# Experiment No. - 1

# GENERATION OF BASIC SIGNALS USING MATLAB

**(a). Program for the generation of unit step function and delayed step function with a delay of 5.**

| % unit step function and its delay %version.  clear all;  n= -10:20;  u=[zeros(1,10),ones(1,21)];  stem(n,u);  axis([-10 20 0 1.2])  title('Unit step sequence');  xlabel('time index n ---->');  ylabel('Amplitude ----->');  %unit delayed step function  n= -10:20;  m=input('\n Enter the delay=');  ud=[zeros(1,10+m),ones(1,21-m)];  stem(n,ud)  axis([-10 20 0 1.2])  title('Unit delayed step sequence');  xlabel('time index n------>');  ylabel('Amplitude------->');  **Input:** |  |
| --- | --- |
| Enter the delay=5 | |

**(b). Program for the generation of unit sample sequence and delayed unit sample sequence with a delay of 5.**

| % unit sample sequence  clc;  close all;  clc  clear all;  n=-10:20;  u=[zeros(1,10),1,zeros(1,20)];  stem(n,u);  title('Unit sample sequence');  xlabel('time index n');  ylabel('Amplitude');  axis([-10 20 0 1.2])  %unit delayed function  m=input('\n Enter the delay=');  ud=[zeros(1,10+m),1,zeros(1,20-m)];  stem(n,ud);  axis([-20 20 0 2])  title('Unit delayed sample by m');  xlabel('time index n------>');  ylabel('Amplitude------>');  **Input:**  Enter the delay=5 |  |
| --- | --- |

**(c). Program for the generation of unit ramp signal.**

% Program for unit ramp signal.

| clc;  close all;  clear all;  n= 0:10;  u= [(n)<=10];  x=n.\*u;  stem(n,x);  title('Plot of x(n)=n\*u(n)');  axis([0 10.5 0 10.5])  xlabel('time----->');  ylabel('Amplitude---->'); |  |
| --- | --- |

**(d). Program for the generation of exponential signal.**

% Program for the generation of exponential signal.

| clc;  close all;  clear all;  n=100;  h=1/n;  t=0:h:1;  a=exp(-2\*t);  plot(t,a);  title('Graph of y=exp(-2\*t)'); %raising exp(2\*t)  xlabel('x-axis');  ylabel('y-axis'); |  |
| --- | --- |

**e). Plot a graph of y=sin3ᴨx for 0≤ x ≤ 1. Also give the title and label the x & y axes of the**

**graph.**

| %Program to generate sine curve  clc;  close all;  clear all;  n=100;  h=1/n;  x=0:h:1;  y=sin(3\*pi\*x);  plot(x,y);  title('Graph of y= sin(3pix)');  xlabel('x-axis');  ylabel('y-axis'); |  |
| --- | --- |

**(f). Plot a graph of y1=cos3ᴨx. Also give the title and label the x & y axes of the graph.**

| % Program to generate cosine curve  clc;  close all;  clear all;  n=100;  h=1/n;  x=0:h:1;  y1=cos(3\*pi\*x);  plot(x,y1);  title('Graph of y= cos(3pix)');  xlabel('x-axis');  ylabel('y-axis'); |  |
| --- | --- |

**(g). Plot a graph of sin(nᴨx) on the interval -1≤x ≤1 for n=1,2,……8 using for loop.[Note use subplot command to plot the 8 graphs in one window].**

| % Program to generate sin(n𝝿x) curve  clc;  close all;  clear all;  m=100;  h=1/m;  x=-1:h:1;  for n=1:8  subplot(4,2,n)  y=sin(n\*pi\*x);  plot(x,y);  title('Graph of sin(n𝝿x)');  xlabel('x-axis');  ylabel('y-axis');  end |  |
| --- | --- |

**h). Plot an exponentially decaying sine wave A= e-2tsin(2𝝿5t), -1≤t ≤1. Also demonstrate the use of ‘subplot’ and ‘figure’ commands.**

| % Program to generate exponentially %decaying sine wave  clc;  clear all;  close all;  n= 100;  h= 1/n;  t=0:h:1;  subplot(3,1,1)  a=exp(-2\*t);  plot(t,a);  title('Graph of y= exp(-2\*t)');  % raising exp(2\*t)  xlabel('x-axis');  ylabel('y-axis');  subplot(3,1,2);  b=(sin(2\*pi\*5\*t));  plot(t,b);  title('Graph of y= sin(2\*pi\*5\*t)');  xlabel('x-axis');  ylabel('y-axis');  subplot(3,1,3);  A= a.\*b;  plot(t,A);  title('Graph of y= exp(-2\*t)sin(2\*pi\*5\*t)');  xlabel('x-axis');  ylabel('y-axis'); |  |
| --- | --- |

**i). Generate and plot a triangular wave.**

| % Generation of triangular wave  clc;  close all;  clear all;  t= 0:0.01:6;  x1=sawtooth(2\*pi\*t,0.5);  plot(t,x1);  title('triangular wave');  xlabel('Time');  ylabel('Amplitude'); |  |
| --- | --- |

**j). Generation and plot square wave with duty cycle 25,50 and 75%.**

%Program to generate square with various duty cycle

| clc;  close all;  clear all;  t=0:0.01:6;  x1= square(2\*pi\*t,25);  subplot(3,1,1)  plot(t,x1);  axis([0 6 -1.4 1.4]);  legend('D=25%');  xlabel('time');  ylabel('Amplitude');  x2=square(2\*pi\*t,50);  subplot(3,1,2);  plot(t,x2);  axis([0 6 -1.4 1.4]);  legend('D=50%');  xlabel('time');  ylabel('Amplitude');  x3=square(2\*pi\*t,75);  subplot(3,1,3);  plot(t,x3);  axis([0 6 -1.4 1.4]);  legend('D=75%');  xlabel('time');  ylabel('Amplitude'); |  |
| --- | --- |

