2) C Program to implement Circular Convolution of two given sequences:

#include<stdio.h>

void main()

{

float h[20],x[20],y[20],sum;

N=4;

x[0]=1.0, x[1]=2.0, x[2]=3.0, x[3]=4.0;

h[0]=2.0, h[1]=1.0, h[2]=2.0, h[3]=1.0;

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

{

sum=0.0;

for(k=0; k <N; k++)

{

if((m-k)>= 0) n=m-k;

else n= m-k+N;

sum += x[k] \* h[n];

}

y[m]=sum;

}

}

E7: DFT AND IDFT

Aim: program to find 8-point DFT of a real sequence

#include <stdio.h>

#include <math.h>

void main(void)

{

float x[10], real\_X[10],img\_X[10], mag\_X[10]

int k, n, N=8;

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

{

x[n]=n+1;

}

for(k 0;k<N; k++)

{

real\_X[k]=0.0;

img\_X[k]=0.0;

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

{

real\_X[k]=real\_X[k]+x[n]\*cos(2\*3.1415 n\*k/N);

img\_X[k]=img\_X[k]+x[n]\*-1\*sin(2\*3.1415\*n\*k/N);

}

mag X[k]=sqrt(real\_X[k]\*real\_X[k]+img\_X[k]\*img X[k]);

}

}

2) Aim: C program to find 4-point DFT of a complex valued sequence

#include <stdio.h>

#include <math.h>

void main(void)

{

float real\_x[4], img\_x[4],real\_X[4], img \_X[4], mag\_X[4];

int k, n, N=4;

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

{

Real\_X[n]=n+1;

}

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

{ img\_x[n]=n+1;

}

For(k=0;k<N;k++)

{ real\_X[k]=0.0;

img\_X[k]=0.0;

for(n-0;n<N;n++)

{real\_X[k]=real\_X[k}+((real\_x{n]\*cos(2\*3.1415\*n\*k/N) )+

(img\_x[n]\*sin(2\*3.1415\*n\*k/N)));

img\_X[k]=img\_X[k]+((img x[n]\*cos(2\*3.1415\*n\*k/N))-

(real\_x[n]"sin(2\*3.1415\*n\*k/N)));

}

mag\_X[k]=sqrt(real\_X[K]\*real\_X[k]+img\_X[K]\*img\_X[k]);

}

}

3)WRITE A C PROGRAM TO F8IND 8-POINT IDFT OF A REAL SEQUENCE

#include<stdio.h

#include math.h>

void main(void)

{

float x[10], real\_X[10].img\_X[10],pi=3.1415926;

int k, n ,N=8;

printf(“enter the IDFT of the sequence\n");

for(k=0;k<N;k++)

{

Scanf(“%f”,&real\_X[k]);

Scanf(“%f", &img\_X[k]);

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

{

x[n]=0.0;

for{k=0;k<N:k++)

{

x[n]=x[n]+real\_X[k]\*cos(2\*pi\*n\*k/N)- img\_X[k]\*sin(2\*pi\*n\*k/N);

}

x[n]=x[n]/N;}}

E2:Verification of Sampling Theorem and AM Wave

1. Aim: To verify sampling theorem for a single tone signal

Close all; clc; clear all;

t=-1:0.001:1;

fm1= input ('Enter the frequency of continuous time signal=’);

x=cos(2\*pi\*fm1\*t);

subplot(311)

plot(t,x);

title(‘Continuous time signal’)

fs=64;n=0:63;

xn1=cos(2\*pi\*n\*fm1/fs);

xm1=abs(fft(xn1,length(n)));

subplot(312)

stem(n,xn1);

hold on;

plot(n,xn1);

title(‘sampling frequency greater than Nyquist Rate’);

subplot(313)

stem(n,xml)

title(‘DFT of signal sampled at greater than Nyquist rate');

figure

fm2=input(‘Enter the frequency of continuous time signal’);

