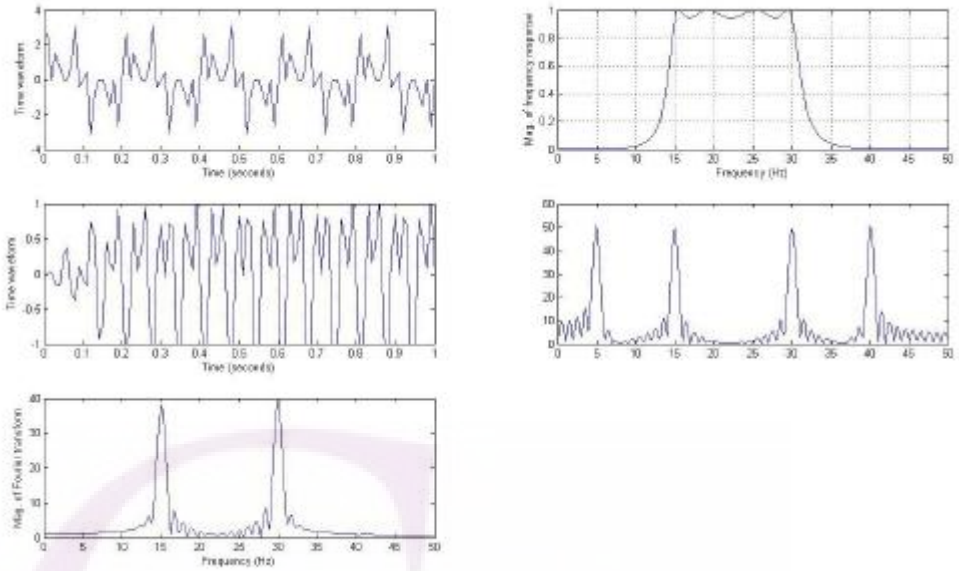
PROGRAM:

```
close all;
clc;
clear all;

Fs = 100;

t = (1:100)/Fs;

s1 = sin(2*pi*t*5);    s2 = sin(2*pi*t*15);
s3 = sin(2*pi*t*30);   s4 = sin(2*pi*t*40);

s = s1+s2+s3+s4;

subplot(3,2,1);plot(t,s);
xlabel('Time (seconds)');
ylabel('Time waveform');

[b,a] = cheby1(4,0.5,[15 30]*2/Fs);
[H,w] = freqz(b,a,512);

subplot(3,2,2);
plot(w*Fs/(2*pi),abs(H));
xlabel('Frequency (Hz)'); ylabel('Mag. of frequency response');
grid;

sf = filter(b,a,s);

subplot(3,2,3);
plot(t,sf);
xlabel('Time (seconds)');
ylabel('Time waveform');
axis([0 1 -1 1]);

S = fft(s,512);
SF = fft(sf,512);
w = (0:255)/256*(Fs/2);

subplot(3,2,4);
plot(w,abs(S(1:256)));
subplot(3,2,5);
plot(w,abs(SF(1:256)));
xlabel('Frequency (Hz)'); ylabel('Mag. of Fourier transform');
```

RESULT:

Matlab program for finding the frequency response of analog BPF using cheby-1 type is written.

Waveforms:


Time waveform
Time (seconds)
Mag. of frequency response
Frequency (Hz)
Time waveform
Time (seconds)
Mag. of Fourier transform
Frequency (Hz)

EXP :02(D)	FREQUENCY RESPONSE OF ANALOG BSF(USING CHEBY2 FILTER)	

AIM: To write Matlab program for finding the frequency response of analog BSF using cheby-2 type.

EQUIPMENTS:

PC with Matlab Software

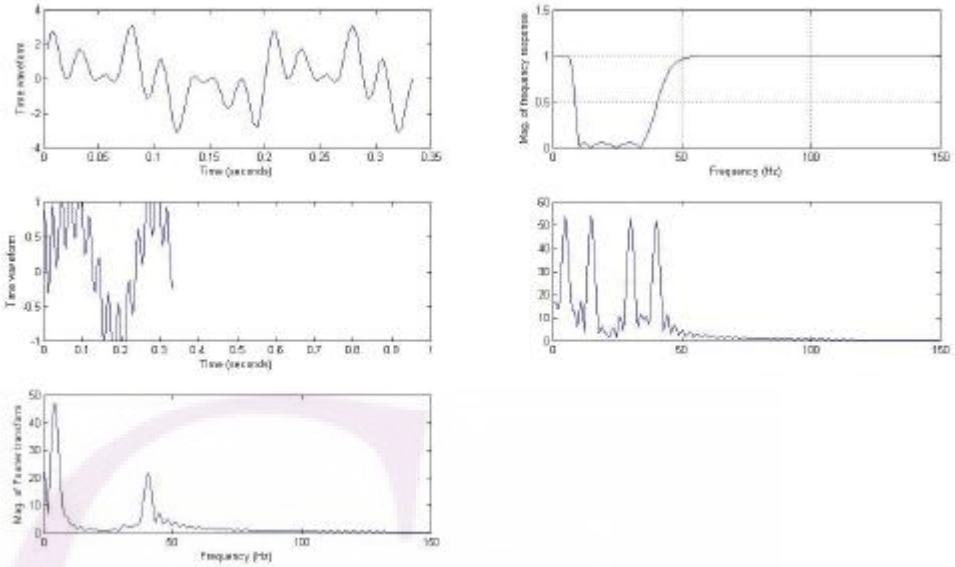
PROGRAM:

```
close all;
clc;
clear all;

Fs = 300;
t = (1:100)/Fs;

s1 = sin(2*pi*t*5);     s2 = sin(2*pi*t*15);
s3 = sin(2*pi*t*30);    s4 = sin(2*pi*t*40);
s = s1+s2+s3+s4;

subplot(3,2,1);plot(t,s);
xlabel('Time (seconds)');
ylabel('Time waveform');

[b,a] = cheby2(4,25,[10 35]*2/Fs,'stop');
[H,w] = freqz(b,a,512);

subplot(3,2,2);
plot(w*Fs/(2*pi),abs(H));
xlabel('Frequency (Hz)'); ylabel('Mag. of frequency response');
grid;

sf = filter(b,a,s);

subplot(3,2,3);
plot(t,sf);
xlabel('Time (seconds)');
ylabel('Time waveform');
axis([0 1 -1 1]);

S = fft(s,512);
SF = fft(sf,512);
w = (0:255)/256*(Fs/2);

subplot(3,2,4);
plot(w,abs(S(1:256)));

subplot(3,2,5);
plot(w,abs(SF(1:256)));
xlabel('Frequency (Hz)'); ylabel('Mag. of Fourier transform');
```

RESULT:

Matlab program for finding the frequency response of analog BSF using cheby-2 type is written.

