**Experiment :- 06**

**Aim:-** Compute the N-point IDFT of the following k-point DFT sequence:-

V[k] = {k/N, 0 <= k <= N-1

0, Otherwise}.

**Software Used**:- Matlab 2018b

**Program**:-

N = input("Enter the value of N = ") %console input for N point DFT

t = -2\*N:2\*N;

v = zeros(size(t));

for i = 0:N; %generating the N point DFT

v(i+1+2\*N) = i/N;

end

subplot(4,1,1)

stem(t,v) %plotting the DFT

xlabel('k')

ylabel('Amplitude')

title('DFT -Shahbaz')

grid on;

subplot(4,1,2)

f = fftshift(ifft(v,length(t))); %inverse DFT of the given DFT signal

stem(t,abs(f)) %plotting the inverse DFT

xlabel('n')

ylabel('Amplitude')

title('IDFT -Shahbaz')

grid on;

subplot(4,1,3)

stem(t,imag(f)) %plotting the imaginary part of inverse DFT

xlabel('Angle')

ylabel('Amplitude')

title('Imaginary Part -Shahbaz')

grid on;

subplot(4,1,4)

stem(t,real(f)) %plotting the real part of inverse DFT

xlabel('n')

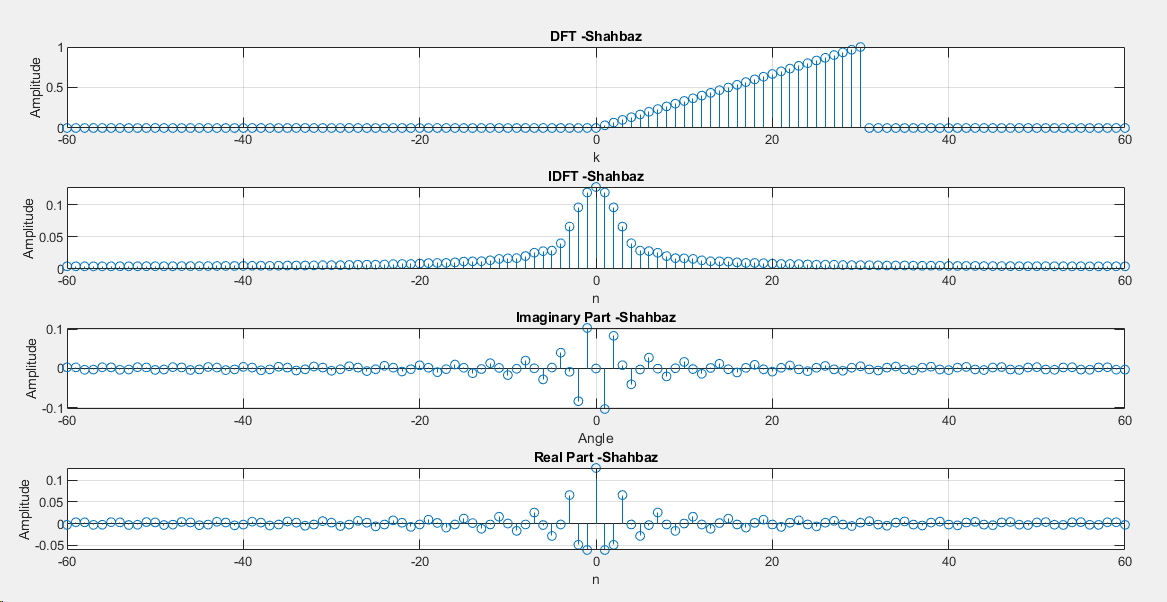
ylabel('Amplitude')

title('Real Part -Shahbaz')

grid on;

**Result:-**

N-point IDFT of the given equation was calculated and verified as following-



**Conclusion:-**

**Experiment:- 07**

**Aim:-** Linear convolution of two discrete signals using DFT method.

**Software Used**:- Matlab 2018b

**Program**:-

y1 = [0 1 2 3 4 5 6 7 8 9]; %generating the first signal

y2 = [1 3 5 7 9]; %generating the second signal

y1dash=y1; %keeping a copy of original signal

y2dash=y2;

n = length(y1); %finding the length of signal

m = length(y2);

