1. Solve:

(a)

Use the Matlab routine **tf2sos** to obtain the coefficients for the cascade implementation of *H(z)* using second order sections. The Matlab code is shown as follows:

```
b=[0.0609 -0.3157 0.8410 -1.4378 1.7082 -1.4378 0.8410 -0.3157
0.0609];
a=[1.0 -0.8961 2.6272 -0.9796 2.1282 -0.0781 0.9172 0.0502
0.2602];
[sos,gain]=tf2sos(b,a)
```

The result is shown as follows:

```
sos =

    1.0000   -1.8810    1.0000    1.0000    0.7493    0.3790
    1.0000   -1.3129    1.0000    1.0000   -0.2219    0.7562
    1.0000   -1.0724    1.0000    1.0000   -0.6471    0.9241
    1.0000   -0.9177    1.0000    1.0000   -0.7763    0.9824


gain =

    0.0609
```

(b)

Calculate, by hand, the IIR lattice coefficients for the first second order section. Assume that the gain is distributed evenly through each second order section.

$gain1 = gain2 = gain3 = gain4 = \sqrt[4]{gain} = \sqrt[4]{0.0609} \approx 0.4968$

$H_1(z) = \frac{0.4968(1 - 1.8810z^{-1} + z^{-2})}{1 + 0.7493z^{-1} + 0.3790z^{-2}} = \frac{0.4968 - 0.9345z^{-1} + 0.4968z^{-2}}{1 + 0.7493z^{-1} + 0.3790z^{-2}}$

The K parameters can be determined as follows:

$k_2 = a(2) = 0.3790$

$k_1 = \frac{a(1)}{1 + a(2)} = \frac{0.7493}{1 + 0.3790} \approx 0.5434$

The V parameters can be determined as follows:

$V(3) = b(2) = 0.4968$

$V(2) = b(1) - (k_1 \times k_2 + k_1) \times V_3 = -0.1345 - (0.5434 \times 0.379 + 0.5434) \times 0.4968 = -1.3068$

$V(1) = b(0) - k_1 V_2 - k_2 V_3 = 0.4968 - 0.5434 \times (-1.3068) - 0.379 \times 0.4968 = 1.0186$

(c)

Use Matlab to determine the IIR lattice coefficients for each of the remaining second order sections. Provide your MATLAB code with your solution. Develop block

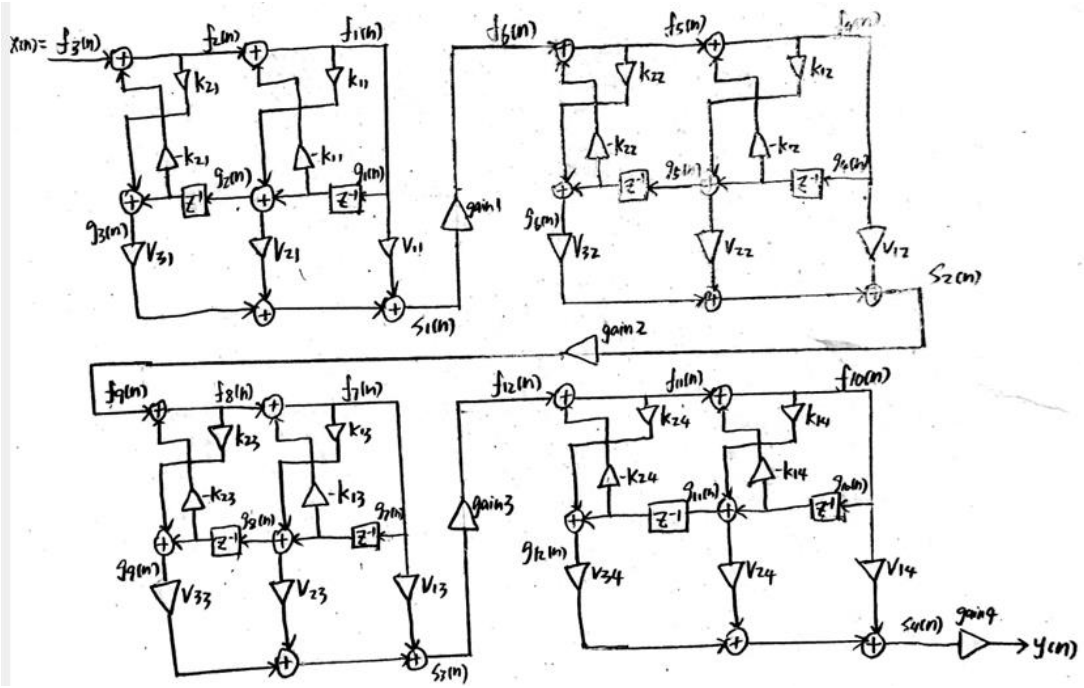
diagrams for the IIR lattice implementation of all second order sections. The Matlab code is shown as follows:

```
sos_change=gain^(1/4)*sos(1:4,1:3); % Distribute the gain
                                    evenly
sos(1:4,1:3)=sos_change;
% The 1st second section
K21=sos(1,6);
K11=sos(1,5)/(1+sos(1,6));
V31=sos(1,3);
V21=sos(1,2)-(K11*K21+K11)*V31;
V11=sos(1,1)-K11*V21-K21*V31;
% The 2nd second section
K22=sos(2,6);
K12=sos(2,5)/(1+sos(2,6));
V32=sos(2,3);
V22=sos(2,2)-(K12*K22+K12)*V32;
V12=sos(2,1)-K12*V22-K22*V32;
% The 3rd second section
K23=sos(3,6);
K13=sos(3,5)/(1+sos(3,6));
V33=sos(3,3);
V23=sos(3,2)-(K13*K23+K13)*V33;
V13=sos(3,1)-K13*V23-K23*V33;
% The 4th second section
K24=sos(4,6);
K14=sos(4,5)/(1+sos(4,6));
V34=sos(4,3);
V24=sos(4,2)-(K14*K24+K14)*V34;
V14=sos(4,1)-K14*V24-K24*V34;
K_parameters=[K21,K11;K22,K12;K23,K13;K24,K14]
V_parameters=[V31,V21,V11;V32,V22,V12;V33,V23,V13;V34,V24,V14]
```