Waveforms:


Time waveform
Time (seconds)
Mag. of frequency response
Frequency (Hz)
Time waveform
Time (seconds)
Mag. of Fourier transform
Frequency (Hz)

DESIGN EXPERIMENTS

EXP: 1 FREQUENCY RESPONSE OF FIR FILTER USING WINDOW TECHNIQUE

Theory:

FIR filters are filters having a transfer function of a polynomial in z^{-} and is an all-zero filter in the sense that the zeroes in the z-plane determine the frequency response magnitude characteristic. The z transform of a N-point FIR filter is given by

$$H(z) = \sum_{n=0}^{N-1} h(n) z^{-n} \tag{1}$$

FIR filters are particularly useful for applications where exact linear phase response is required. The FIR filter is generally implemented in a non-recursive way which guarantees a stable filter. FIR filter design essentially consists of two parts
(i) approximation problem
(ii) realization problem

The approximation stage takes the specification and gives a transfer function through four steps. They are as follows:
(i) A desired or ideal response is chosen, usually in the frequency domain.
(ii) An allowed class of filters is chosen (e.g. the length N for a FIR filters).
(iii) A measure of the quality of approximation is chosen.
(iv) A method or algorithm is selected to find the best filter transfer function.

The realization part deals with choosing the structure to implement the transfer function which may be in the form of circuit diagram or in the form of a program.

There are essentially three well-known methods for FIR filter design namely:

(1) The window method
(2) The frequency sampling technique
(3) Optimal filter design methods

The Window Method

In this method, The desired frequency response specification $H_d(w)$, corresponding unit sample response $h_d(n)$ is determined using the following relation

$$h_d(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H_d(w) e^{jwn} dw \tag{2}$$

where

$$H_d(w) = \sum_{n=-\infty}^{\infty} h_d(n) e^{-jwn} \tag{3}$$

In general, unit sample response $h_d(n)$ obtained from the above relation is infinite in duration, so it must be truncated at some point say $n = M$-1 to yield an FIR filter of length M (i.e. 0 to M-1). This truncation of $h_d(n)$ to length M-1 is same as multiplying $h_d(n)$ by the rectangular window defined as

$$w(n) = \begin{cases} 1 & 0 \le n \le M- \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Thus the unit sample response of the FIR filter becomes

$$\begin{aligned} h(n) &= h_d(n)\, w(n) \\ &= h_d(n) \qquad 0 \le n \le M\text{-}1 \\ &= 0 \qquad \text{otherwise} \end{aligned} \tag{5}$$

Now, the multiplication of the window function $w(n)$ with $h_d(n)$ is equivalent to convolution of $H_d(w)$ with $W(w)$, where $W(w)$ is the frequency domain representation of the window function

$$W(w) = \sum_{n=0}^{M-1} w(n)e^{-jwn} \tag{6}$$

Thus the convolution of $H_d(w)$ with $W(w)$ yields the frequency response of the truncated FIR filter

$$H(w) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H_d(v)W(w-v)dw \tag{7}$$

The frequency response can also be obtained using the following relation

$$H(w) = \sum_{n=0}^{M-1} h(n)e^{-jwn} \tag{8}$$

But direct truncation of $h_d(n)$ to M terms to obtain $h(n)$ leads to the Gibbs phenomenon effect which manifests itself as a fixed percentage overshoot and ripple before and after an approximated discontinuity in the frequency response due to the non-uniform convergence of the fourier series at a discontinuity.Thus the frequency response obtained by using (8) contains ripples in the frequency domain. In order to reduce the ripples, instead of multiplying $h_d(n)$ with a rectangular window $w(n)$, $h_d(n)$ is multiplied with a window function that contains a taper and decays toward zero gradually, instead of abruptly as it occurs in a rectangular window. As multiplication of sequences $h_d(n)$ and $w(n)$ in time domain is equivalent to convolution of $H_d(w)$ and $W(w)$ in the frequency domain, it has the effect of smoothing $H_d(w)$.

The several effects of windowing the Fourier coefficients of the filter on the result of the frequency response of the filter are as follows:

(i) A major effect is that discontinuities in $H(w)$ become transition bands between values on either side of the discontinuity.

(ii) The width of the transition bands depends on the width of the main lobe of the frequency response of the window function, $w(n)$ i.e. $W(w)$.

(iii) Since the filter frequency response is obtained via a convolution relation , it is clear that the resulting filters are never optimal in any sense.

(iv) As M (the length of the window function) increases, the mainlobe width of $W(w)$ is reduced which reduces the width of the transition band, but this also introduces more ripple in the frequency response.

(v) The window function eliminates the ringing effects at the bandedge and does result in lower sidelobes at the expense of an increase in the width of the transition band of the filter.

Some of the windows commonly used are as follows:

1. Bartlett triangular window:

$$W(n) = \frac{2(n+1)}{N+1} \qquad n = 0,1,2,\ldots\ldots,(N\text{-}1)/2 \tag{9}$$

$$= 2 - \frac{2(n+1)}{N+1} \qquad n = (N\text{-}1)/2,\ldots\ldots,N\text{-}1$$

$$= 0 , \qquad \text{otherwise}$$

2-5. Generalized cosine windows
(Rectangular, Hanning, Hamming and Blackman)

$$W(n) = a - b\cos(2p(n+1)/(N+1)) + c\cos(4p(n+1)/(N+1)) \quad n = 0,1\ldots.N\text{-}1 \qquad (10)$$
$$= 0 \qquad \text{otherwise}$$

6. Kaiser window with parameter ß :

$$W(n) = \frac{I_o(\beta\sqrt{1-(2(n+1)/(N+1))^2})}{I_o(\beta)} \qquad n = 0,1,\ldots,N\text{-}1 \qquad (11)$$
$$= 0 \qquad \text{otherwise}$$

The general cosine window has four special forms that are commonly used. These are determined by the parameters *a,b,c*

TABLE I
Value of coefficients for a,b and c

Window	a	b	c
Rectangular	1	0	0
Hanning	0.5	0.5	0
Hamming	0.54	0.46	0
Blackman	0.42	0.5	0.08

The Bartlett window reduces the overshoot in the designed filter but spreads the transition region considerably. The Hanning, Hamming and Blackman windows use progressively more complicated cosine functions to provide a smooth truncation of the ideal impulse response and a frequency response that looks better. The best window results probably come from using the Kaiser window, which has a parameter ß that allows adjustment of the compromise between the overshoot reduction and transition region width spreading.

The major advantages of using window method is their relative simplicity as compared to other methods and ease of use. The fact that well defined equations are often available for calculating the window coefficients has made this method successful.

There are following problems in filter design using window method:

(i) This method is applicable only if $H_d(w)$ is absolutely integrable i.e only if (2) can be evaluated. When $H_d(w)$ is complicated or cannot easily be put into a closed form mathematical expression, evaluation of $h_d(n)$ becomes difficult.

(ii) The use of windows offers very little design flexibility e.g. in low pass filter design, the passband edge frequency generally cannot be specified exactly since the window smears the discontinuity in frequency. Thus the ideal LPF with cut-off frequency fc, is smeared by the window to give a frequency response with passband response with passband cutoff frequency f_1 and stopband cut-off frequency f_2.

(iii) Window method is basically useful for design of prototype filters like lowpass,highpass,bandpass etc. This makes its use in speech and image processing applications very limited.

EXP :1a(1)	FREQUENCY RESPONSE OF FIR FILTER (USING KAISER WINDOW)	

AIM: To write a matlab program to find the frequency response of LPF using Kaiser window.