L = n+m-1;

k1 = 0:n-1; %generating the time axis for the signal

k2 = 0:m-1;

k = 0:L-1;

figure(1);

subplot(3,1,1)

stem(k1,y1) %plotting the first signal

xlabel('n')

ylabel('Amplitude')

title('First Signal -Shahbaz')

grid on;

subplot(3,1,2)

stem(k2,y2) %plotting the second signal

xlabel('n')

ylabel('Amplitude')

title('Second Signal -Shahbaz')

grid on

y1 = [y1,zeros(1,L-n)] %zero padding to the signals

y2 = [y2,zeros(1,L-m)]

Yf1 = fftshift(fft(y1,L)); %DFT of zero padded signals

Yf2 = fftshift(fft(y2,L));

Y = Yf1.\*Yf2; %Multiplication of DFT signals

subplot(3,1,3)

stem(k,abs(Y)); %Plotting their products

xlabel('n')

ylabel('Amplitude')

title('Magnitude of DFT of given signals -Shahbaz')

grid on;

f = (ifft(Y,L)); %finding the inverse of product of two DFT signals

figure(2)

subplot(4,1,3)

stem(k,abs(f)) %plotting the inverse of DFT products

xlabel('n')

ylabel('Amplitude')

title('Convolution using DFT method -Shahbaz')

grid on;

subplot(4,1,1)

stem(k,imag(f)) %plotting its imaginary part

xlabel('Angle')

ylabel('Amplitude')

title('Imaginary Part of IDFT signal -Shahbaz')

grid on;

subplot(4,1,2)

stem(k,real(f)) %plotting its real part

xlabel('n')

ylabel('Amplitude')

title('Real Part of IDFT signal -Shahbaz')

grid on;

subplot(4,1,4)

a = conv(y1dash,y2dash); %finding the convolution in time domain

stem(k,a) %plotting the convolution

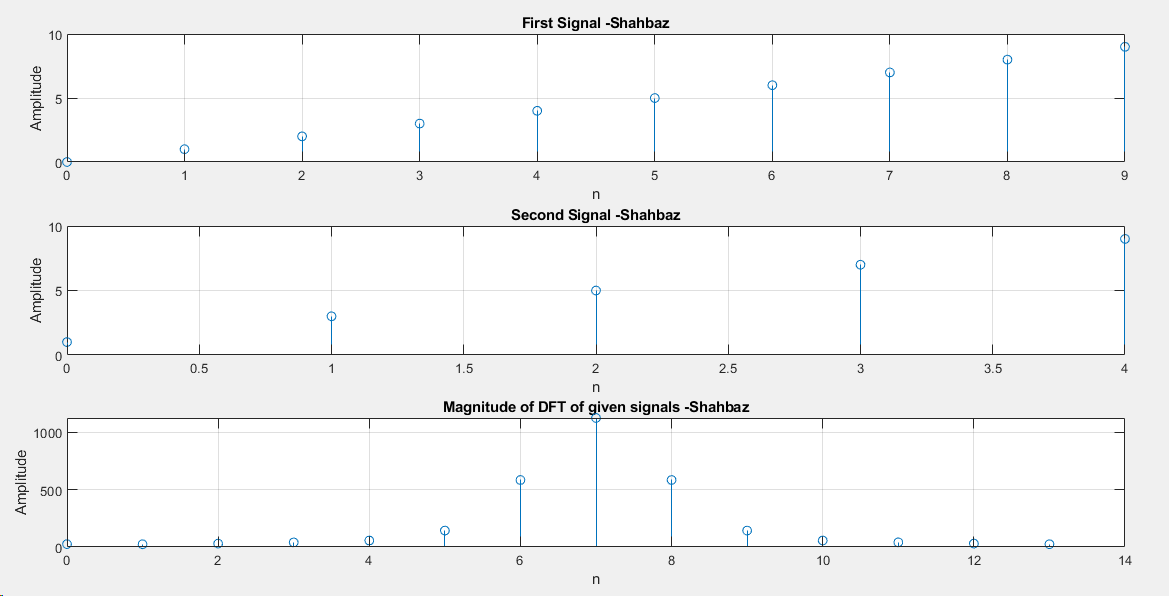
xlabel('n')