The result is shown as follows:

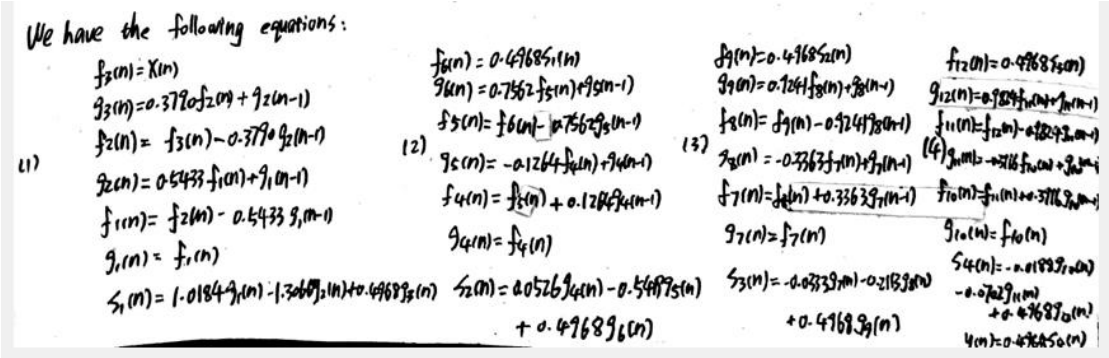
```
K_parameters =

    0.3790    0.5433
    0.7562   -0.1264
    0.9241   -0.3363
    0.9824   -0.3916


V_parameters =

    0.4968   -1.3066    1.0184
    0.4968   -0.5419    0.0526
    0.4968   -0.2113   -0.0333
    0.4968   -0.0702   -0.0188
```

The block diagram is shown as follows:



(d)

Develop appropriate difference equations for the implementation of the overall filter using the second order IIR lattice sections.



2. Solve:

(a)

Write a Matlab function to implement a second order section IIR lattice filter. The Matlab code is shown as follows:

```
function ba=implement_sos_IIR(K,V)
ba=ones(1,6); % The coefficients [b0,b1,b2,a0,a1,a2]
ba(6)=K(2);
ba(5)=K(1)*(1+ba(6));
ba(3)=V(3);
ba(2)=V(2)+(K(1)*K(2)+K(1))*ba(3);
ba(1)=V(1)+V(2)*K(1)+V(3)*K(2);
return
```

(b)

The Matlab routine **sampdata** has been provided in the course website. Use the sequence generated by this routine as input for the filter simulations in this problem.The figure of the input sequence x(n) is shown in Fig. 1:

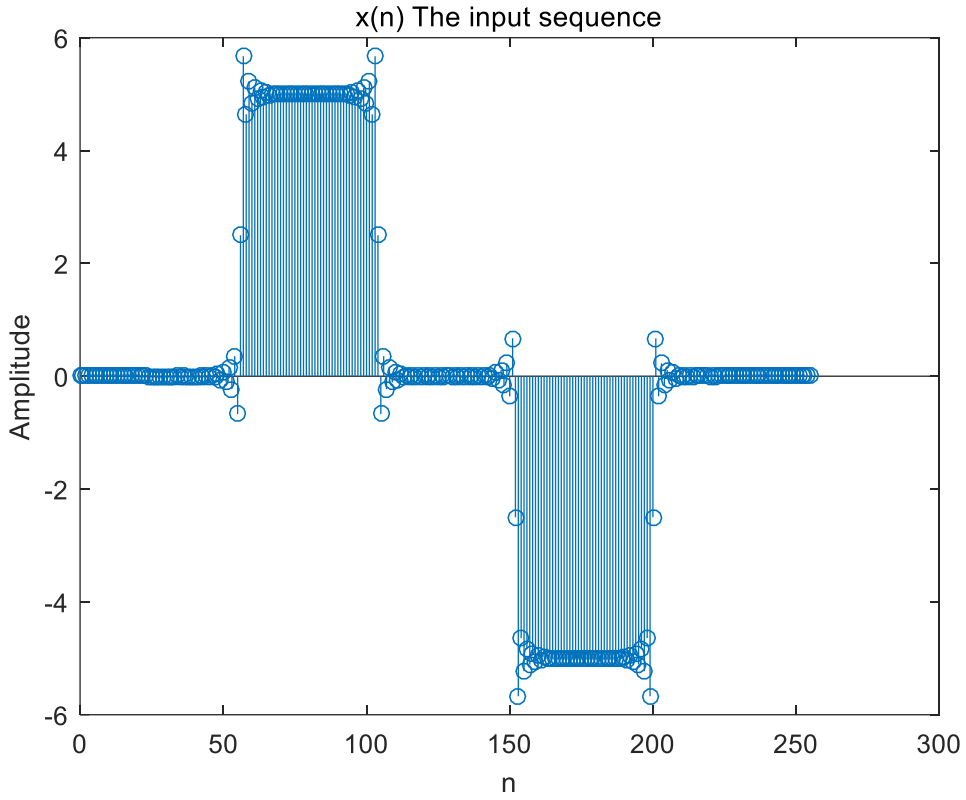


Fig. 1. The input sequence x(n)