EQUIPMENTS:

PC with Matlab Software

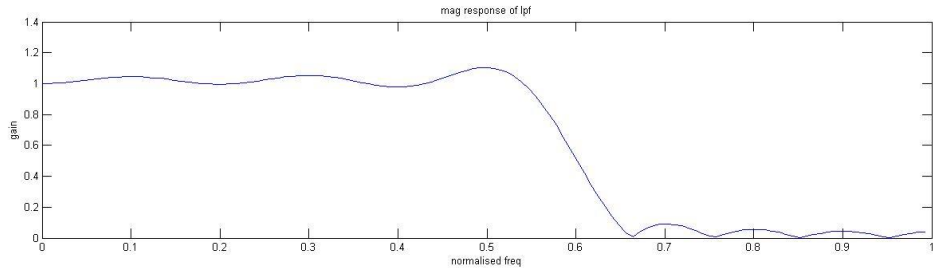
PROGRAM:

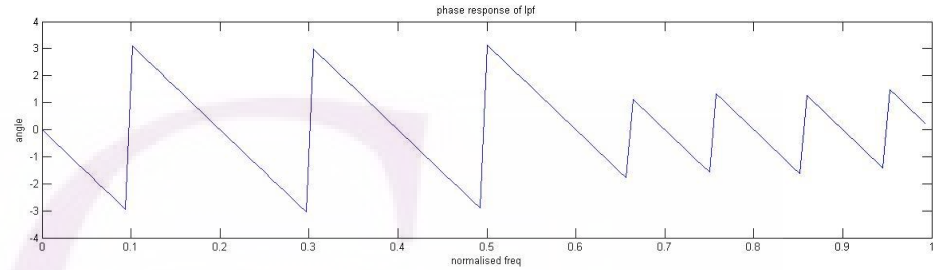
```
clear all;
close all;
clc;

n=20;
fp=300;
fs=1000;
fn=2*fp/fs;

window=kaiser(n+1);

b=fir1(n,fn,'low',window);
[h,w]=freqz(b,1,128);

subplot(2,1,1);
plot(w/pi,abs(h));
title('mag response of lpf');
ylabel('gain');
xlabel('normalised freq');

subplot(2,1,2);
plot(w/pi,angle(h));
title('phase response of lpf');
ylabel('angle');
xlabel('normalised freq');
```

RESULT:

Matlab program to find the frequency response of LPF using Kaiser window is written.

Waveforms:


mag response of lpf
gain
normalised freq
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0.2
0.4
0.6
0.8
1
1.2
1.4


phase response of lpf
angle
normalised freq
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
-4
-3
-2
-1
1
2
3
4

EXP :1a(2)	FREQUENCY RESPONSE OF HPF USING KAISER WINDOW	

AIM: To write a Matlab program to find the frequency response of HPF using Kaiser window.

EQUIPMENTS:

PC with Matlab Software

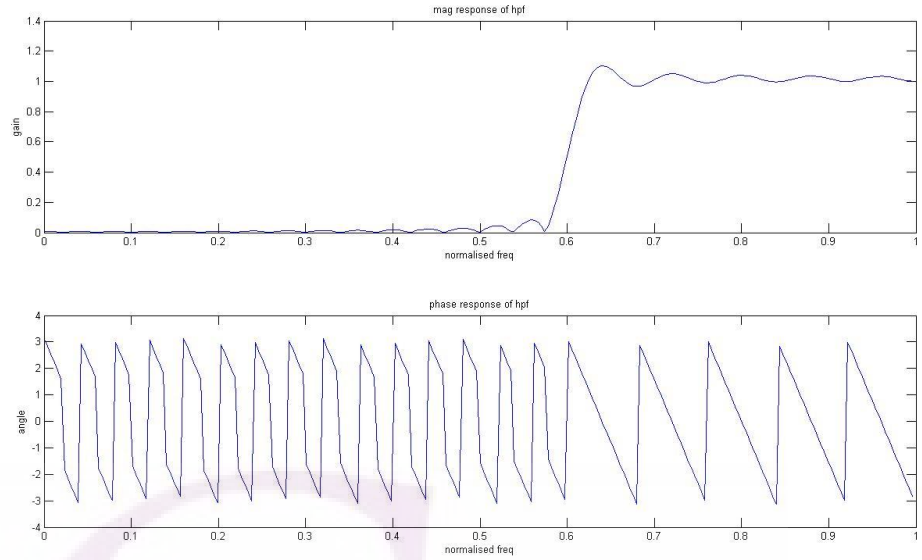
PROGRAM:

```
clear all;
close all;
clc;

n=50;
fp=300;
fs=1000;
fn=2*fp/fs;

window=kaiser(n+1);

b=fir1(n,fn,'high',window);
[h,w]=freqz(b,1,256);

subplot(2,1,1);
plot(w/pi,abs(h));
title('mag response of hpf');
ylabel('gain');
xlabel('normalised freq');

subplot(2,1,2);
plot(w/pi,angle(h));
title('phase response of hpf');
ylabel('angle');
xlabel('normalised freq');
```

RESULT:

Matlab program to find the frequency response of HPF using Kaiser window is written.

Waveforms:


mag response of hpf
gain
normalised freq
phase response of hpf
angle
normalised freq

EXP :1a(3)	FREQUENCY RESPONSE OF BPF USING KAISER WINDOW	

AIM: To write a Matlab program to find the frequency response of BPF using Kaiser window.

EQUIPMENTS:

PC with Matlab Software

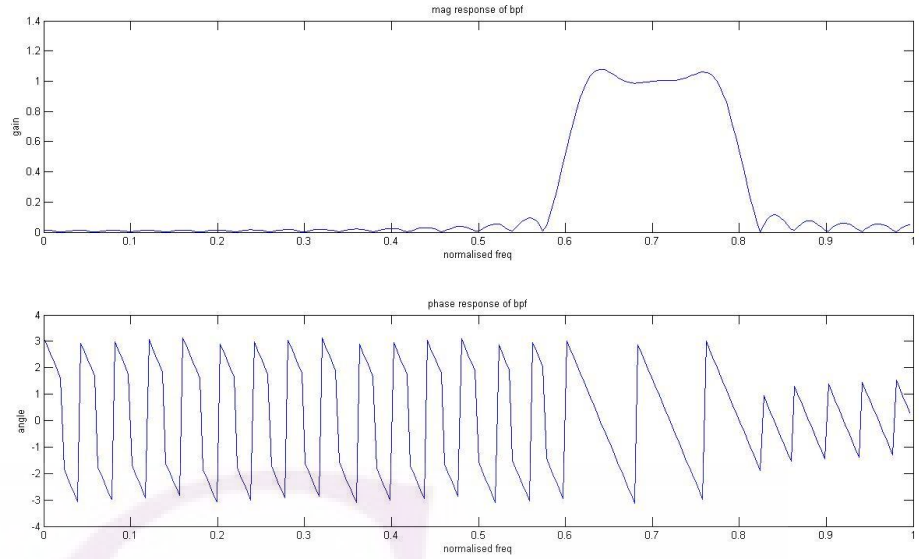
PROGRAM:

```
clear all;
close all;
clc;

n=50;
fp=300;
fst=400;
fs=1000;

window=kaiser(n+1);
b=fir1(n,[fp fst]*2/fs,window);
[h,w]=freqz(b,1,256);

subplot(2,1,1);
plot(w/pi,abs(h));
title('mag response of bpf');
ylabel('gain');
xlabel('normalised freq');

subplot(2,1,2);
plot(w/pi,angle(h));
title('phase response of bpf');
ylabel('angle');
xlabel('normalised freq');
```

RESULT:

Matlab program to find the frequency response of BPF using Kaiser window is written.