**Conclusion:**

**Viva Questions:**

1. Define sinusoidal signal

2. Define C.T.S

3. Define D.T.S.

4. Compare C.T.S & D.T.S

5. Define Stem, Plot, Plot3,fplot, ezplot, linspace, flyplr, grid,mesh and legend

6. Draw the C.T.S & D.T.S diagrams

# Experiment No. 2

# EVALUATION OF IMPULSE RESPONSE OF A SYSTEM

## **PROGRAM: IMPLULSE RESPONSE USING DECONV FUNCTION**

clc; % clear screen

close all % close all figure windows

clear all; % clear work space

y = input('Output sequence y(n) of the system = '); % enter the output sequence

x = input('Input sequence x(n) of the system = '); % enter the input sequence

[h,r] = deconv(y,x); % deconvolute output and input to get the impulse response disp('Impulse response of the system is = ');

disp(h); % display result

N= length(h); % find the length of h

n = 0:1:N-1; % define time axis

stem(n,h); % plot the impulse response

xlabel('n'); % label x axis

ylabel('h(n)'); % label y axis

title('Impulse Response of the system'); % graph title

## **OUTPUT:**

Output sequence y(n) of the system = [1 -3/4]

Input sequence x (n) of the system = [1 -3/4 1/8]

Impulse response of the system = [1 0 –0.125 -0.0937 -0.0546]

## **PROGRAM: IMPLULSE RESPONSE USING IMPZ FUNCTION**

%Program for impulse response using IMPZ function

clc; % clear screen

close all; % close all figure windows

clear all; % clear workspace

y=input('Output sequence y(n) of the system = ');% enter the coefficients of y terms

x=input('Input sequence x(n) of the system = '); % enter the coefficient of x terms

N=input('Enter the length of impulse response =');% define the length of the response

h=impz(y,x,N); % calculate the impulse response

disp('Impulse Response of system h(n)'); % display the values of impulse response

disp(h); % graphical plot

n=0:1:N-1; % define x axis

stem(n,h); % plot the impulse response

xlabel('n'); % label x axis

ylabel('h(n)'); % label y axis

title('Impulse Response'); % graph title

## **OUTPUT:**

Output sequence y(n) of the system = [1 -3/4]

Input sequence x(n) of the system = [1 -3/4 1/8]

Enter the length of impulse response =5

Impulse Response of system h(n)= 1.0000 0 -0.1250 -0.0938 -0.0547

Graphical user interface

Description automatically generated

## **PROGRAM: IMPLULSE RESPONSE USING FILTER FUNCTION**

%Program for impulse response using filter function

clc; % clear screen

close all; % close all the figure windows

clear all; % clear the work space

y=input('Enter the co-efficient of y = '); % define the coefficient of y terms

x=input('Enter the co-efficient of x = '); % define the coefficient of x terms

N=input('Enter the length of impulse sequence =');% define the length of the response

xi=[1,zeros(1,N-1)]; % define the input signal

h=filter(x,y,xi); % calculate the impulse response

disp( 'Impulse Response of the system is = ');% display the response

disp(h); % graphical display

n=0:1:N-1; % define the time axis

stem(n,h); % plot the impulse response

xlabel('n'); % label x axis

ylabel('h(n)'); % label y axis

title('Impulse Response of the system'); % graph title

## **OUTPUT:**

Enter the co-efficient of y = [1 -3/4 1/8]

Enter the co-efficient of x = [1 -3/4]

Enter the length of impulse sequence = 5

Impulse Response of the system is =

1.0000 0 -0.1250 -0.0938 -0.0547

**VIVA QUESTIONS WITH ANSWERS**

1. What are the various methods available to determine the response of LTI systems?

2. What is impulse response and what is its significance?

3. Define the transfer function of an LTI system

4. What is BIBO stability? What is the condition to be satisfied foe stability?

5. What do you mean by real time signal? Give example.

# Experiment No. 3

# COMPUTATION OF N – POINT DFT AND TO PLOT THE MAGNITUDE AND PHASE SPECTRUM

**Program:**

%without fft for dft %

clc;

clear all;

close all;

x=input('enter the 1st value');

N=length(x);

for k=0:N-1

y(k+1)=0;

for n=0:N-1

y(k+1)=y(k+1)+x(n+1)\*exp(-i\*2\*pi\*n\*k/N);

end

end

y

mag=abs(y)

phase=angle(y)

subplot(2,2,1)

n=0:1:length(x)-1;

stem(n,x)

title('the values of x');

subplot(2,2,2)

n1=0:1:length(y)-1;

stem(n1,y)

title('output of y');

subplot(2,2,3)

n2=0:1:length(y)-1;

stem(n2,mag)

title('magnitude plot');

subplot(2,2,4)

n3=0:1:length(y)-1

stem(n3,phase)

title('phase plot');

**Output:**

enter the 1st value[1 2 3 4]