(c)

Write a Matlab routine to simulate the second order section implementation of *H(z)* using second order IIR lattice sections and your function from part a. Since the order of the filter is 6, your implementation should have 3 second order sections and you should call your routine 3 times to implement the cascaded filter.

```
x_n=sampdata;
sos=zeros(3,6);
ba1=implement_sos_IIR(K1,V1);          % The 1st transfer
function
sos(1,1:6)=ba1(1:6);
ba2=implement_sos_IIR(K2,V2);          % The 2nd transfer
function
sos(2,1:6)=ba2(1:6);
ba3=implement_sos_IIR(K3,V3);          % The 3rd transfer
function
sos(3,1:6)=ba3(1:6);
[h,~]=impz(sos);
y1_n=conv(x_n,h');
figure(2)
stem(0:length(y1_n)-1,y1_n)
title('y1(n) The output sequence')
xlabel('n')
ylabel('Amplitude')
```

The figure of the output sequence y1(n) is shown in Fig. 2:

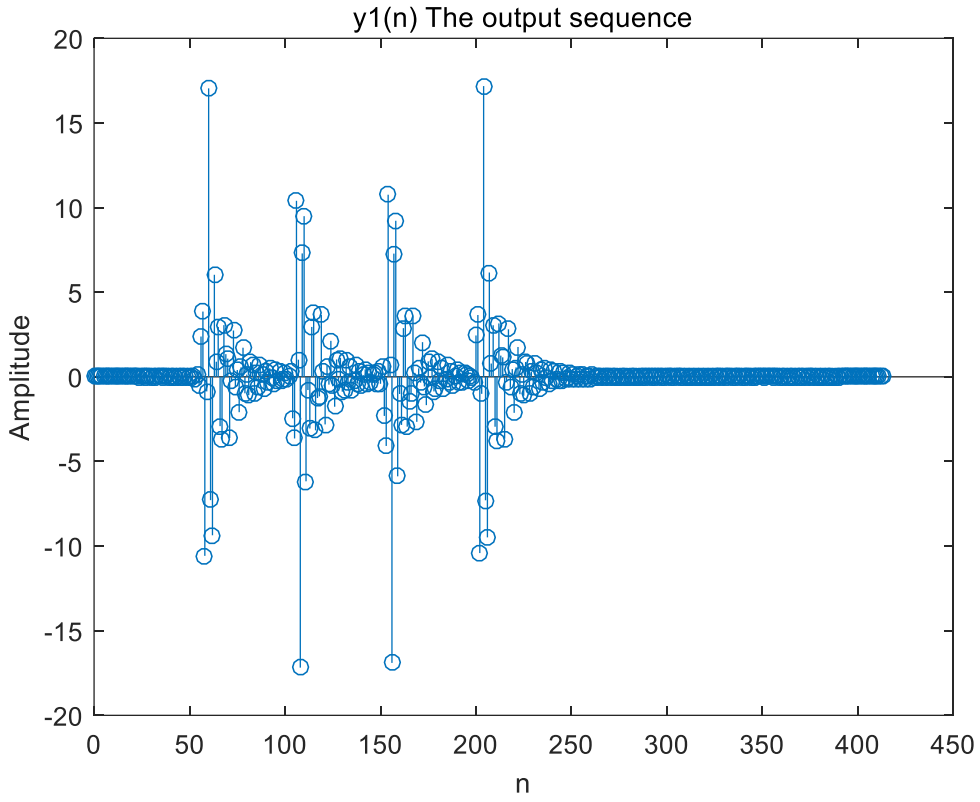


Fig. 2. The output sequence y1(n)

(d)

Use the Matlab routine **latc2tf** to obtain the transfer function equivalent for each of the second order sections. Then write a Matlab routine to call the Matlab routine filter three times in cascade to filter the input sequence. The Matlab code is shown as follows:

```
x_n=sampdata;
[NUM1,DEN1]=latc2tf(K1,V1);            % The 1st transfer
                                         function
s21_n=filter(NUM1,DEN1,x_n);
[NUM2,DEN2]=latc2tf(K2,V2);            % The 2nd transfer
                                         function
s22_n=filter(NUM2,DEN2,s21_n);
[NUM3,DEN3]=latc2tf(K3,V3);            % The 3rd transfer
                                         function
y2_n=filter(NUM3,DEN3,s22_n);
```

The figure of the output sequence y2(n) is shown in Fig. 3:

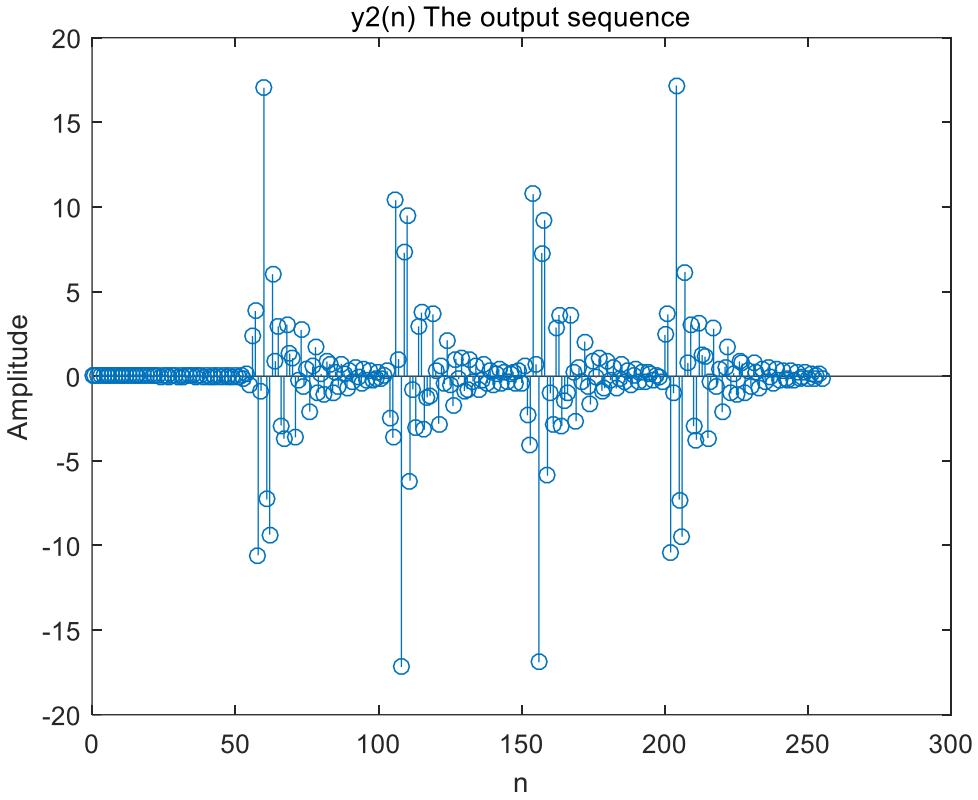


Fig.3. The output sequence y2(n)

(e)

Compare the output from your routine that uses your second order lattice section filter with the output from your routine that uses the Matlab filter routine and note any differences. Compute the absolute error between the output sequences y1(n) and y2(n). The Matlab code is shown as follows:

```
y2_n(length(y2_n)+1:length(y1_n))=zeros(1,length(y1_n)-
length(y2_n));
difference=abs(y1_n-y2_n);
figure(4)
stem(0:length(difference)-1,difference)
title('The difference between two y1(n) and y2(n)')
xlabel('n')
ylabel('Amplitude')
```

The figure of the absolute error between y1(n) and y2(n) is shown in Fig. 4:

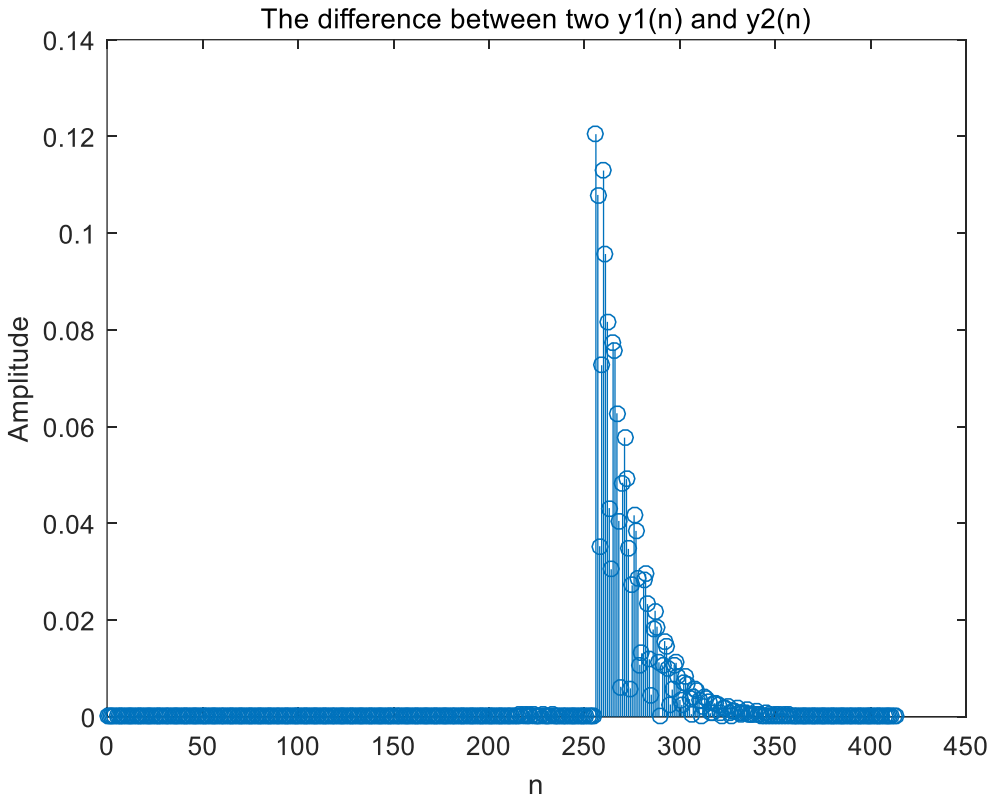


Fig. 4. Absolute error between y1(n) and y2(n)

(f)

Turn in your Matlab code, the output plots, and a summary of your observations.