Waveforms:


mag response of bpf
gain
normalised freq
phase response of bpf
angle
normalised freq

EXP :1a(4)	FREQUENCY RESPONSE OF BSF USING KAISER WINDOW	

AIM: To write a Matlab program to find the frequency response of BSF using Kaiser window.

EQUIPMENTS:

PC with Matlab Software

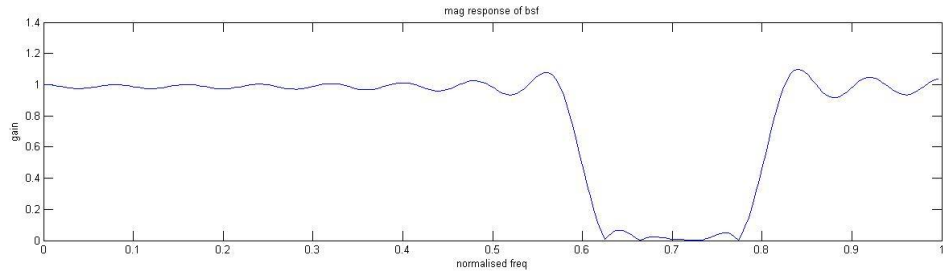
PROGRAM:

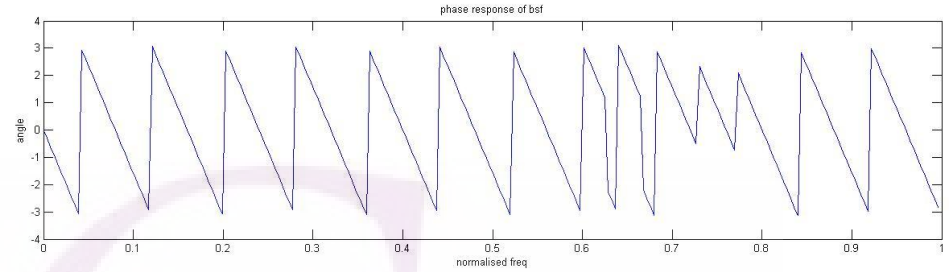
```
clear all;
close all;
clc;

n=50;
fp=300;
fst=400;
fs=1000;

window=kaiser(n+1);
b=fir1(n,[fp fst]*2/fs,'stop',window);
[h,w]=freqz(b,1,256);

subplot(2,1,1);
plot(w/pi,abs(h));
title('mag response of bsf');
ylabel('gain');
xlabel('normalised freq');

subplot(2,1,2);
plot(w/pi,angle(h));
title('phase response of bsf');
ylabel('angle');
xlabel('normalised freq');
```

RESULT:

Matlab program to find the frequency response of BSF using Kaiser window is written.

Waveforms:


mag response of bsf
gain
normalised freq
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0.1
0.3
0.5
0.7
0.9


phase response of bsf
angle
normalised freq
-4
-3
-2
-1
0
1
2
3
4
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

EXP :1b(1)	FREQUENCY RESPONSE OF LPF USING RECTANGULAR WINDOW	

AIM: To write a Matlab program to find the frequency response of LPF using Rectangular window is written.

EQUIPMENTS:

PC with Matlab Software

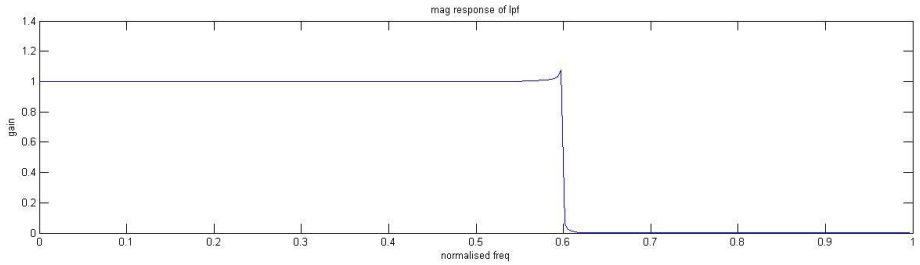
PROGRAM:

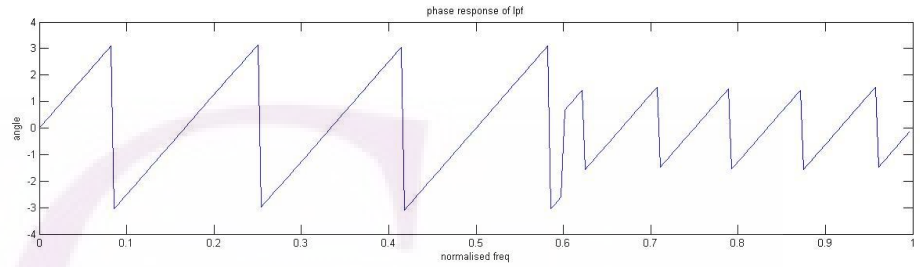
```
clear all;
close all;
clc;

n=1000;
fp=300;
fs=1000;
fn=2*fp/fs;

window=rectwin(n+1);
b=fir1(n,fn,'low',window);
[h,w]=freqz(b,1,256);

subplot(2,1,1);
plot(w/pi,abs(h));
title('mag response of lpf');
ylabel('gain');
xlabel('normalised freq');

subplot(2,1,2);
plot(w/pi,angle(h));
title('phase response of lpf');
ylabel('angle');
xlabel('normalised freq');
```

RESULT:

Matlab program to find the frequency response of LPF using Rectangular window is written.

Waveforms:


mag response of lpf
gain
normalised freq
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0.1
0.3
0.5
0.7
0.9


phase response of lpf
angle
normalised freq
-4
-3
-2
-1
0
1
2
3
4
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

EXP :1b(2)	FREQUENCY RESPONSE OF HPF USING RECTANGULAR WINDOW	

AIM: To write a Matlab program to find the frequency response of HPF using Rectangular window.

EQUIPMENTS:

PC with Matlab Software

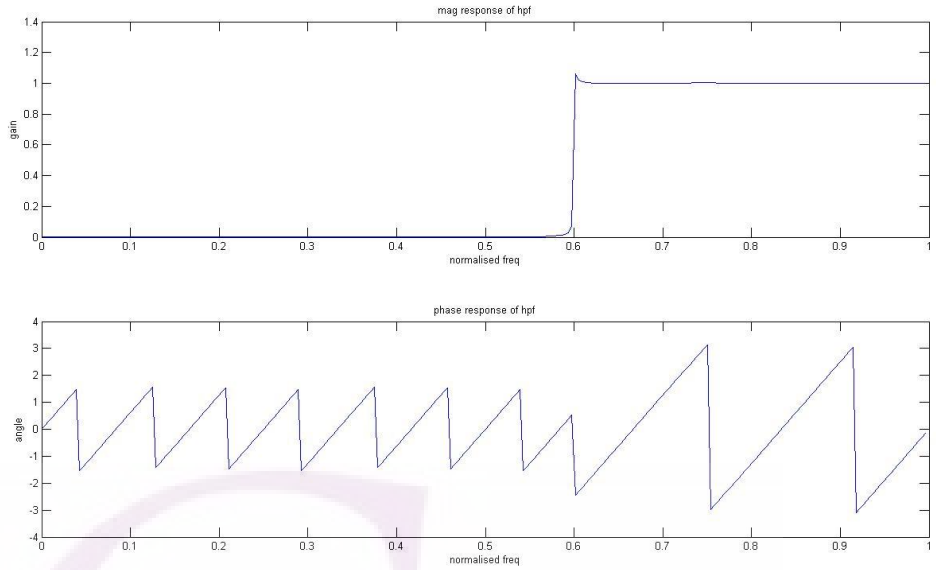
PROGRAM:

```
clear all;
close all;
clc;

n=1000;
fp=300;
fs=1000;
fn=2*fp/fs;

window=rectwin(n+1);
b=fir1(n,fn,'high',window);
[h,w]=freqz(b,1,256);

subplot(2,1,1);
plot(w/pi,abs(h));
title('mag response of hpf');
ylabel('gain');
xlabel('normalised freq');

subplot(2,1,2);
plot(w/pi,angle(h));
title('phase response of hpf');
ylabel('angle');
xlabel('normalised freq');
```