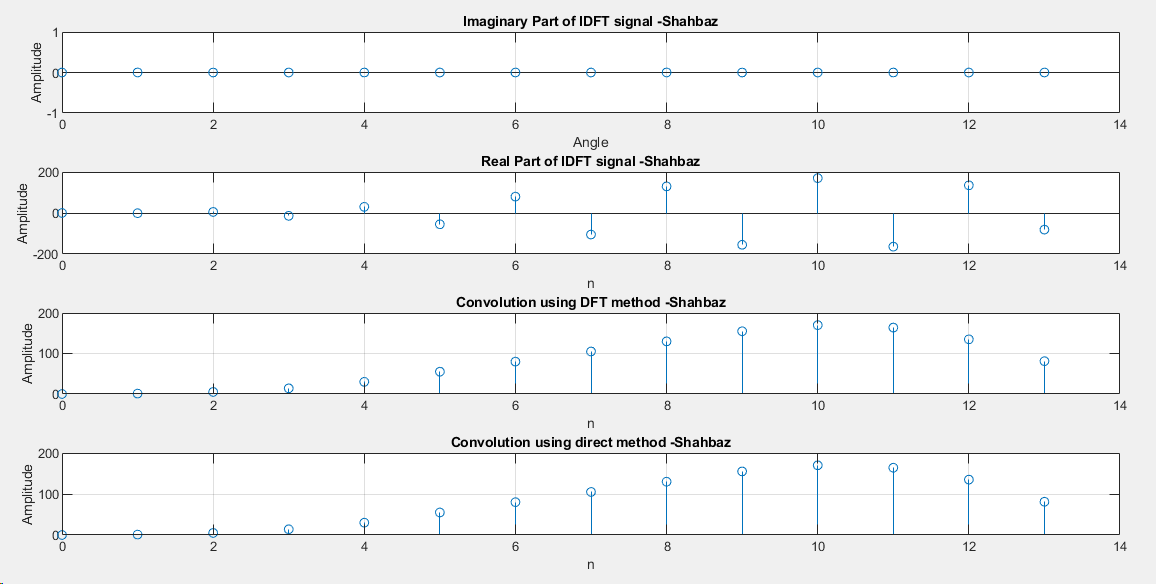
ylabel('Amplitude')

title('Convolution using direct method -Shahbaz')

grid on;

**Result:-** Convolution of two given signals was calculated using DFT method and verified with the following outputs-





**Conclusion:-**

**Experiment :- 08**

**Aim:-** Circular Convolution using-

1. Direct Command
2. Using Linear Convolution
3. Using Circular Convolution

**Software Used**:- Matlab 2018b

**Program**:-

function main() %main function

y1 = [0 1 2 3 4 5 6 7 8 9]; %generating the first signal

y2 = [1 3 5 7 9]; %generating the second signal

m = length(y1); %finding the length of signal

n = length(y2);

t = max(m,n); %finding the maximum length among two signals

k = 0:t-1;

kdash = 0:(2\*t-2);

k1 = 0:(m-1); %forming the time axis for signal

k2 = 0:(n-1);

subplot(5,1,1)

stem(k1,y1); %plotting the first signal

xlabel('n')

ylabel('Amplitude')

title('First Signal -Shahbaz')

grid on

subplot(5,1,2)

stem(k2,y2); %plotting the second signal

xlabel('n')

ylabel('Amplitude')

title('Second Signal -Shahbaz')

grid on

subplot(5,1,3)

stem(k,fftshift(cconv(y1,y2,t))) %calculating and plotting of circular conv using direct method

xlabel('n')

ylabel('Amplitude')

title('Circular Convolution using Direct Method -Shahbaz')

grid on

y1dash = [y1,zeros(1,t-m)]; %zero padding to both signals

y2dash = [y2,zeros(1,t-n)];

subplot(5,1,4)

stem(kdash,conv(y1dash,y2dash)) %calculating and plotting the linear conv of zero padded signal

xlabel('n')

ylabel('Amplitude')

title('Circular Convolution using Linear Convolution -Shahbaz')

grid on

if(m>n) %making both signal of equal length by padding zeros

y2 = [y2,zeros(1,m-n)];

else

y1 = [y1,zeros(1,n-m)];

end

for i = 1:length(y1)

r = rotate(y1,i-1); %rotating one of the signal

c(i) = sum(r.\*reverse(y2)) %multiplying the corresponding elements and summing it

end

t2 = 0:length(y1)-1;