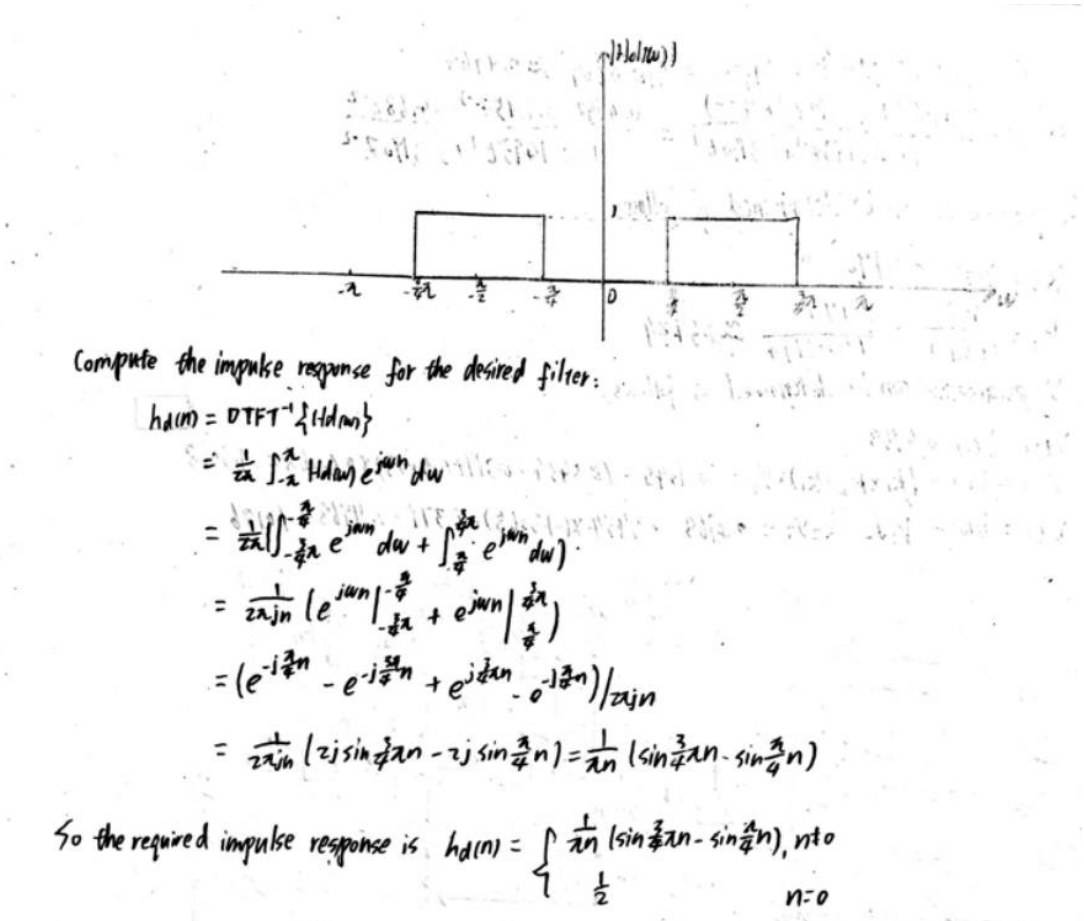
Summary: As is clearly demonstrated in Fig. 4., the absolute error is definitely small so that people can actually ignore it while conducting research. It's safe to draw a conclusion that the result of my routine is almost the same as the Matlab **latc2tf** and **filter** routines.

3. Solve:

(a)

Determine the coefficients for linear phase filter with 61 filter coefficients based on the use of the window method with a Blackman window. You are required to

i. Develop an equation for the required samples of the impulse response for the desired linear phase filter



Compute the impulse response for the desired filter:

$$h_d(n) = DTFT^{-1}\{H_d(jw)\}$$

$$= \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(jw)\, e^{jwn}\, dw$$

$$= \frac{1}{2\pi}\left(\int_{-\frac{3}{4}\pi}^{-\frac{\pi}{4}} e^{jwn}\, dw + \int_{\frac{\pi}{4}}^{\frac{3}{4}\pi} e^{jwn}\, dw\right)$$

$$= \frac{1}{2\pi jn}\left(e^{jwn}\Big|_{-\frac{3}{4}\pi}^{-\frac{\pi}{4}} + e^{jwn}\Big|_{\frac{\pi}{4}}^{\frac{3}{4}\pi}\right)$$

$$= \left(e^{-j\frac{\pi}{4}n} - e^{-j\frac{3\pi}{4}n} + e^{j\frac{3\pi}{4}n} - e^{-j\frac{\pi}{4}n}\right)\Big/ 2\pi jn$$

$$= \frac{1}{2\pi jn}\left(2j\sin\frac{3}{4}\pi n - 2j\sin\frac{\pi}{4}n\right) = \frac{1}{\pi n}\left(\sin\frac{3}{4}\pi n - \sin\frac{\pi}{4}n\right)$$