RESULT:

Matlab program to find the frequency response of HPF using Rectangular window is written.

Waveforms:


mag response of hpf
gain
normalised freq
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0.1
0.3
0.5
0.7
0.9

phase response of hpf
angle
normalised freq
-4
-3
-2
-1
0
1
2
3
4
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

EXP :1b(3)	FREQUENCY RESPONSE OF BPF USING RECTANGULAR WINDOW	

AIM: To write a Matlab program to find the frequency response of BPF using Rectangular.

EQUIPMENTS:

PC with Matlab Software

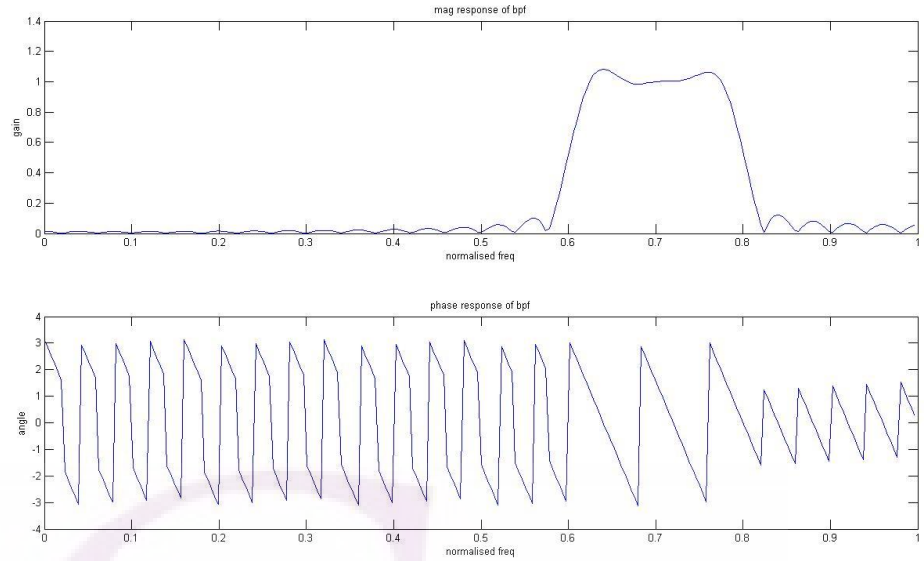
PROGRAM:

```
clear all;
close all;
clc;

n=50;
fp=300;
fst=400;
fs=1000;

window=rectwin(n+1);
b=fir1(n,[fp fst]*2/fs,window);
[h,w]=freqz(b,1,256);

subplot(2,1,1);
plot(w/pi,abs(h));
title('mag response of bpf');
ylabel('gain');
xlabel('normalised freq');

subplot(2,1,2);
plot(w/pi,angle(h));
title('phase response of bpf');
ylabel('angle');
xlabel('normalised freq');
```

RESULT:

Matlab program to find the frequency response of BPF using Rectangular window is written.

Waveforms:


mag response of bpf
gain
normalised freq
phase response of bpf
angle
normalised freq

EXP :1b(4)	FREQUENCY RESPONSE OF BSF USING RECTANGULAR WINDOW	

AIM: To write a Matlab program to find the frequency response of BSF using Rectangular window

EQUIPMENTS:

PC with Matlab Software

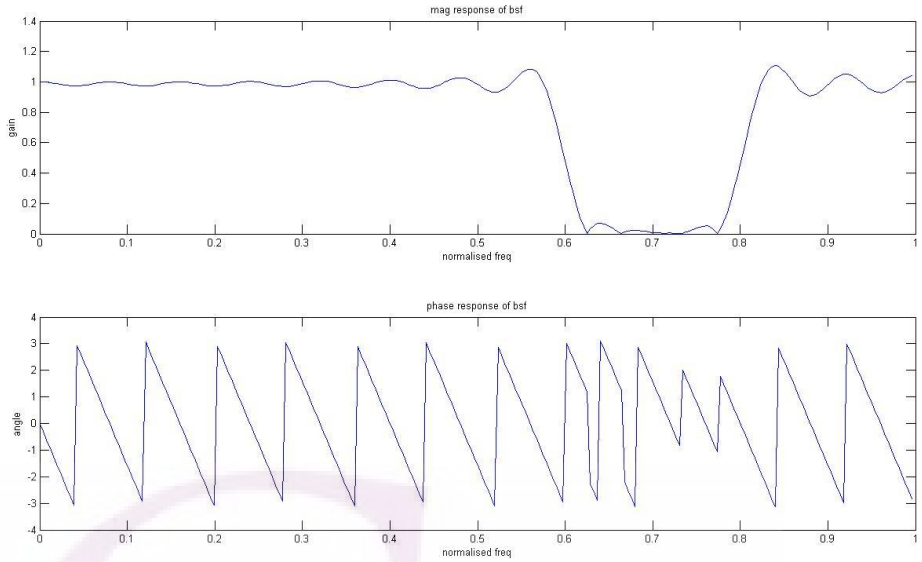
PROGRAM:

```
clear all;
close all;
clc;

n=50;
fp=300;
fst=400;
fs=1000;

window=rectwin(n+1);
b=fir1(n,[fp fst]*2/fs,'stop',window);
[h,w]=freqz(b,1,256);

subplot(2,1,1);
plot(w/pi,abs(h));
title('mag response of bsf');
ylabel('gain');
xlabel('normalised freq');

subplot(2,1,2);
plot(w/pi,angle(h));
title('phase response of bsf');
ylabel('angle');
xlabel('normalised freq');
```

RESULT:

Matlab program to find the frequency response of BSF using Rectangular window is written.

Waveforms:


mag response of bsf
gain
normalised freq
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0.1
0.3
0.5
0.7
0.9

phase response of bsf
angle
normalised freq
-4
-3
-2
-1
0
1
2
3
4
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

EXP :1c(1)	FREQUENCY RESPONSE OF LPF USING TRIANGULAR WINDOW	

AIM: To write a Matlab program to find the frequency response of LPF using traingular window.

