**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

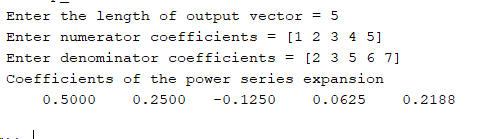
den = input('Enter denominator coefficients = ');           %denominator coefficients

[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

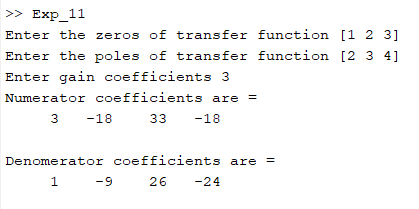
disp('Numerator coefficients are = ')            %printing numerator coefficient

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)

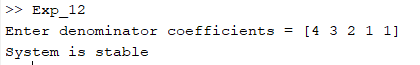
    display(‘System is stable’)                %printing the result

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:-**

**Experiment :- 13**

**Aim:-** To determine frequency response of discrete LTI system.  
    a.) 1 + 2z + 3z2   
    b.) 11 + 2z + 3z2

**Software Used**:- Matlab 2019a

**Program**:-

k = input('Enter number of frequency points = ');    %taking frequency point from user

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

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

w = 0:pi/(k-1):pi;                    %forming k point frequency vector

h = freqz(num,deno,w);                %calculating frequency response

subplot(2,1,1)

stem(w/pi,abs(h));                %plotting the magnitude of frequency response

grid on

title('Magnitude Spectrum')

xlabel('\omega/\pi')

ylabel('magnitude')

subplot(2,1,2)

plot(w/pi,angle(h));                %plotting the phase of frequency response

grid on

title('Phase Spectrum')

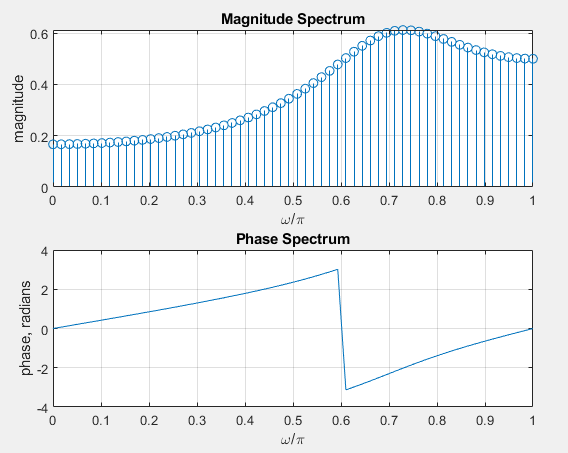
xlabel('\omega/\pi')

ylabel('phase, radians')

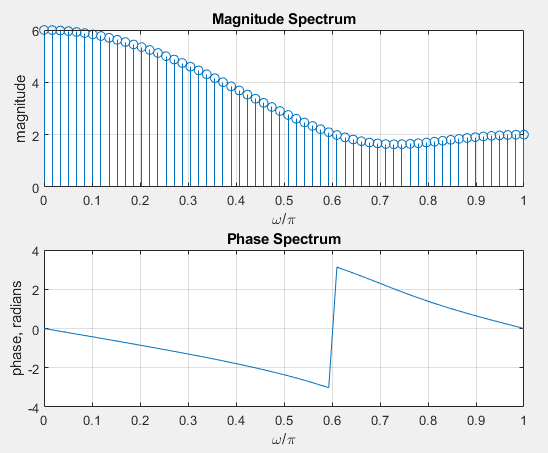
**Result:-** Frequency response of a given discrete LTI system was determined and it was verified by the following output.

Number of frequency points = 60.

a.)



b.)