So the required impulse response is $h_d(n) = \begin{cases} \frac{1}{\pi n}\left(\sin\frac{3}{4}\pi n - \sin\frac{\pi}{4}n\right), & n \neq 0 \\ \frac{1}{2}, & n = 0 \end{cases}$

ii. Write a Matlab script to

A. Compute the required 61 samples for the desired impulse response.

B. Compute the required 61 samples for the Blackman window.

C. Multiply the two sequences to obtain the filter coefficients for the filter.

D. determine and plot the magnitude response and the phase response for the filter you designed.

The Matlab code is shown as follows:

```
w1=pi/4;
w2=3*pi/4;
M=(61-1)/2;
n1=-M:-1;
w_neg=0.42+0.5*cos(2*pi*(n1)/(2*M))+0.08*cos(4*pi*(n1)/(2*M));
b2_neg=(1/pi)*(sin(w2*(n1))./(n1)-sin(w1*(n1))./(n1));
n2=1:M-1;
w_pos=0.42+0.5*cos(2*pi*(n2)/(2*M))+0.08*cos(4*pi*(n2)/(2*M));
b2_pos=(1/pi)*(sin(w2*(n2))./(n2)-sin(w1*(n2))./(n2));
b2_31=1/2;
b2=[b2_neg b2_31 b2_pos];     % The required 61 samples for the
```

```
                                desired impulse response
w_Blackman=[w_neg 1 w_pos];   % The required 61 samples for the
                                Blackman window
b2=b2.*w_Blackman;            % Multiply the two sequences
% Plot the magnitude and phase response of the resulting
filter
w=pi*(0:0.005:1.0);
h=freqz(b2,1,w);
hmag=abs(h);
hphase=angle(h);
tol=0.95*pi;
figure(5)
subplot(2,1,1)                % Plot the magnitude response of
                                the resulting filter
hmag=unwrap(hmag,tol);
plot(w,hmag)
title('Frequency Response Plot')
xlabel('Frequency (radians)')
ylabel('Magnitude')
subplot(2,1,2)                % Plot the phase response of the
                                resulting filter
hphase=unwrap(hphase,tol);
plot(w,hphase)
title('Phase Response Plot')
xlabel('Frequency (radians)')
ylabel('Magnitude (decibels)')
```

Therefore, the magnitude response and the phase response for the filter I designed is shown in Fig. 5.

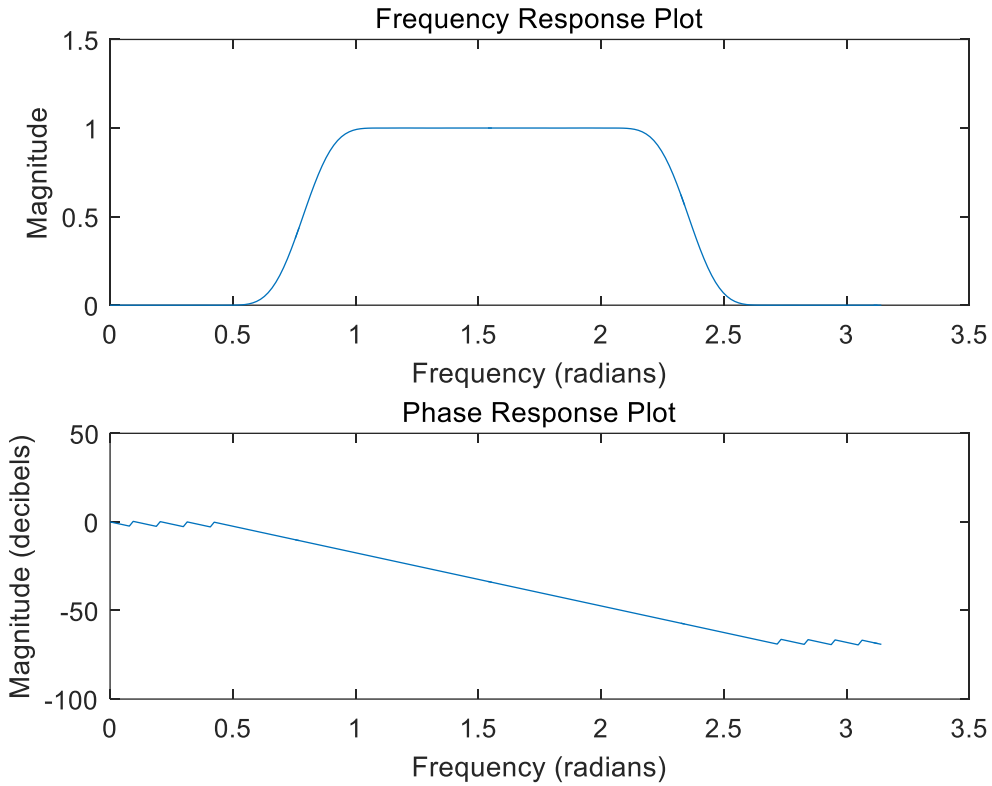


Fig. 5. The magnitude response and the phase response for the filter I designed

iii. Write a Matlab script using the Matlab fir1 function to design a digital filter to meet the specifications given above using a Blackman window. The Matlab blackman function can be used to obtain the Blackman window using Matlab.