EQUIPMENTS:

PC with Matlab Software

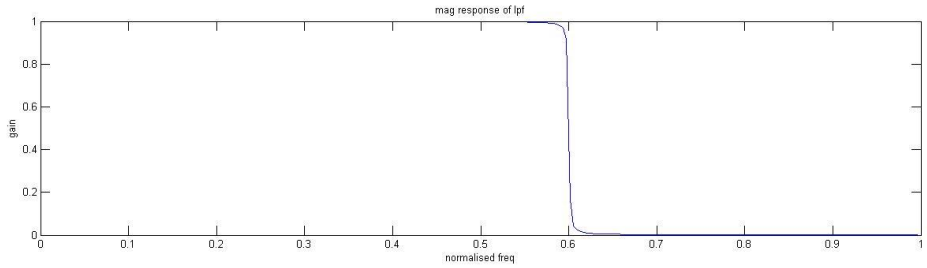
PROGRAM:

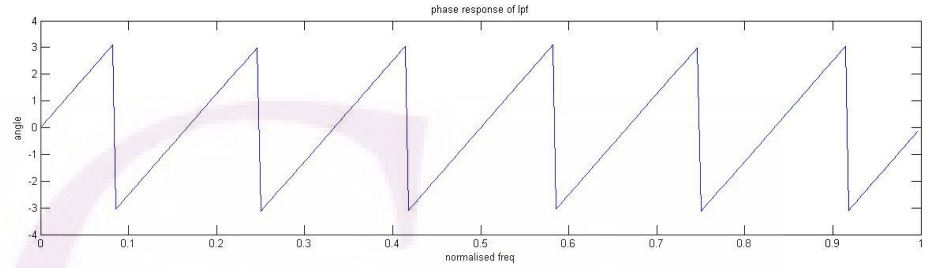
```
clear all;
close all;
clc;

n=1000;
fp=300;
fs=1000;
fn=2*fp/fs;

window=triang(n+1);
b=fir1(n,fn,'low',window);
[h,w]=freqz(b,1,256);

subplot(2,1,1);
plot(w/pi,abs(h));
title('mag response of lpf');
ylabel('gain');
xlabel('normalised freq');

subplot(2,1,2);
plot(w/pi,angle(h));
title('phase response of lpf');
ylabel('angle');
xlabel('normalised freq');
```

RESULT:

a Matlab program to find the frequency response of LPF using traingular window is written.

Waveforms:


mag response of lpf
gain
normalised freq
0
0.2
0.4
0.6
0.8
1
0.1
0.3
0.5
0.7
0.9


phase response of lpf
angle
normalised freq
4
3
2
1
0
-1
-2
-3
-4
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

EXP :1c(2)	FREQUENCY RESPONSE OF HPF USING TRIANGULAR WINDOW	

AIM: To write a Matlab program to find the frequency response of hPF using traingular window.

EQUIPMENTS:

PC with Matlab Software

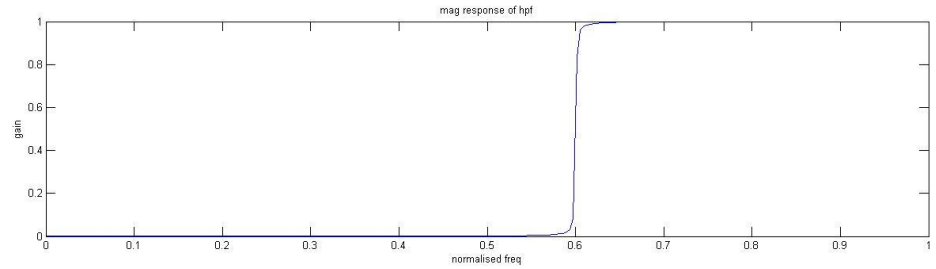
PROGRAM:

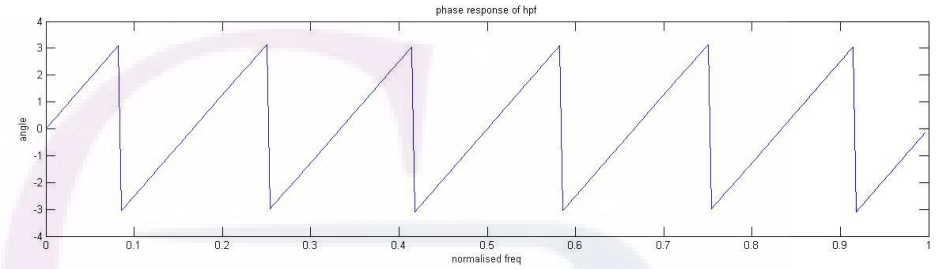
```
clear all;
close all;
clc;

n=1000;
fp=300;
fs=1000;
fn=2*fp/fs;

window=triang(n+1);
b=fir1(n,fn,'high',window);
[h,w]=freqz(b,1,256);

subplot(2,1,1);
plot(w/pi,abs(h));
title('mag response of hpf');
ylabel('gain');
xlabel('normalised freq');

subplot(2,1,2);
plot(w/pi,angle(h));
title('phase response of hpf');
ylabel('angle');
xlabel('normalised freq');
```

RESULT:

a Matlab program to find the frequency response of hPF using traingular window is written.

Waveforms:


mag response of hpf
gain
normalised freq
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0.2
0.4
0.6
0.8


phase response of hpf
angle
normalised freq
4
3
2
1
0
-1
-2
-3
-4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

EXP :1c(3)	FREQUENCY RESPONSE OF BPF USING TRIANGULAR WINDOW	

AIM: To write a Matlab program to find the frequency response of bPF using traingular window is written.

EQUIPMENTS:

PC with Matlab Software

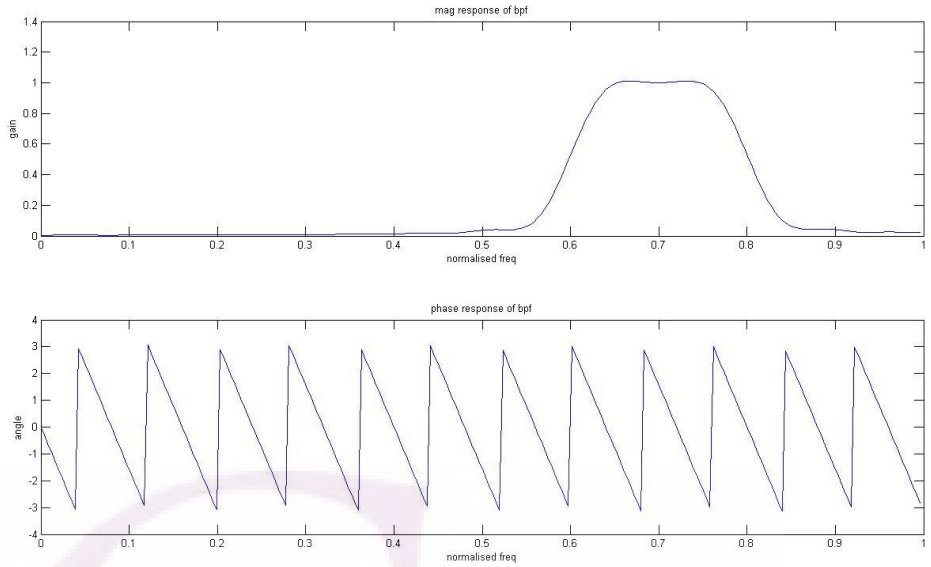
PROGRAM:

```
clear all;
close all;
clc;

n=50;
fp=300;
fst=400;
fs=1000;

window=triang(n+1);
b=fir1(n,[fp fst]*2/fs,window);
[h,w]=freqz(b,1,256);

subplot(2,1,1);
plot(w/pi,abs(h));
title('mag response of bpf');
ylabel('gain');
xlabel('normalised freq');

subplot(2,1,2);
plot(w/pi,angle(h));
title('phase response of bpf');
ylabel('angle');
xlabel('normalised freq');
```

RESULT:

a Matlab program to find the frequency response of LPF using traingular window is written.