N = 4

y = 10.0000 -2.0000 + 2.0000i -2.0000 - 0.0000i -2.0000 - 2.0000i

mag = 10.0000 2.8284 2.0000 2.8284

phase = 0 2.3562 -3.1416 -2.3562

n3 = 0 1 2 3

# Experiment No. 4

# TO PERFORM LINEAR CONVOLUTION OF GIVEN SEQUENCES

**Program:**

%linear convolution %

clc;

clear all;

close all;

x=input('enter the 1st value');

h=input('enter the 2nd value');

y=conv(x,h);

disp('the o/p of linear convolution is');

disp(y)

subplot(2,2,1);

n=0:1:length(x)-1;

stem(n,x)

title('the 1st o/p value');

xlabel('time index');

ylabel('ampitude');

subplot(2,2,2);

n2=0:1:length(h)-1;

stem(n2,h)

title('the 2nd o/p value');

xlabel('time index');

ylabel('ampitude');

subplot(2,2,[3 4]);

n3=0:1:length(y)-1;

stem(n3,y)

title('the linear convolution o/p values ');

xlabel('time index');

ylabel('ampitude');

**Output:**

enter the 1st value[1 2 3]

enter the 2nd value[1 2 3]

the o/p of linear convolution is

1 4 10 12 9

# Experiment No. 5

# LINEAR AND CIRCULAR CONVOLUTION BY DFT AND IDFT METHOD.

**Program:**

**(i) Linear convolution using DFT and IDFT:**

clc;

clear all;

close all;

x=input('enter the 1st value');

h=input('enter the 2nd value');

N=length(x)+length(h)-1

Xk=fft(x,N)

Hk=fft(h,N)

Yk=Xk.\*Hk

yn=ifft(Yk,N)

yr=real(yn);

subplot(2,2,1)

n=0:1:length(x)-1;

stem(n,x)

title('the values of x');

subplot(2,2,2)

n1=0:1:length(h)-1;

stem(n1,h)

title(' values of h');

subplot(2,2,[3 4])

n2=0:1:N-1;

stem(n2,yr)

title('real of ifft');

**Output:**

enter the 1st value[1 2 3]

enter the 2nd value[1 1]

N =4

Xk =6.0000 -2.0000 - 2.0000i 2.0000 -2.0000 + 2.0000i

Hk =2.0000 1.0000 - 1.0000i 0 1.0000 + 1.0000i

Yk =12 -4 0 -4

yn = 1 3 5 3

**(ii)Circular convolution using DFT and IDFT:**

clc;

clear all;

close all;

x=input('enter the 1st value');

h=input('enter the 2nd value');

N1=length(x);

N2=length(h);

N=max(N1,N2);

Xk=fft(x,N)

Hk=fft(h,N)

Yk=Xk.\*Hk

yn=ifft(Yk,N)

yr=real(yn)

subplot(2,2,1)

n=0:1:length(x)-1;

stem(n,x)

title('the values of x');

subplot(2,2,2)

stem(n,h)

title(' values of h');

subplot(2,2,[3 4])

stem(n,yr)

title('real of ifft');

disp(yr);

**Output:**

enter the 1st value[1 2 3 4]

enter the 2nd value[1 1 1 1]

Xk =10.0000 -2.0000 + 2.0000i -2.0000 -2.0000 - 2.0000i

Hk =4 0 0 0

Yk =40 0 0 0

yn =10 10 10 10

yr =10 10 10 10

# Experiment No. 5

# TO PERFORM CIRCULAR CONVOLUTION OF GIVEN SEQUENCES USING (A) THE CONVOLUTION SUMMATION FORMULA (B) THE MATRIX METHOD AND (C) LINEAR CONVOLUTION FROM CIRCULAR CONVOLUTION WITH ZERO PADDING

1. **Circular convolution using convolution summation formula:**

**Program:**

%circular convolution%

clc;

close all

clear all

x=input('enter the 1st value');

h=input('enter the 2nd value');

N1=length(x);

N2=length(h);

N=max(N1,N2);

for n=0:1:N-1

y(n+1)=0;

for M=0:1:N-1

i=n-M;

if(i<0)

i=i+N;

end

y(n+1)=y(n+1)+x(M+1)\*h(i+1);

end

end

disp(y)

n=0:1:N-1;

stem(n,y)

title('circular convolution plot');

xlabel('time');

ylabel('amplitude');

**Output:**

enter the 1st value[1 2 2 1]

enter the 2nd value[4 3 2 1]

13 15 17 15

1. **Circular convolution using matrix method:**

**1st Method**

%matrix convolution method%

clc;

clear all;

close all;

x=input('enter the 1st sequence');

h=input('enter the 2nd sequence');

y=h\*x;

disp('the circular conv o/p is');

disp(y);

subplot(3,1,1)

n=0:1:length(x)-1;

stem(n,x);

title('input of x');

xlabel('time');

ylabel('amplitude');

subplot(3,1,2)

n1=0:1:length(h)-1;

stem(n1,h);

title('input of h');

xlabel('time');

ylabel('amplitude');

subplot(3,1,3)

n2=0:1:length(y)-1;

stem(n2,y);

title('output of y');

xlabel('time');

ylabel('amplitude');

**Output:**

enter the 1st sequence[1;2;3]

enter the 2nd sequence[1 3 2;2 1 3;3 2 1]

the circular conv o/p is

13

13

10

**2nd Method**

%circular matrix mutlipication%

clc;

clear all;

close all;

x=input('enter the 1st sequence');

h=input('enter the 2nd sequence');

a=h'

A=circshift(a,1);

B=circshift(a,2);

C=[a A B]

y=C\*x';

disp('the circular conv o/p is');

disp(y);

subplot(3,1,1)

n=0:1:length(x)-1;

stem(n,x);

title('input of x');

xlabel('time');

ylabel('amplitude');

subplot(3,1,2)

n1=0:1:length(h)-1;

stem(n1,h);

title('input of h');

xlabel('time');

ylabel('amplitude');

subplot(3,1,3)

n2=0:1:length(y)-1;

stem(n2,y);

title('input of y');

xlabel('time');

ylabel('amplitude');