The Matlab code is shown as follows:

```
order=60;
wn=[1/4 3/4];
w=window(@blackman,order+1);
b1=fir1(order,wn,'bandpass',w);
```

iv. Determine and plot the magnitude response and the phase response for the filter you designed using the Matlab fir1 function.

The Matlab code is shown as follows:

```
% Plot the magnitude and phase response of the resulting
filter
w=pi*(0:0.005:1.0);
h=freqz(b1,1,w);
hmag=abs(h);
hphase=angle(h);
figure(6)                          % Plot the magnitude response of
                                    the resulting filter
subplot(2,1,1)
hmag=unwrap(hmag,tol);
```

```
plot(w,hmag)
title('Frequency Response Plot')
xlabel('Frequency (radians)')
ylabel('Magnitude')
subplot(2,1,2)                % Plot the phase response of the
                               resulting filter
hphase=unwrap(hphase,tol);
plot(w,hphase)
title('Phase Response Plot')
xlabel('Frequency (radians)')
ylabel('Magnitude (decibels)')
```

Therefore, the magnitude response and the phase response for the filter I designed using the Matlab **fir1** function is shown in Fig. 6.

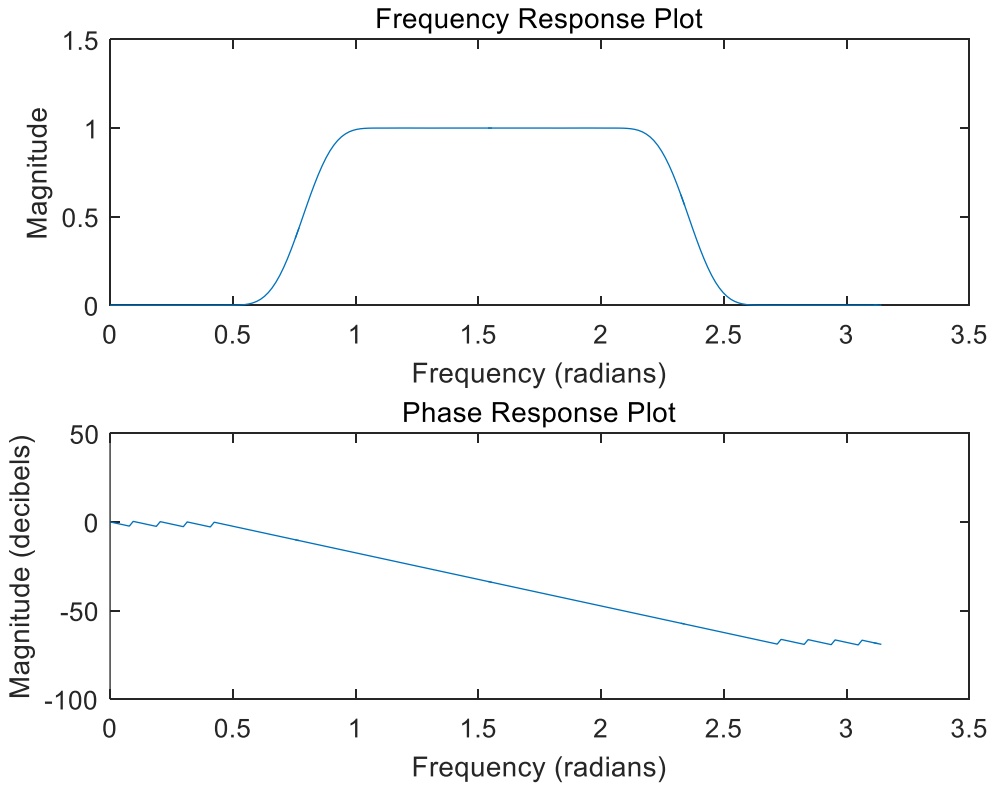


Fig. 6. The magnitude response and the phase response for the filter using fir1 function

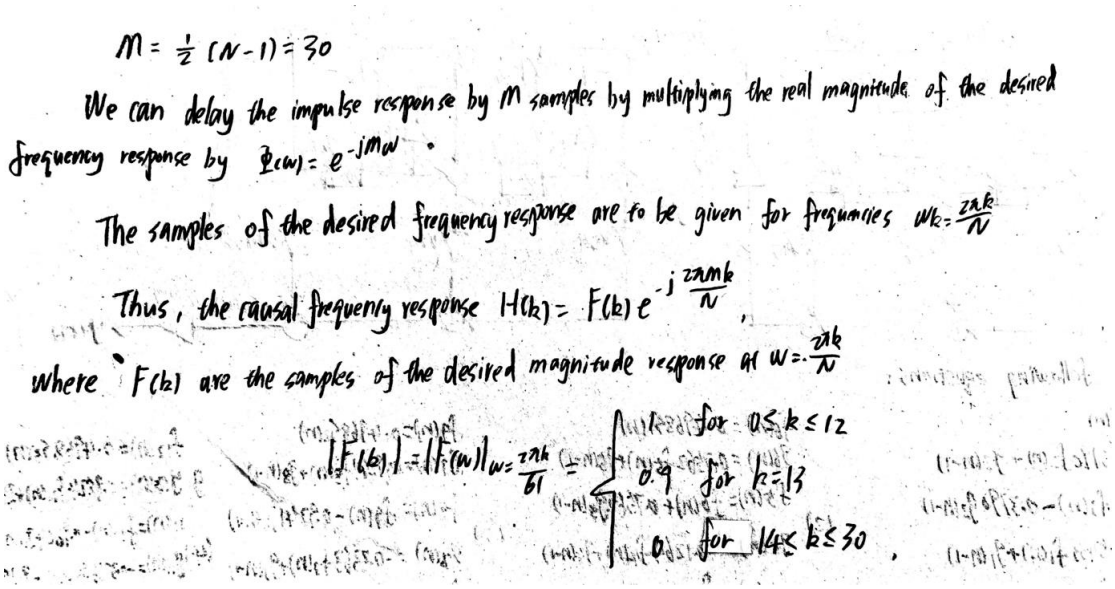
v. Compare your coefficients to the coefficients obtained with the Matlab fir1 function.