x=cos(2\*pi\*fm2\*t); % continuous time signal

subplot (311)

plot(t,x);

title (‘Continuous time signal’)

xn2=cos(2\*pi\*n\*fm2/fs);

xm2=abs(fft(xn2,length(n)));

subplot(312)

stem(n,xn2);

hold on

plot(n,xn2);

title(‘sampling frequency equal to Nyquist Rate’);

subplot(313);

stem(n,xm2)

title(‘DFT of signal sampled at Nyquist rate');

figure;

fm3=input(‘Enter the frequency of continuous time signal=’);

x=cos(2\*pi\*fm3\*t);

subplot(311):

plot(t,x);

title(‘Continuous time signal')

xn3=cos(2\*pi\*n\* fm3/(fs));

xm3-abs(fft(xn3,length(n)));

subplot(312)

stem(n,xn3);

hold on

plot(n,xn3);

title('sampling frequency less than Nyquist Rate')

subplot(313);

stem(n,xm3)

title(‘DFT of signal sampled at less than Nyquist rate’);

2) sampling theorem verification for multitone signal

close all; clc; clear all;

t=-1:0.001:1;

fm1=input('Enter the frequency of first continuous time signal=');

fm2=input(‘Enter the frequency of second continuous time signal=’):

x=cos(2\*pi\*fm1\*t)+cos(2\*pi\*fm2\*t) ;

subplot(311)

plot(t,x);

title(‘Continuous time signal')

fs=64;n=0:63;

xnl=cos(2\*pi\*n\*fm1/s) cos(2\*pi\*n\*fm2/fs);

subplot(312)

stem(n,nx1);

hold on;

plot(n,xn1);

title('sampling frequency greater than Nyquist Rate')

subplot(313)

stem(n,xm1)

title(‘DFT of signal sampled at greater than Nyquist rate');

figure;

fm1=input(‘Enter the frequency of first continuous time signal=’);

fm2=input(Enter the frequency of second continuous time signal=’);

x=cos(2\*pi\*fm1\*t)+cos(2\*pi\*fm2\*t);

subplot(311)

plot(t,x):

title('Continuous time signal')

xn2=cos(2\*pi\*n\*fm1/fs) +cos(2\*pi n\*fm2/fs);

xm2=abs(fft(xn2,length(n)));

subplot(312)

stem(n,xn2)

hold on

plot(n,xn2);

title(‘sampling frequency equal to Nyquist Rate’);

subplot(313);

stem(n,xm2)

title(‘DIFT of signal sampled at Nyquist rate');

figure;

fm1=input(‘Enter the frequency of first continuous time signal=’ );

fm2=input(‘Enter the frequency of second continuous time signal=’);

x=cos(2\*pi\*fm1\*t)+cos(2\*pi\*fm2\*t);

subplot(311);

plot(t,x);

title(‘Continuous one signal’)

xn3=cos(2\*pi\*n\*fm1/fs)+cos(2\*pi\*n\*fm2/fs);

xm3=abs(fft(xn3,length(n)));

subplot(312);

stem(n,xn3);

hold on

title('sampling frequency less than Nyquist Rate’)

subplot(313);

stem(n,xm3);

title(‘DFT of signal sampled at less than Nyquist rate’);

3)Aim: To Generate Amplitude Modulated Wave and to display the spectrum

close all; clc; clear all

n-0:127;

fm=input(‘Enter the frequency of modulating signal’);

fc=input(‘Enter the frequency of carrier signal=’);

Am=2;

Ac=5;

x1=Am\*cos(2\*pi\*fm\*n/128);

subplot(211);

plot(n,x1,’m’):

hold on;

x2=Ac\*cos(2\*pi\*fc\*n/128):

plot(n,x2,'k');

X=x1.\*x2/Ac+x2;

subplot(212)

plot(n,x,’b')

figure;

X=abs(fft(x,128)); stem(n,X);