**Output:**

**Plot:**

enter the 1st sequence[1 2 3]

enter the 2nd sequence[1 2 3]

a =

1

2

3

C =

1 3 2

2 1 3

3 2 1

the circular conv o/p is

13

13

10

1. **Linear convolution from circular convolution with zero padding**

**Program:**

clc;

clear all;

close all;

x=input('enter the 1st value');

h=input('enter the 2nd value');

N=length(x)+length(h)-1;

X=[x,zeros(1,N-length(x))];

H=[h,zeros(1,N-length(h))];

for n=0:N-1

Y(n+1)=0;

for M=0:N-1

i=n-M;

if(i<0)

i=i+N;

end

Y(n+1)=Y(n+1)+X(M+1)\*H(i+1);

end

end

disp(Y)

stem(Y)

title('the output values');

xlabel('time');

ylabel('amplitude');

**Output:**

enter the 1st value[1 2 3 1]

enter the 2nd value[1 1 1]

1 3 6 6 4 1

# Experiment No. 7

# VERIFICATION OF SAMPLING THEOREM BOTH IN TIME AND FREQUENCY DOMAINS

**Program:**

clc; % clears the command window

clear all; % clears the variables declared

t=0:0.001:0.1;

f1=input ('Enter the input frequency1 = ');

f2=input ('Enter the input frequency2 = ');

y=cos(2\*pi\*f1\*t)+cos(2\*pi\*f2\*t);

f3=max(f1,f2);

% under sampling

fs=f3; %fs = sampling frequency

ts=1/fs;

tx=0:ts:0.1;

ys=cos(2\*f1\*pi\*tx)+cos(2\*pi\*f2\*tx);

figure(1);

subplot(3,1,1);

plot(t,y);

title('The cosine signal cos(2\*pi\*f1\*t)+cos(2\*pi\*f2\*t)');

xlabel('Time in seconds');

ylabel('Amplitude in volts');

subplot(3,1,2);

stem(tx,ys);

title('The cosine signal sampled at fs Hz');

xlabel('Time in seconds');

ylabel('Amplitude in volts');

subplot(3,1,3);

plot(tx,ys);

title('The recovered cosine sampled at fs Hz');

xlabel('Time in seconds');

ylabel('Amplitude in volts');

% Right sampling

fs=2\*f3; %fs = sampling frequency

ts=1/fs;

tx=0:ts:0.1;

ys=cos(2\*pi\*f1\*tx)+cos(2\*pi\*f2\*tx);

figure(2);

subplot(3,1,1);

plot(t,y);

title('The cosine signal cos(2\*pi\*f1\*t)+cos(2\*pi\*f2\*t)');

xlabel('Time in seconds');

ylabel('Amplitude in volts');

subplot(3,1,2);

stem(tx,ys);

title('The cosine signal sampled at fs Hz');

xlabel('Time in seconds');

ylabel('Amplitude in volts');

subplot(3,1,3);

plot(tx,ys);

title('The recovered cosine sampled at fs Hz');

xlabel('Time in seconds');

ylabel('Amplitude in volts');

% over sampling

fs=3\*f3; %fs = sampling frequency

ts=1/fs;

tx=0:ts:0.1;

ys=cos(2\*pi\*f1\*tx)+cos(2\*pi\*f2\*tx);

figure(3);

subplot(3,1,1);

plot(t,y);

title('The cosine signal cos(2\*pi\*f1\*t)+cos(2\*pi\*f2\*t)');

xlabel('Time in seconds');

ylabel('Amplitude in volts');

subplot(3,1,2);

stem(tx,ys);

title('The cosine signal sampled at fs Hz');

xlabel('Time in seconds');

ylabel('Amplitude in volts');

subplot(3,1,3);

plot(tx,ys);

title('The recovered cosine sampled at fs Hz');

xlabel('Time in seconds');

ylabel('Amplitude in volts');

**Output:**

Enter the input frequency1 = 100

Enter the input frequency2 = 200

# Experiment No. 8

# SOLUTION OF A GIVEN DIFFERENCE EQUATION

**Given difference equation is y(n)+0.5y(n-1)=2x(n).**

1. **Step input**

clc;

clear all;

close all;

a=input('enter the 1st values');

b=input('enter the 2nd values');

N=input('enter the length of the impulse response requirement');

x=[ones(1,N)];

y=filter(b,a,x);

y

subplot(2,1,1)

n=0:1:length(x)-1;

stem(n,x)

title('the impulse input');

xlabel('time index');

ylabel('Ampiltude');

subplot(2,1,2)

n1=0:1:length(y)-1;

stem(n1,y)

title('output values');

xlabel('time index');

ylabel('Ampiltude');

**Output**:

enter the 1st values[1 0.5]

enter the 2nd values[2]

enter the length of the impulse response

requirement4

y = 2.0000 1.0000 1.5000 1.2500

1. **Impulse input**

clc;

clear all;

close all;

a=input('enter the 1st values');

b=input('enter the 2nd values');

N=input('enter the length of the impulse response requirement');

x=[1,zeros(1,N-1)];

y=filter(b,a,x);

y

subplot(2,1,1)

n=0:1:length(x)-1;

stem(n,x)

title('the impulse input');

xlabel('time index');

ylabel('Ampiltude');

subplot(2,1,2)

n1=0:1:length(y)-1;

stem(n1,y)

title('output values');

xlabel('time index');

ylabel('Ampiltude');

**Output:**

enter the 1st values[1 0.5]

enter the 2nd values[2]

enter the length of the impulse response

requirement4

y = 2.0000 -1.0000 0.5000 -0.2500

1. **Arbitrary input**

clc;

clear all;

close all;

a=input('enter the 1st values');

b=input('enter the 2nd values');