subplot(4,1,4)

stem(t2,c) %plotting the final convolution matrix

xlabel('n')

ylabel('Amplitude')

title('Circular Convolution using Circular Method -Shahbaz')

grid on

end

function rev = reverse(y) %function to reverse a given row vector

x = 1;

for i = length(y):-1:1

rev(x) = y(i);

x = x+1;

end

end

function r = rotate(y,n) %function to rotate a row vector in a particular direction by ‘n’

n = rem(n,length(y)); %amount

r = [];

x = 1;

if(n>0)

for i = 1:length(y)

if((i+n)<=length(y))

r(i) = y(i+n);

else

r(i) = y(x);

x = x+1;

end

end

else

r(1) = y(length(y));

for i = 2:length(y)

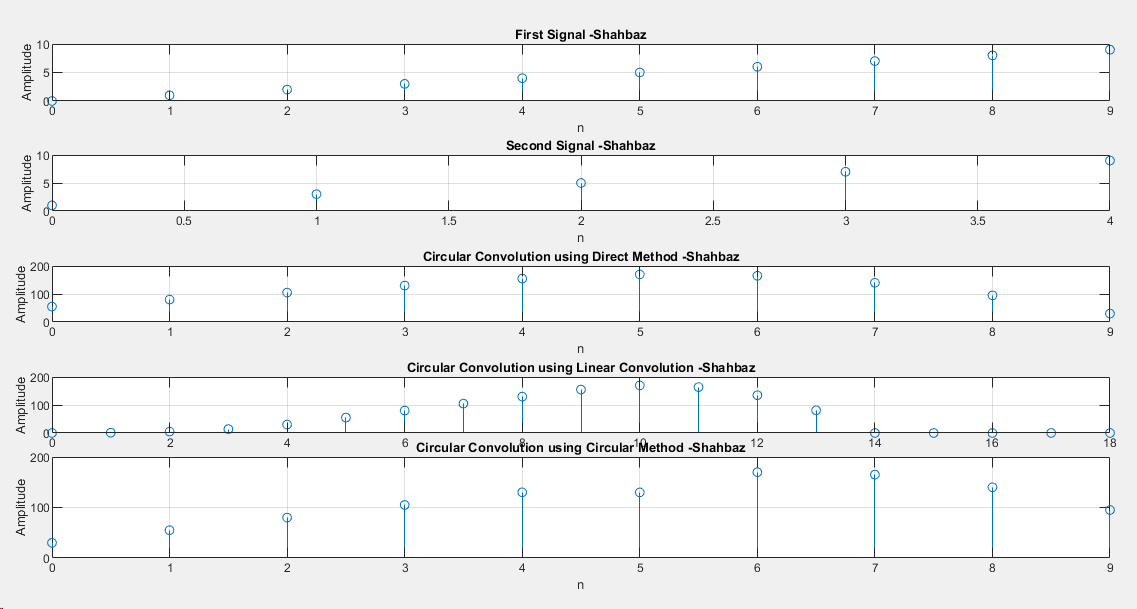
r(i) = y(i-1);

end

end

end

**Result:-** Circular convolution of two given signals was calculated and verified with the following outputs-



**Conclusion:-**

**Experiment:- 09**

**Aim:-** Determine the factored form of a given ration Z-transform, plot its poles and zeros, determine it ROCs and stability.

**Software Used**:- Matlab 2018b

**Program**:-

numerator = [2 16 44 56 32]; %numerator coefficients

denominator = [3 3 -15 18 -12]; %denominator coefficients

[z p k] = tf2zp(numerator,denominator); %finding the zero and pole from the given equation

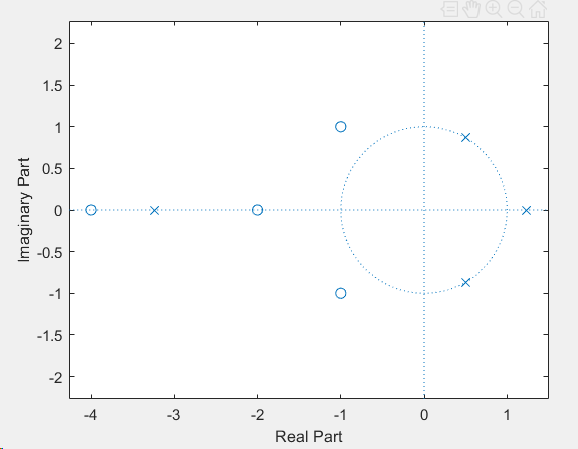
figure(1)