**Conclusion:-**

**Experiment :- 14**

**Aim:-** Design of FIR lowpass and highpass filter using Rectangular Window.

**Software Used**:- Matlab 2019a

**Program**:-

rp = input('Enter passband ripple = ');        %taking passband ripple from user

rs = input('Enter stopband ripple = ');        %taking stopband ripple from user

fp = input('Enter passband frequency = ');        %taking passband frequency from user

fs = input('Enter stopband frequency = ');        %taking stopband frequency from user

f = input('Enter sampling frequency = ');        %%taking sampling frequency from user

wp = 2\*fp/f;                    %calculating the omega

num = -20\*log10(sqrt(rp\*rs))-13;            %converting to decibel

den = 14.6\*(fs-fp)/f;

n = ceil(num/den);                %forming the n vector

n1 = n+1;

if(rem(n,2)~=0)

    n1 = n;

    n = n-1;

end

y = boxcar(n1);                    %using boxcar window

b = fir1(n,wp,y);                    %designing the filter

[h,o] = freqz(b,1,256);                %finding its frequency response

m = 20\*log10(abs(h));                %finding its decibel form

figure(1)

subplot(2,1,1)                    %plottin the LPF filter

plot(o/pi,m)

title('Boxcar (LPF) ')

ylabel('Gain(dB)');

xlabel('Normalized freq');

b = fir1(n,wp,'high',y);                %designing the HPF filter

[h,o] = freqz(b,1,256);                %finding its frequency response

m = 20\*log10(abs(h));                %its decibel form

subplot(2,1,2)

plot(o/pi,m)

title('Boxcar (HPF) ')            %plotting the output

ylabel('Gain(dB)');

xlabel('Normalized freq');

y = hanning(n1);                    %repeating the above step for hanning window filter

b = fir1(n,wp,y);

[h,o] = freqz(b,1,256);

m = 20\*log10(abs(h));

figure(2)

subplot(2,1,1)

plot(o/pi,m)

title('Hanning (LPF) ')

ylabel('Gain(dB)');

xlabel('Normalized freq');

b = fir1(n,wp,'high',y);

[h,o] = freqz(b,1,256);

m = 20\*log10(abs(h));

subplot(2,1,2)

plot(o/pi,m)

title('Hanning (HPF) ')

ylabel('Gain(dB)');

xlabel('Normalized freq');

y = bartlett(n1);                    %repeating the above step for bartlett window filter

b = fir1(n,wp,y);

[h,o] = freqz(b,1,256);

m = 20\*log10(abs(h));

figure(4)

subplot(2,1,1)

plot(o/pi,m)

title('Bartlett (LPF) ')

ylabel('Gain(dB)');

xlabel('Normalized freq');

b = fir1(n,wp,'high',y);

[h,o] = freqz(b,1,256);

m = 20\*log10(abs(h));

subplot(2,1,2)

plot(o/pi,m)

title('Bartlett (HPF) ')

ylabel('Gain(dB)');

xlabel('Normalized freq');

y = blackman(n1);               %repeating the above step for blackman window filter

b = fir1(n,wp,y);

[h,o] = freqz(b,1,256);

m = 20\*log10(abs(h));

figure(5)

subplot(2,1,1)

plot(o/pi,m)

title('Blackman (LPF) ')

ylabel('Gain(dB)');

xlabel('Normalized freq');

b = fir1(n,wp,'high',y);

[h,o] = freqz(b,1,256);

m = 20\*log10(abs(h));

subplot(2,1,2)

plot(o/pi,m)

title('Blackman (HPF) ')

ylabel('Gain(dB)');

xlabel('Normalized freq');

y = hamming(n1);                %repeating the above step for hamming window filter

b = fir1(n,wp,y);

[h,o] = freqz(b,1,256);

m = 20\*log10(abs(h));

figure(3)

subplot(2,1,1)

plot(o/pi,m)

title('Hamming (LPF) ')

ylabel('Gain(dB)');

xlabel('Normalized freq');

b = fir1(n,wp,'high',y);

[h,o] = freqz(b,1,256);

m = 20\*log10(abs(h));

subplot(2,1,2)

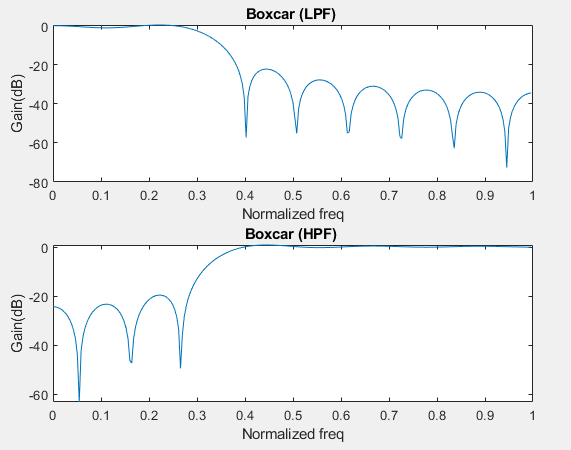
plot(o/pi,m)