N=input('enter the length of the impulse response requirement');

n=0:1:N-1;

x=2.^n

y=filter(b,a,x);

y

subplot(2,1,1)

n1=0:1:length(x)-1;

stem(n1,x)

title('the impulse input');

xlabel('time index');

ylabel('Ampiltude');

subplot(2,1,2)

n2=0:1:length(y)-1;

stem(n2,y)

title('output values');

xlabel('time index');

ylabel('Ampiltude');

**Output:**

enter the 1st values[1 0.5]

enter the 2nd values[2]

enter the length of the impulse response

requirement4

x =1 2 4 8

y = 2.0000 3.0000 6.5000 12.7500

1. **Steady state input**

clc;

clear all;

close all;

a=input('enter the 1st values');

b=input('enter the 2nd values');

N=input('enter the length of the impulse response requirement');

n=0:1:N-1;

x=cos(0.5\*pi\*n)

y=filter(b,a,x);

y

subplot(2,1,1)

n1=0:1:length(x)-1;

stem(n1,x)

title('the impulse input');

xlabel('time index');

ylabel('Ampiltude');

subplot(2,1,2)

n2=0:1:length(y)-1;

stem(n2,y)

title('output values');

xlabel('time index');

ylabel('Ampiltude');

**Output:**

enter the 1st values[1 0.5]

enter the 2nd values[2]

enter the length of the impulse response

requirement4

x =1.0000 0.0000 -1.0000 -0.0000

y =2.0000 -1.0000 -1.5000 0.7500

# Experiment No. 9

# CALCULATION OF DFT AND IDFT BY FFT

**Program:**

clc;

clear all;

close all;

x=input('Enter the sequence')

N=length(x);

y=fft(x,N)

mag=abs(y)

phase=angle(y)

phase1=phase\*(180/pi);

%verification%

yi=ifft(y)

yd=real(yi)

subplot(2,2,1)

n=0:1:length(x)-1;

stem(n,x)

title('the values ox x');

subplot(2,2,2)

stem(n,mag)

title('magnitude values of y');

subplot(2,2,3)

stem(n,phase1)

title('phase values of y');

subplot(2,2,4)

stem(n,yd)

title('real of ifft')

**Output:**

Enter the sequence[1 0 -1 0]

x =1 0 -1 0

y =0 2 0 2

mag =0 2 0 2

phase =0 0 0 0

yi =1 0 -1 0

yd =1 0 -1 0

# Experiment No. 10

# DESIGN AND IMPLEMENTATION OF IIR FILTERS TO MEET GIVEN SPECIFICATION (LOW PASS, HIGH PASS, BAND

# PASS AND BAND REJECT FILTERS)

**Aim:** To design and implement Butterworth and Chebyshev-1 IIR filter for given specifications.

1. **Design of Low pass Butterworth filter using Bilinear Transformation**

The equation to pre warp the frequencies





The equation to find the order of the filter

Text

Description automatically generated

The equation to find the cut off frequency of the filter





To convert to digital domain

**ALGORITHM:**

1. Get the order of the filter

2. Find the filter coefficients

3. Plot the magnitude response

**PROGRAM:**

clc;

close all;

clear all;

ap=input('enter the pass band attenuation');

as=input('enter the stop band attenuation');

fp=input('enter the pass band frequency');

fs=input('enter the stop band frequency');

sf=input('sampling frequency');

wp=2\*pi\*fp/sf

ws=2\*pi\*fs/sf

wp1=2\*tan(wp/2)

wp2=2\*tan(ws/2)

[n,fc]=buttord(wp1,wp2,ap,as,'s')

[b,a]=butter(n,1,'s')

[b1,a1]=lp2lp(b,a,fc)

fs=1;

[num,den]=bilinear(b1,a1,fs)

freqz(num,den);

**Output: Plot:**

enter the pass band attenuation3.01Chart, scatter chart

Description automatically generated

enter the stop band attenuation15

enter the pass band frequency500

enter the stop band frequency750

sampling frequency2000

wp =1.5708

ws =2.3562

wp1 =2.0000

wp2 =4.8284

n =2

fc =2.0526

b =0 0 1

a =1.0000 1.4142 1.0000

b1 =4.2130

a1 = 1.0000 2.9027 4.2130

num =0.3005 0.6011 0.3005

den =1.0000 0.0304 0.1717

1. **Design of High pass Butterworth filter using Bilinear Transformation**

The equation to pre warp the frequencies





The equation to find the order of the filter

Text

Description automatically generated

The equation to find the cut off frequency of the filter



Lowpass to highpass transformation is given by  

To convert to digital domain

**Program:**

clc;

close all;

clear all;

ap=input('enter the pass band attenuation');

as=input('enter the stop band attenuation');

fp=input('enter the pass band frequency');

fs=input('enter the stop band frequency');

sf=input('sampling frequency');

wp=2\*pi\*fp/sf

ws=2\*pi\*fs/sf

wp1=2\*tan(wp/2)

wp2=2\*tan(ws/2)

[n,fc]=buttord(wp1,wp2,ap,as,'s')

[b,a]=butter(n,1,'s')

[b1,a1]=lp2hp(b,a,fc)

fs=1;

[num,den]=bilinear(b1,a1,fs)

freqz(num,den);