Waveforms:


mag response of bpf
gain
normalised freq
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0.1
0.3
0.5
0.7
0.9

phase response of bpf
angle
normalised freq
-4
-3
-2
-1
0
1
2
3
4
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

EXP :1c(4)	FREQUENCY RESPONSE OF BSF USING TRIANGULAR WINDOW	

AIM: To write a Matlab program to find the frequency response of bsF using traingular window.

EQUIPMENTS:

PC with Matlab Software

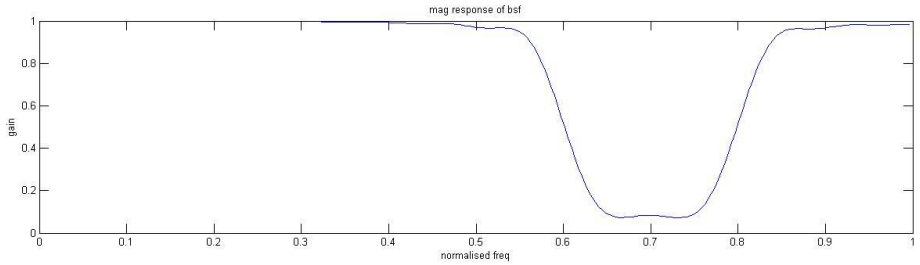
PROGRAM:

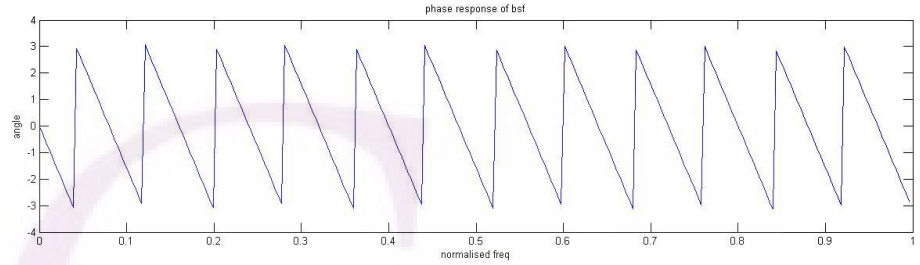
```
clear all;
close all;
clc;

n=50;
fp=300;
fst=400;
fs=1000;

window=triang(n+1);
b=fir1(n,[fp fst]*2/fs,'stop',window);
[h,w]=freqz(b,1,256);

subplot(2,1,1);
plot(w/pi,abs(h));
title('mag response of bsf');
ylabel('gain');
xlabel('normalised freq');

subplot(2,1,2);
plot(w/pi,angle(h));
title('phase response of bsf');
ylabel('angle');
xlabel('normalised freq');
```

RESULT:

a Matlab program to find the frequency response of bsfusing traingular window is written.

Waveforms:


mag response of bsf
gain
normalised freq
0
0.2
0.4
0.6
0.8
1
0.1
0.3
0.5
0.7
0.9


phase response of bsf
angle
normalised freq
4
3
2
1
0
-1
-2
-3
-4
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

EXP :1d(1)	FREQUENCY RESPONSE OF LPF USING HAMMING WINDOW	

AIM: To write a Matlab program to find the frequency response of LPF using hammining window.

EQUIPMENTS:

PC with Matlab Software

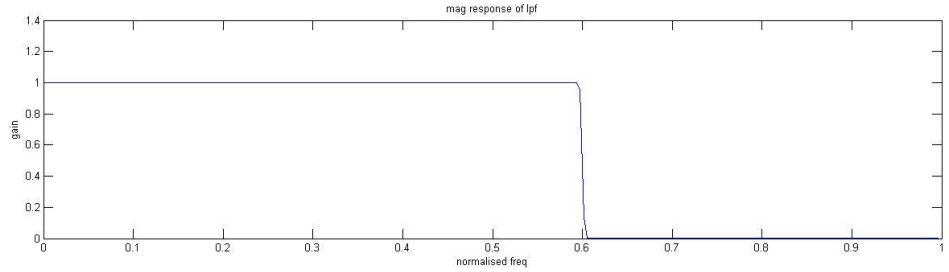
PROGRAM:

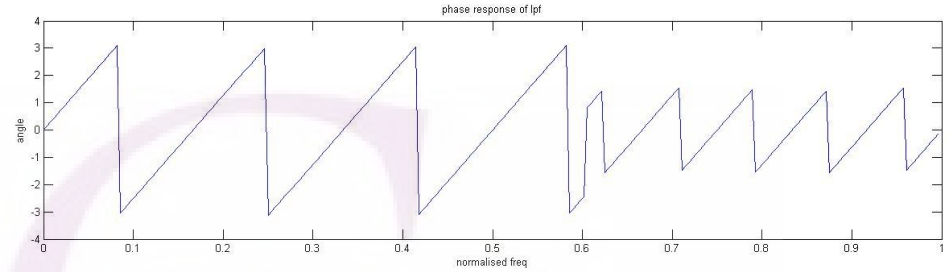
```
clear all;
close all;
clc;

n=1000;
fp=300;
fs=1000;
fn=2*fp/fs;

window=hamming(n+1);
b=fir1(n,fn,'low',window);
[h,w]=freqz(b,1,256);

subplot(2,1,1);
plot(w/pi,abs(h));
title('mag response of lpf');
ylabel('gain');
xlabel('normalised freq');

subplot(2,1,2);
plot(w/pi,angle(h));
title('phase response of lpf');
ylabel('angle');
xlabel('normalised freq');
```

RESULT:

a Matlab program to find the frequency response of LPF using hammining window is written..

Waveforms:


mag response of lpf
gain
normalised freq
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0.1
0.3
0.5
0.7
0.9


phase response of lpf
angle
normalised freq
-4
-3
-2
-1
0
1
2
3
4
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

EXP :1d(2)	FREQUENCY RESPONSE OF HPF USING HAMMING WINDOW	

AIM: To write a Matlab program to find the frequency response of HPF using hammining window.