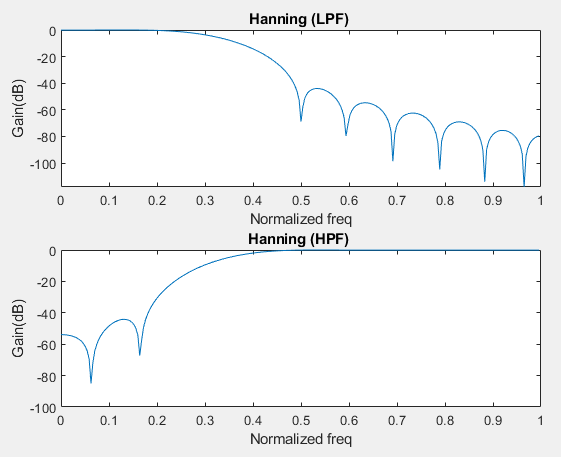
title('Hamming (HPF) ')

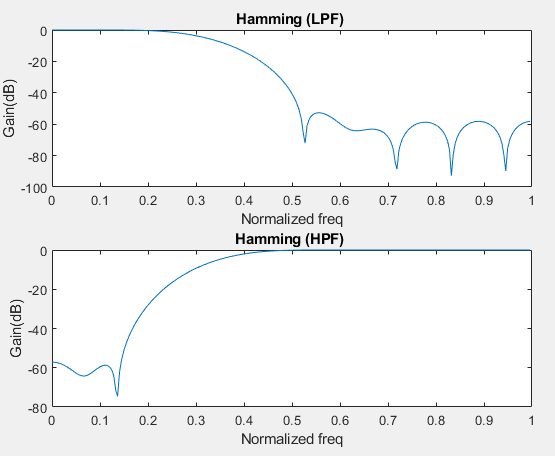
ylabel('Gain(dB)');

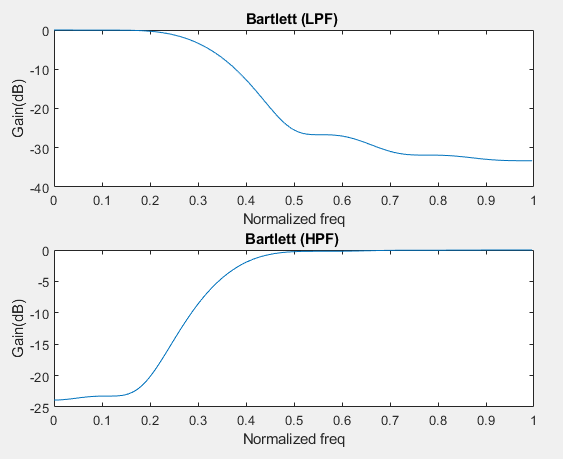
xlabel('Normalized freq');

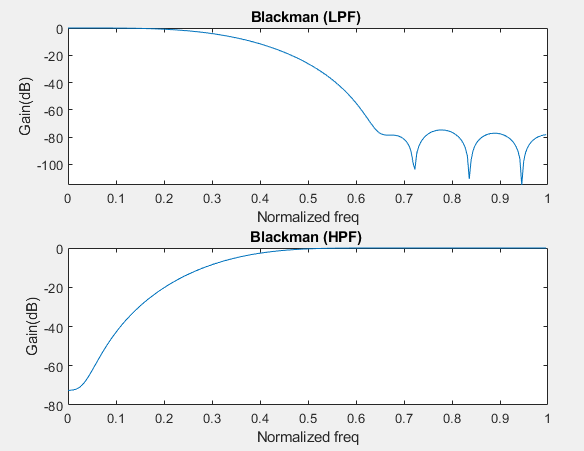
**Result:-** FIR lowpass and highpass filter were designed and it was verified by the following output.











**Conclusion:-**