**Output: Plot** Chart

Description automatically generated

enter the pass band attenuation3.01

enter the stop band attenuation15

enter the pass band frequency500

enter the stop band frequency750

sampling frequency2000

wp =1.5708

ws =2.3562

wp1 =2.0000

wp2 =4.8284

n =2

fc =2.0526

b =0 0 1

a =1.0000 1.4142 1.0000

b1 =1.0000 -0.0000 -0.0000

a1 =1.0000 2.9027 4.2130

num =0.2853 -0.5707 0.2853

den = 1.0000 0.0304 0.1717

1. **Design of Low pass Chebyshev-I filter using Bilinear Transformation**

The equation to pre warp the frequencies



Equations used to design filter are





To find the order of the filter

A picture containing chart

Description automatically generated

To find the poles of the equation







To convert to digital domain

**Program:**

clc;

close all;

clear all;

ap=input('enter the pass band attenuation');

as=input('enter the stop band attenuation');

fp=input('enter the pass band frequency');

fs=input('enter the stop band frequency');

sf=input('sampling frequency');

wp=2\*pi\*fp/sf

ws=2\*pi\*fs/sf

wp1=2\*tan(wp/2)

wp2=2\*tan(ws/2)

[n,fc]=cheb1ord(wp1,wp2,ap,as,'s')

[b,a]=cheby1(n,ap,1,'s')

[b1,a1]=lp2lp(b,a,fc)

fs=1;

[num,den]=bilinear(b1,a1,fs)

freqz(num,den);

**Output:**

enter the pass band attenuation2

enter the stop band attenuation20

enter the pass band frequency100

enter the stop band frequency500

sampling frequency4000

wp =0.1571

ws =0.7854

wp1 =0.1574

wp2 = 0.8284

n =2

fc =0.1574

b = 0 0 0.6538

a =1.0000 0.8038 0.8231

b1 =0.0162

a1 = 1.0000 0.1265 0.0204

num =0.0038 0.0076 0.0038

den =1.0000 -1.8625 0.8816

**Plot:**

Chart

Description automatically generated

1. **Design of High pass Chebyshev-I filter using Bilinear Transformation**

**Program:**

clc;

close all;

clear all;

ap=input('enter the pass band attenuation');

as=input('enter the stop band attenuation');

fp=input('enter the pass band frequency');

fs=input('enter the stop band frequency');

sf=input('sampling frequency');

wp=2\*pi\*fp/sf

ws=2\*pi\*fs/sf

wp1=2\*tan(wp/2)

wp2=2\*tan(ws/2)

[n,fc]=cheb1ord(wp1,wp2,ap,as,'s')

[b,a]=cheby1(n,ap,1,'s')

[b1,a1]=lp2hp(b,a,fc)

fs=1;

[num,den]=bilinear(b1,a1,fs)

freqz(num,den);

**Output: Plot:**

enter the pass band attenuation2Chart

Description automatically generated

enter the stop band attenuation20

enter the pass band frequency100

enter the stop band frequency500

sampling frequency4000

wp =0.1571

ws =0.7854

wp1 = 0.1574

wp2 =0.8284

n =2

fc =0.1574

b =0 0 0.6538

a =1.0000 0.8038 0.8231

b1 =0.7943 -0.0000 0.0000

a1 =1.0000 0.1537 0.0301

num =0.7325 -1.4650 0.7325

den =1.0000 -1.8305 0.8582

**VIVA QUESTIONS:**

1) What are IIR filters? What does "IIR" mean?

2) Why is the impulse response "infinite"?

3) What is the alternative to IIR filters?

4) What are the advantages of IIR filters (compared to FIR filters)?

5) What are the disadvantages of IIR filters (compared to FIR filters)?

6) What is prewrapping?

7) Give the frequency transformation techniques?

8) Give the disadvantage of impulse invariance method?

9) What is an elliptic filter?

10) Differentiate the three prototype filters

a) Based on frequency characteristics

b) Based on poles and zeros

11) What is the condition for stability of an IIR system?

# Experiment No. 11

# DESIGN AND IMPLEMENTATION OF FIR FILTERS TO MEET GIVEN SPECIFICATION (LOW PASS, HIGH PASS, BAND PASS AND BAND REJECT FILTERS) USING DIFFERENT WINDOW FUNCTIONS

**Aim:** To design and implement a FIR filter for given specifications.

**Theory:**

FIR filters are digital filters with finite impulse response. They are also known as non-recursive digital filters as they do not have the feedback.

An FIR filter has two important advantages over an IIR design:

* Firstly, there is no feedback loop in the structure of an FIR filter. Due to not having a feedback loop, an FIR filter is inherently stable. Meanwhile, for an IIR filter, we need to check the stability.
* Secondly, an FIR filter can provide a linear-phase response. As a matter of fact, a linear-phase response is the main advantage of an FIR filter over an IIR design otherwise, for the same filtering specifications; an IIR filter will lead to a lower order.

**Fir filter design**

An FIR filter is designed by finding the coefficients and filter order that meet certain specifications, which can be in the time-domain (e.g. a matched filter) and/or the frequency domain (most common). Matched filters perform a cross-correlation between the input signal and a known pulse-shape. The FIR convolution is a cross-correlation between the input signal and a time-reversed copy of the impulse-response. Therefore, the matched-filter's impulse response is "designed" by sampling the known pulse-shape and using those samples in reverse order as the coefficients of the filter.