EQUIPMENTS:

PC with Matlab Software

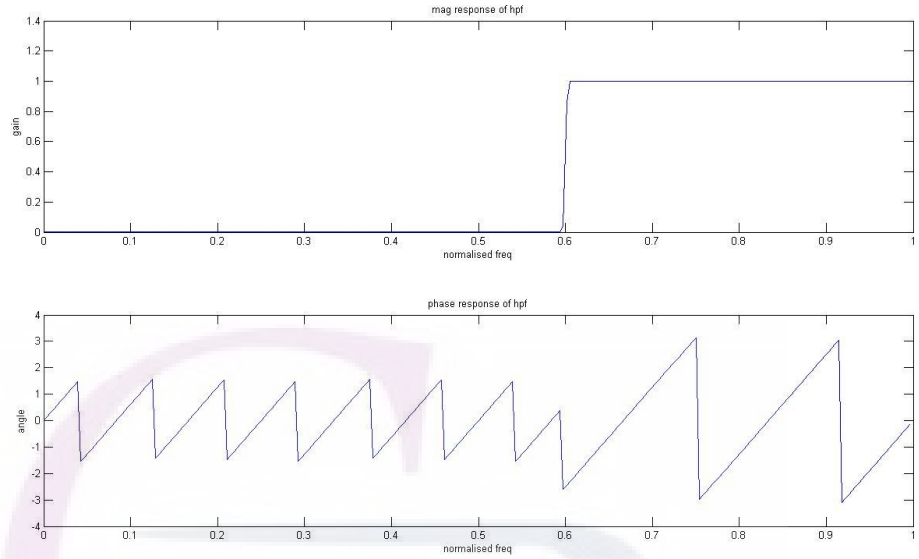
PROGRAM:

```
clear all;
close all;
clc;

n=1000;
fp=300;
fs=1000;
fn=2*fp/fs;

window=hamming(n+1);
b=fir1(n,fn,'high',window);
[h,w]=freqz(b,1,256);

subplot(2,1,1);
plot(w/pi,abs(h));
title('mag response of hpf');
ylabel('gain');
xlabel('normalised freq');

subplot(2,1,2);
plot(w/pi,angle(h));
title('phase response of hpf');
ylabel('angle');
xlabel('normalised freq');
```

RESULT:

a Matlab program to find the frequency response of HPF using hammining window is written.

Waveforms:


mag response of hpf
gain
normalised freq
phase response of hpf
angle
normalised freq

EXP :1d(3)	FREQUENCY RESPONSE OF BPF USING HAMMING WINDOW	

AIM: To write a Matlab program to find the frequency response of BPF using hammining window.

EQUIPMENTS:

PC with Matlab Software

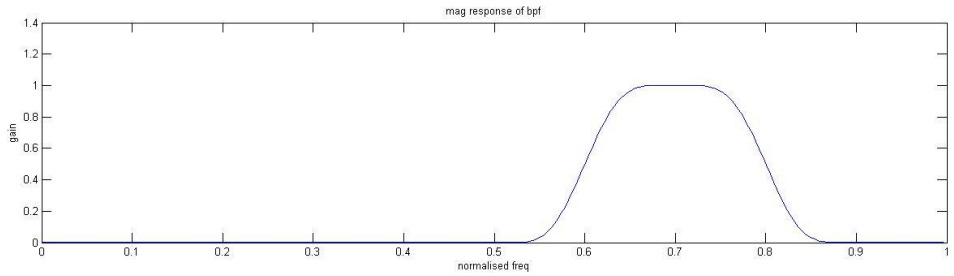
PROGRAM:

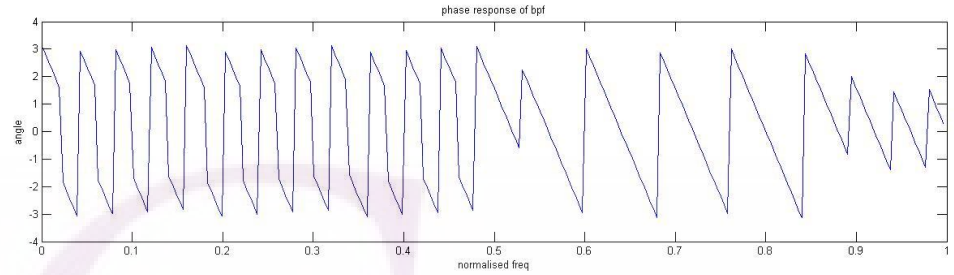
```
clear all;
close all;
clc;

n=50;
fp=300;
fst=400;
fs=1000;

window=hamming (n+1);
b=fir1(n,[fp fst]*2/fs,window);
[h,w]=freqz(b,1,256);

subplot(2,1,1);
plot(w/pi,abs(h));
title('mag response of bpf');
ylabel('gain');
xlabel('normalised freq');

subplot(2,1,2);
plot(w/pi,angle(h));
title('phase response of bpf');
ylabel('angle');
xlabel('normalised freq');
```

RESULT:

To write a Matlab program to find the frequency response of BSP using hammining window is written.

Waveforms:


mag response of bpf
gain
normalised freq
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0.1
0.3
0.5
0.7
0.9


phase response of bpf
angle
normalised freq
-4
-3
-2
-1
0
1
2
3
4
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

EXP :1d(4)	FREQUENCY RESPONSE OF BSF USING HAMMING WINDOW	

AIM: To write a Matlab program to find the frequency response of BSF using hammining window is written.

EQUIPMENTS:

PC with Matlab Software

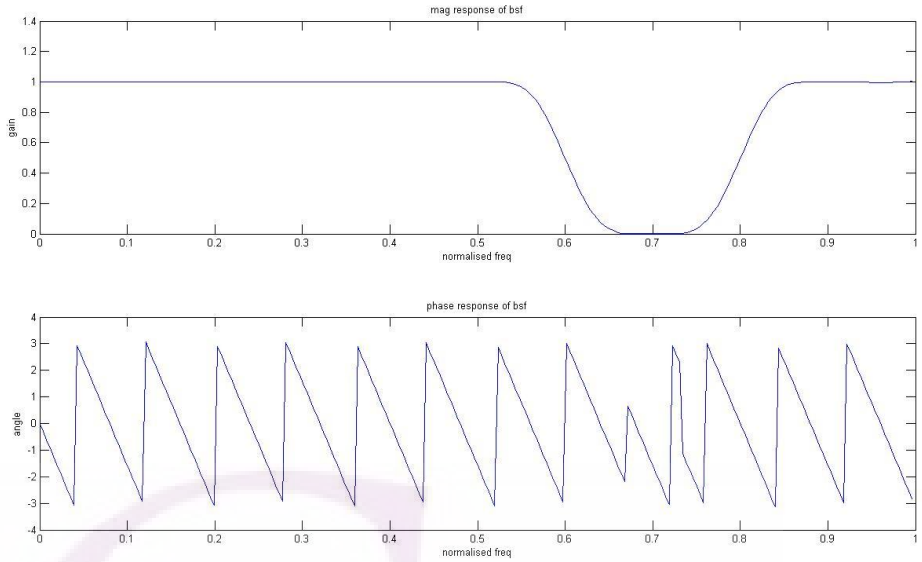
PROGRAM:

```
clear all;
close all;
clc;

n=50;
fp=300;
fst=400;
fs=1000;

window=hamming(n+1);
b=fir1(n,[fp fst]*2/fs,'stop',window);
[h,w]=freqz(b,1,256);

subplot(2,1,1);
plot(w/pi,abs(h));
title('mag response of bsf');
ylabel('gain');
xlabel('normalised freq');

subplot(2,1,2);
plot(w/pi,angle(h));
title('phase response of bsf');
ylabel('angle');
xlabel('normalised freq');
```

RESULT:

a Matlab program to find the frequency response of BSF using hammining window is written.

Waveforms:


mag response of bsf
gain
normalised freq
phase response of bsf
angle
normalised freq