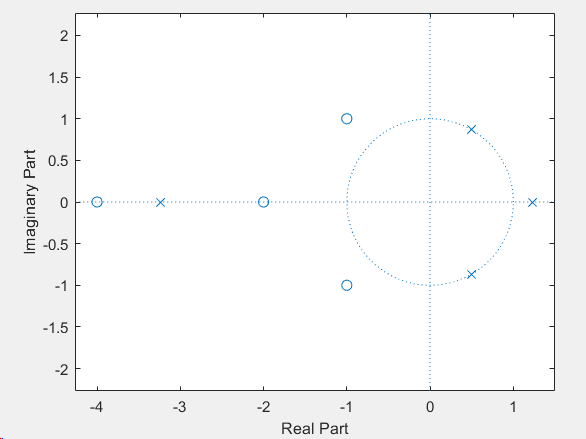
zplane(numerator,denominator); %plotting the equation on Z plane

figure(2)

zplane(z,p); %plotting the zeros and poles on Z plane

**Result:-** Poles and zeros of a given Z-transform was calculated and verified using the following outputs-





**Conclusion:-**

**Experiment :- 10**

**Aim:-** Power series expansion of rational Z transform.

**Software Used**:- Matlab 2018b

**Program**:-

L = input('Enter the length of output vector = '); %length of output vector input

num = input('Enter numerator coefficients = '); %numerator coefficients

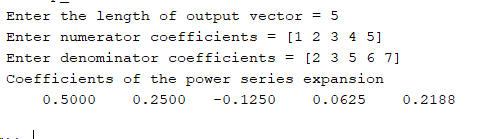
den = input('Enter denominator coefficients = '); %denominator coefficients

[y,t] = impz(num,den,L); %impulse response of system.

display('Coefficients of the power series expansion');

display(y'); %display impulse response

**Result:-** Power series of rational Z transform was calculated and verified as below-

****

**Conclusion:-**

**Experiment :- 11**

**Aim:-**  Finding the transfer function using its poles and zeros.

**Software Used**:- Matlab 2018b

**Program**:-

zr = input('Enter the zeros of transfer function '); %console input of zeros

po = input('Enter the poles of transfer function '); %console input of poles

k = input('Enter gain coefficients '); %gain coefficient

[num den] = zp2tf(zr',po',k); %converting zero pole to transfer function

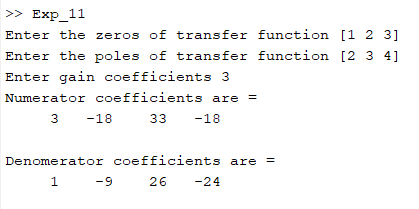
disp('Numerator coefficients are = ') %printing numerator coefficient

disp(num);

disp('Denominator coefficients are = ') %printing denominator coefficient

disp(den);

**Result:-** Transfer function for a given system was generated using its zeros, poles and gain and it was verified by the following output.

****

**Conclusion:-**

**Experiment :- 12**

**Aim:-** Schur Cohn stability test.

**Software Used**:- Matlab 2018b

**Program**:-

deno = input("Enter denominator coefficients = ") %console input of denominator coefficients

deno = deno/deno(1); %making the coefficient of highest power as 1

a = deno; %saving the coefficients into another variable

flag = true;

m = length(deno); %calculating the length of coefficients

for i = m:-1:1

k = a(i); %calculating ‘k’ for different ‘a1,a2,etc..’

if(k>1)

flag = false; %if k>1 then it is unstable system

break

end

b = flip(a); %calculating b by changing the coefficients of a

a =(a - k\*b)/(1-k\*k); %calculating a(n-1) coefficients

a = a(1:i-1); %removing the zero coefficient

end

if(flag)

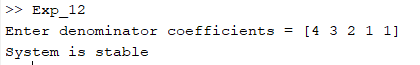
display(‘System is stable’) %printing the result

else

display(‘System is unstable’)

end

**Result:-** Stability of a given system was tested using Schur Cohn stability test and it was verified by the following output.

****

**Conclusion:-**