When a particular frequency response is desired, several different design methods are common: 1. Window design method 2. Frequency Sampling method 3. Weighted least squares design

**Window Design Method**

In the window design method, one first designs an ideal IIR filter and then truncates the infinite impulse response by multiplying it with a finite length window function. The result is a finite impulse response filter whose frequency response is modified from that of the IIR filter

**Table: Commonly used window function characteristics**

| Window Name | Transition Approximate | Width Exact values | Min. Stop band Attenuation | Matlab  Command |
| --- | --- | --- | --- | --- |
| Rectangular |  |  | 21dB | **B = FIR1(N,Wn,boxcar)** |
| Bartlett |  |  | 25dB | **B = FIR1(N,Wn,bartlett)** |
| Hanning |  |  | 44dB | **B = FIR1(N,Wn,hanning)** |
| Hamming |  |  | 53dB | **B= FIR1(N,Wn,hamming)** |
| Blackman |  |  | 74dB | **B= FIR1(N,Wn,blackman)** |

**FIR filter using Hamming window**

**Method 1:**

**FIR ( LPF using hamming window)**

clc;

clear all;

close all;

wp= input('Enter the pass band edge (rad)= ');

ws= input('Enter the stop band edge (rad)= ');

ks= input('Enter the stop band attenuation (dB)= ');

%If 43<Ks<54 choose hamming window.

%To select N,order of filter.

N= (2\*pi\*4)/(ws-wp) % k=4 for Hamming window.

N= ceil(N) %To round-off N to the next integer.

r = rem(N,2) %Choose odd N.

if(r==0)

N=N+1;

end

wc=(wp+(ws-wp)/2)./pi

% To obtain h(n)

h= fir1(N-1,wc,hamming(N))

% Frequency response

freqz(h,1); % 1 is the normalized frequency

title('Frequency response of the lowpass digital FIR filter')

figure(2), stem(h);

title('Impulse response of the lowpass digital FIR filter')

**Method-2**

1. **Design Low Pass filter by using Hamming window**

**Hamming Window:**

The impulse response of an N – term Hamming window is defined as follows:

Text

Description automatically generated

The low pass filter equation is given by,



Where  **= **

The FIR filter coefficients are given by,

Graphical user interface, application

Description automatically generated

**Program: Plot:**

Chart, scatter chart

Description automatically generated

clc; close all;

clear all;

wc=input('enter the cut off freq');

N=input('enter order of the filter');

a=(N-1)/2;

n=0:1:N-1;

hd=sin(wc1\*(n-a+.001))./(pi\*(n-a+.001))

wn=hamming(N);

hn=hd.\*wn'

freqz(hn,1);

**Output:**

enter the cut off freq1

enter order of the filter7

hd = 0.0151 0.1449 0.2679 0.3183 0.2678 0.1446 0.0149

hn = 0.0012 0.0449 0.2063 0.3183 0.2062 0.0448 0.0012

1. **Design High Pass filter by using Hamming window**

**Hamming Window:**

The impulse response of an N – term Hamming window is defined as follows:

Text

Description automatically generated

The high pass filter equation is given by,



Where  **= **

The FIR filter coefficients are given by,

A picture containing graphical user interface

Description automatically generated

**Program:**

clc; close all; clear all;

wc=input('enter the cut off freq');

N=input('enter order of the filter');

a=(N-1)/2;

n=0:1:N-1;

hd=(sin(pi\*(n-a+.001))-sin(wc\*(n-a+.001)))./(pi\*(n-a+.001))

wn=hamming(N);

hn=hd.\*wn'

freqz(hn,1);

**Output:**

enter the cut off freq1

enter order of the filter7

hd =-0.0147 -0.1454 -0.2669 0.6817 -0.2688 -0.1441 -0.0152

hn = -0.0012 -0.0451 -0.2055 0.6817 -0.2069 -0.0447 -0.0012

**Plot:**

**Chart

Description automatically generated**

1. **Design Band Pass filter by using Hamming window**

**Hamming Window:**

The impulse response of an N – term Hamming window is defined as follows:

Text

Description automatically generated

The band pass filter equation is given by,



Where  **= **

The FIR filter coefficients are given by,

Text

Description automatically generated

**Program:**

clc; close all; clear all;

wc1=input('enter the lower cut off freq');

wc2=input('enter the upper cut off freq');

N=input('enter order of the filter');

a=(N-1)/2;

n=0:1:N-1;

hd=(sin(wc2\*(n-a+.001))-sin(wc1\*(n-a+.001)))./(pi\*(n-a+.001))

wn=hamming(N);

hn=hd.\*wn'

freqz(hn,1);

**Output:**

enter the lower cut off freq1

enter the upper cut off freq2

enter order of the filter7

hd = -0.0449 -0.2652 0.0220 0.3183 0.0211 -0.2652 -0.0443

hn = -0.0036 -0.0822 0.0170 0.3183 0.0163 -0.0822 -0.0035

**Plot:**

**Chart, scatter chart

Description automatically generated**

1. **Design Band Reject filter by using Hamming window**

**Hamming Window:**

The impulse response of an N – term Hamming window is defined as follows:

Text

Description automatically generated

The band rejection filter equation is given by,



Where  **= **

The FIR filter coefficients are given by,

Text

Description automatically generated

**Program:**

clc; close all; clear all;

wc1=input('enter the lower cut off freq');

wc2=input('enter the upper cut off freq');

N=input('enter order of the filter');

a=(N-1)/2;

n=0:1:N-1;

hd=(sin(wc1\*(n-a+.001))-sin(wc2\*(n-a+.001))+sin(pi\*(n-a+.001)))./(pi\*(n-a+.001))

wn=hamming(N);

hn=hd.\*wn'

freqz(hn,1);

**Output:**

enter the lower cut off freq1

enter the upper cut off freq2

enter order of the filter7

hd =0.0453 0.2647 -0.0210 0.6817 -0.0221 0.2657 0.0440

hn = 0.0036 0.0820 -0.0162 0.6817 -0.0170 0.0824 0.0035

**Plot:**

**Chart, scatter chart

Description automatically generated**

**VIVA QUESTIONS:**

1. Define filter.
2. What are the different types of filters?
3. Difference between IIR and FIR filters?
4. Differentiate ideal filter and practical filter responses.
5. What is the filter specifications required to design the analog filters?
6. What is meant by frequency response of filter?
7. What is meant by magnitude response?
8. What is meant by phase response?
9. Difference between FIR low pass filter and high pass filter.