Comparing the Fig. 5 and Fig. 6, the two figures are almost the same so that people can actually ignore it while conducting research. It's safe to draw a conclusion that the result of my designed filter by using traditional computation is almost the same as the result by using the Matlab **fir1** function.

4. Solve:

(a)

Write an equation to determine a causal frequency response $H(k)$ $\forall\ 0 \le k \le 60$.



(b)

Write a Matlab script to determine the 61 filter coefficients for the filter. The Matlab code is shown as follows:

```
hd=zeros(1,31);
hd(1:13)=ones(1,13);
hd(14)=0.9;
M=length(hd)-1;
N=2*M+1;
t=0:M;
s=exp(-1i*2*pi*M*t/N);
hw=s.*hd;
hw(M+2:N)=zeros(1,M);
for k=32:1:61                          % Use the relationship
                                       H(-k)=conjugate(H(k))
   hw(k)=conj(hw(N+1+1-k));
end
```

(c)

Plot the magnitude response and the phase response for your filter. The Matlab code is shown as follows:

```
b=real(ifft(hw));
a=1;
[h,w]=freqz(b,a,200);
hmag=abs(h);
hang=angle(h);
```

```
tol=0.99*pi;
figure(7)                          % Plot the magnitude response
                                    of the resulting filter
subplot(2,1,1)
hmag=unwrap(hmag,tol);
plot(w,hmag)
title('Magnitude response plot for frequency sampling')
xlabel('Frequency (radians)')
ylabel('Magnitude')
subplot(2,1,2)                    % Plot the phase response of the
                                    resulting filter
hang=unwrap(hang,tol);
plot(w,hang)
title('Phase response plot for frequency sampling')
xlabel('Frequency (radians)')
ylabel('Magnitude (decibels)')
```

Therefore, the magnitude response and the phase response for the filter I designed is shown in Fig. 7.

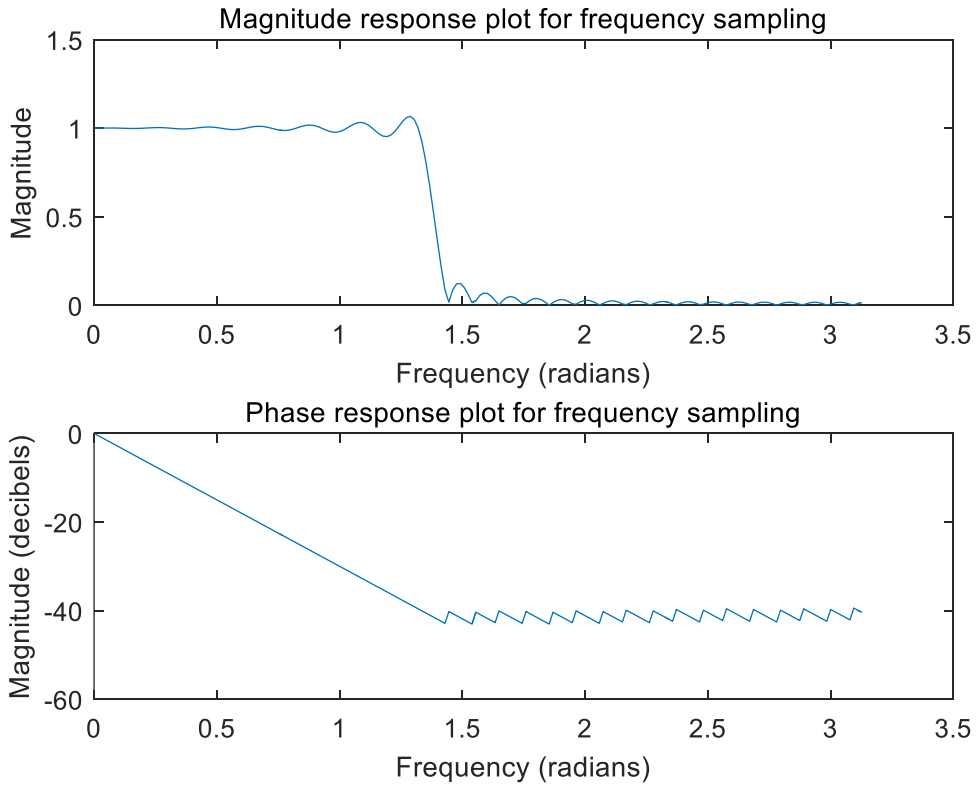


Fig. 7. The magnitude response and the phase response for the filter I designed

(d)

Determine if the magnitude response of your filter matches the desired magnitude response at the specified frequencies.

It's obvious that this is a low-pass filter, which matches the required specifications.