**To generate Amplitude Modulated Wave and to display the spectrum**

% Amplitude Modulation

close all;

clc;

clear all;

n=0:127;

% fm and fc must be less than Fs/2=64

% fc>fm

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);

// C program to implement Linear Convolution of two given sequences

// Note: Linear convolution (size of the two sequences may be equal or different)

#include<stdio.h>

void main()

{

float x1[10],x2[10],y[20];

int L=5,M=4,n,m,a,b,c,min,max;

x1[0]=1;

x1[1]=2;

x1[2]=2;

x1[3]=3;

x1[4]=2;

x2[0]=1;

x2[1]=2;

x2[2]=3;

x2[3]=3;

if(L>=M)

{

c=M-L;

max=L;

min=M;

for(n=0;n<L+M-1;n++)

{

if(n-max<c)

a=0;

else

a=n-min+1;

if(n>=max)

b=max-1;

else

b=n;

for(y[n]=0,m=a;m<=b;m++)

y[n]+=x1[m]\*x2[n-m];

}

}

else

{

c=L-M;

max=M;

min=L;

for(n=0;n<L+M-1;n++)

{

if(n<min)

b=n;

else

b=min-1;

for(y[n]=0,m=0;m<=b;m++)

y[n]+=x1[m]\*x2[n-m];

}

}

}

// c program to implement Circular Convolution of two given sequences

#include<stdio.h>

void main()

{

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

int N,n,k,m;

/\* Length of the sequences \*/

N=4;

// First Sequence

x[0]=1.0;

x[1]=2.0;

x[2]=3.0;

x[3]=4.0;

// Second Sequences

h[0]=2.0;

h[1]=1.0;

h[2]=2.0;

h[3]=1.0;

// Calculates circular convolution

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

{

sum=0.0;

// computation of one value in circular convolution

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

{

// Modulo index

if((m-k)>=0)

n=m-k;

else

n=m-k+N;

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

}

y[m]=sum;

}

}

// C 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]);

}

}

// 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[10];

int k,n,N=4;

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

{

real\_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]);

}

}

// C program to find 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;

}

}

/\* C program to plot the impulse response and the output of the system for the input x(n)=u(n)-u(n-10) , for the discrete system given by the difference

equation y(n)-0.9y(n-1)=x(n) \*/

#include<stdio.h>

#include<math.h>>

void main()

{

float a[5],b[5],x[25],y[25],h[25],sumXn\_k,sumYn\_k;

int N,M,k,L,n;

// Number of denominator coefficients and ak (a subscript k)

N=1;

a[1]=-0.9;

// Number of numerator coefficients and bk (b subscript k)

M=1;

b[0]=1;

// Number of input samples and the input sequences

L=25;

// Impulse sequence from n=0 to n=24

x[0]=1.0;

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

{

x[k]=0.0;

}

// Computation of h(n) that is the impulse response

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

{

sumYn\_k=0;

sumXn\_k=0;

// computation of a1\*y(n-1)+a2\*y(n-2)+a3\*y(n-3)+....

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

{

sumYn\_k+=a[k]\*h[n-k];

}

// computation of b0\*x(n)+b1\*x(n-1)+b2\*x(n-2)+.....

for(k=0;(k<=n)&&(k<M);k++)

{

sumXn\_k+=b[k]\*x[n-k];

}

h[n]=-sumYn\_k+sumXn\_k;

}

// Number of input samples and the input sequence x(n)=u(n)-u(n-10)

L=25;

// input sequence from n=0 to n=24

for(k=0;k<10;k++)

{

x[k]=1.0;

}

for(k=10;k<L;k++)

{

x[k]=0.0;

}

// computation of y(n) that is the output sequence

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

{

sumYn\_k=0;

sumXn\_k=0;

// computation of a1\*y(n-1)+a2\*y(n-2)+a3\*y(n-3)+....

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

{

sumYn\_k+=a[k]\*y[n-k];

}

// computation of b0\*x(n)+b1\*x(n-1)+b2\*x(n-2)+.....

for(k=0;(k<=n)&&(k<M);k++)

{

sumXn\_k+=b[k]\*x[n-k];

}

y[n]=-sumYn\_k+sumXn\_k;

}

}

// C program to plot the impulse response of a second order system given by the difference equation y(n)-y(n-1)+0.9y(n-2)=x(n)

/\* Impulse Response of a given system

Solve the given difference equation y(n)-y(n-1)+0.9y(n-2)=x(n)

Calculate the impulse response h(n) at n=0,1,...,100

Calculate the unit step response s(n) at n=0,1,....,100

b=[1] and a=[1 -1 0.9] \*/

#include<stdio.h>

#include<math.h>>

void main()

{

float a[5],b[5],x[25],h[25],sumXn\_k,sumYn\_k;

int N,M,k,L,n;

// Number of denominator coefficients and ak (a subscript k)

N=2;

a[1]=-1.0;

a[2]=0.9;

// Number of numerator coefficients and bk (b subscript k)

M=1;

b[0]=1;

// Number of input samples and the input sequences

L=25;

// Impulse sequence from n=0 to n=25

x[0]=1.0;

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

{

x[k]=0.0;

}

// Computation of h(n) that is the impulse response

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

{

sumYn\_k=0;

sumXn\_k=0;

// computation of a1\*y(n-1)+a2\*y(n-2)+a3\*y(n-3)+....

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

{

sumYn\_k+=a[k]\*h[n-k];

}

// computation of b0\*x(n)+b1\*x(n-1)+b2\*x(n-2)+.....

for(k=0;(k<=n)&&(k<M);k++)

{

sumXn\_k+=b[k]\*x[n-k];

}

h[n]=-sumYn\_k+sumXn\_k;

}

}