# Experiment No. 12

# **Part B: EXPERIMENTS USING DSP PROCESSOR**

**Procedure for execution in TMS3206713 Simulator**

* Open CCS Studio Setup3.1Graphical user interface, application

  Description automatically generated
* Select Family 🡪67xx

# Graphical user interface, text, application Description automatically generated

* Platform 🡪 SimulatorGraphical user interface, text, application

  Description automatically generated
* Select 6713 Device cycle accurate simulator

Graphical user interface, text, application, email

Description automatically generated

Select Little endian. If little endian is not selected, building/linking error can occur. Add it to the left panel. Save and quit.

Graphical user interface, application, Word

Description automatically generated

Project🡪New 🡪Project Name🡪 Location(Location of project) 🡪 Project type (.out Executable) 🡪 Target (TMS320C67xx)Graphical user interface, application

Description automatically generated

Write the code in a new source file. Save it in the project folder with .C file format.

Graphical user interface, application

Description automatically generated

* Write the code in a new source file. Save it in the project folder with .C file format.
* Add this to the project. Project will be having .pjt extension. Right click on .pjt file created, add the .c file you have written.

Graphical user interface, text, application

Description automatically generated

* Two other files are to be added to project. One is library file (\*.lib) and other is Linker command file (\*.cmd)
* Add rts6713.lib C:\CCStudio\_v3.1\C6000\cgtools\librts6700.lib
* Add hello.cmd C:\CCStudio\_v3.1\tutorial\dsk6713\hello1\hello.cmd
* Debug🡪 Build
* File🡪 Load Program(Often this is the most comman mistake to forget this..!) Load the .out file which is in the DEBUG folder of the project folder. Select this and open.
* Debug 🡪 Debug Run

**Procedure for execution in TMS320DSK6713 kit**

* CCS Studio Setup v3.1 🡪 Family (67xx) 🡪 Platform (dsk) 🡪 Endianness 🡪Little endian🡪 Add it to panel. Click on 6713dsk, save and quit.
* Connect the power card to the DSP kit.
* Connect the data cable 🡪 USB from PC
* After getting the project window, DEBUG🡪 CONNECT.
* Rest of the procedure is same as compared to simulator running.

# Experiment No. 13a

## **Linear convolution of two given sequences.**

**Aim:** To obtain convolution of two finite duration sequences.

**Theory:** Convolution is the integral concatenation of two signals. It has many applications in numerous areas of signal processing. The most popular application is the determination of the output signal of a linear time-invariant system by convolving the input signal with impulse response of the system.

**Mathematical formula:**

The linear convolution of two given continuous time signals x(t) and h(t) is defined by



For discrete time signals x(n) and h(n), is defined by



Where  = input signal &  = impulse response of the system. In linear convolution the length of the output sequence is,

Length [] = Length [] + Length [] – 1

**Program:**

/\* To find the linear convolution of given sequnces \*/

#include<stdio.h>

#define LENGHT1 6 /\*Lenght of i/p samples sequence\*/

#define LENGHT2 4 /\*Lenght of impulse response Co-efficients \*/

int x[9]={1,2,3,4,5,6,0,0,0}; /\*Input Signal Samples\*/

int h[9]={1,2,3,4,0,0,0,0,0}; /\*Impulse Response Coefficients\*/

int y[LENGHT1+LENGHT2-1];

main()

{

int i=0,j;

for(i=0;i<(LENGHT1+LENGHT2-1);i++)

{

y[i]=0;

for(j=0;j<=i;j++)

y[i]+=x[j]\*h[i-j];

}

for(i=0;i<(LENGHT1+LENGHT2-1);i++)

printf("%d\n",y[i]);

}

# Experiment No. 13b

## **Circular convolution of two given sequences.**

**Program:**

/\* program to implement circular convolution \*/

#include<stdio.h>

int m,n,x[30],h[30],y[30],i,j, k,x2[30],a[30];

void main()

{

printf(" enter the length of the first sequence\n");

scanf("%d",&m);

printf(" enter the length of the second sequence\n");

scanf("%d",&n);

printf(" enter the first sequence\n");

for(i=0;i<m;i++)

scanf("%d",&x[i]);

printf(" enter the second sequence\n");

for(j=0;j<n;j++)

scanf("%d",&h[j]);

if(m-n!=0) /\*If length of both sequences are not equal\*/

{

if(m>n) /\* Pad the smaller sequence with zero\*/

{

for(i=n;i<m;i++)

h[i]=0;

n=m;

}

for(i=m;i<n;i++)

x[i]=0;

m=n;

}

y[0]=0;

a[0]=h[0];

for(j=1;j<n;j++) /\*folding h(n) to h(-n)\*/

a[j]=h[n-j];

/\*Circular convolution\*/

for(i=0;i<n;i++)

y[0]+=x[i]\*a[i];

for(k=1;k<n;k++)

{

y[k]=0;

/\*circular shift\*/

for(j=1;j<n;j++)

x2[j]=a[j-1];

x2[0]=a[n-1];

for(i=0;i<n;i++)

{

a[i]=x2[i];

y[k]+=x[i]\*x2[i];

}

}

/\*displaying the result\*/

printf(" the circular convolution is\n");

for(i=0;i<n;i++)

printf("%d \t",y[i]);

}