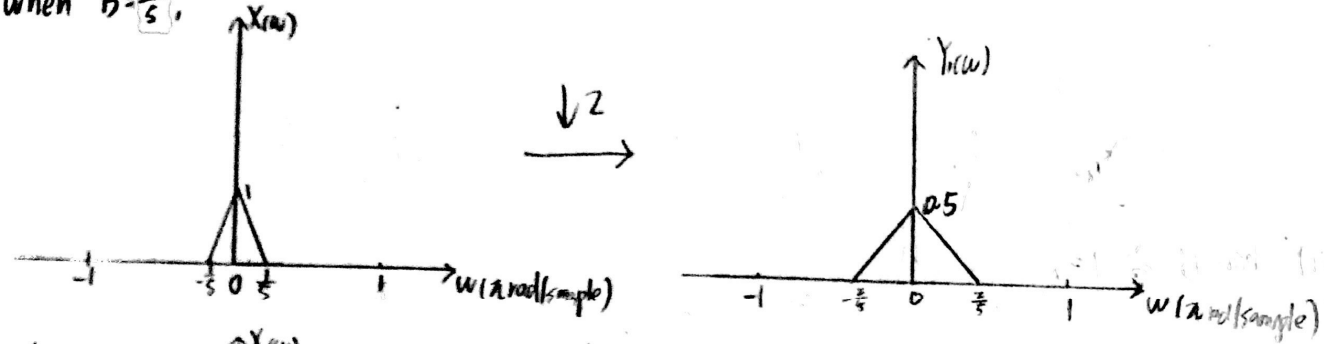
ECE513 Hmwk9

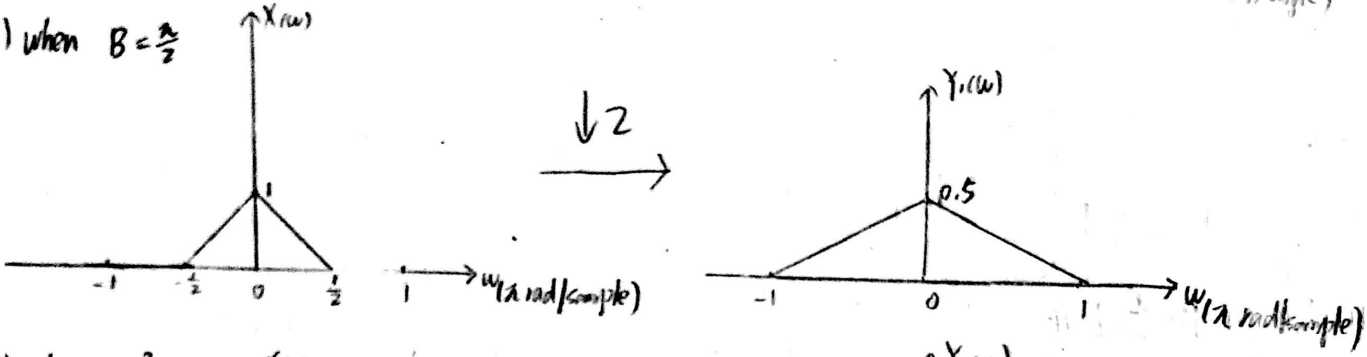
1. Solve:

(a) $m=2$,

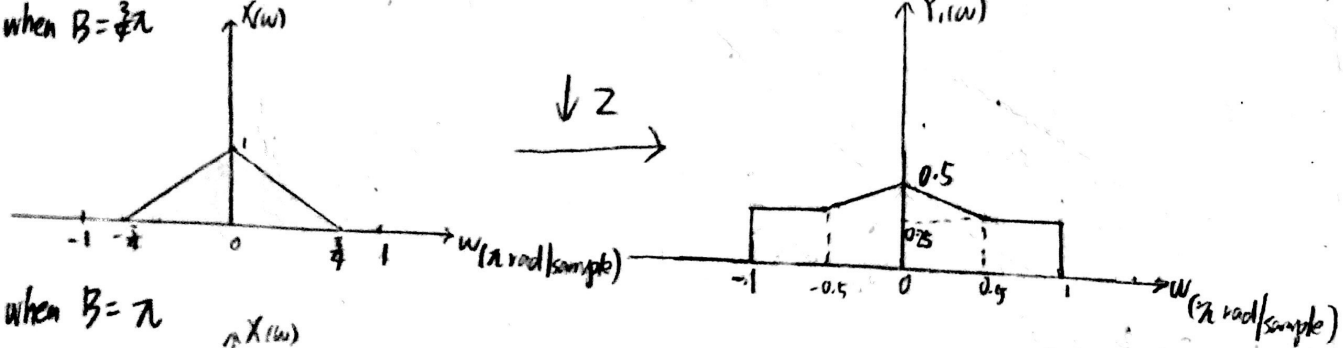
(i) when $B=\frac{\pi}{5}$,



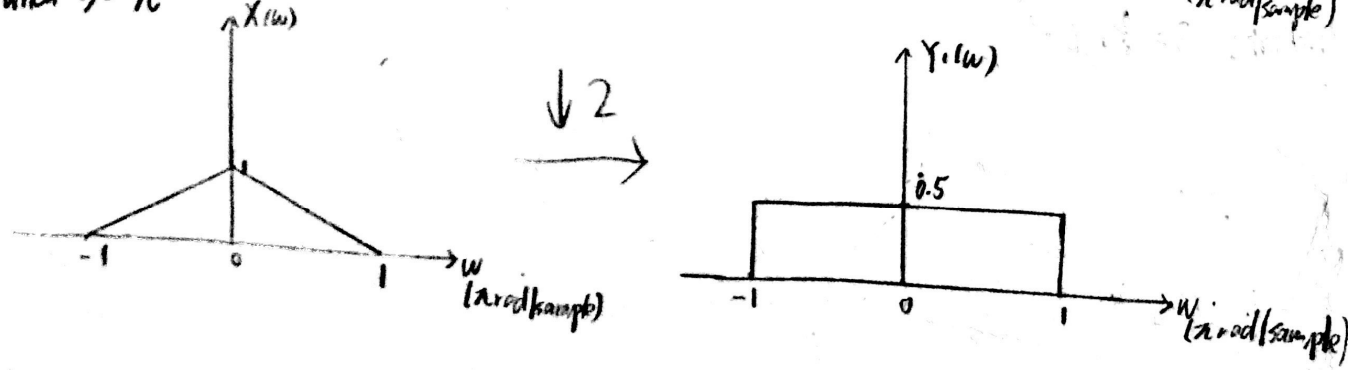
(ii) when $B=\frac{\pi}{2}$



(iii) when $B=\frac{3}{4}\pi$



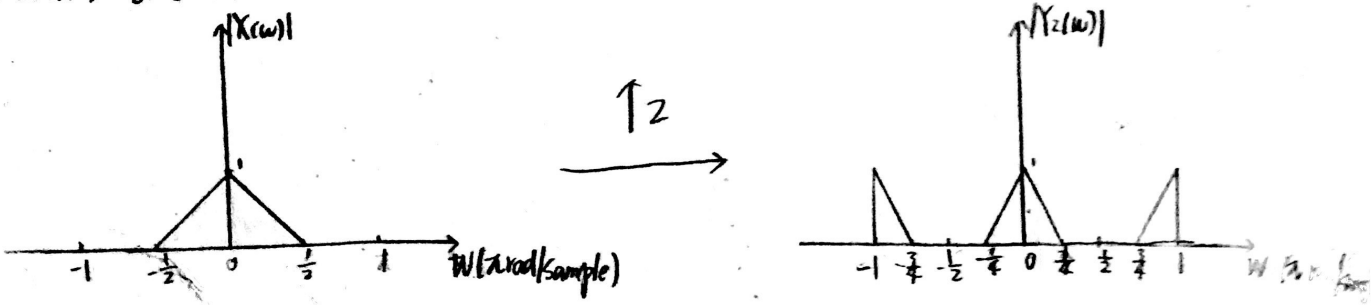
(iv) when $B=\pi$



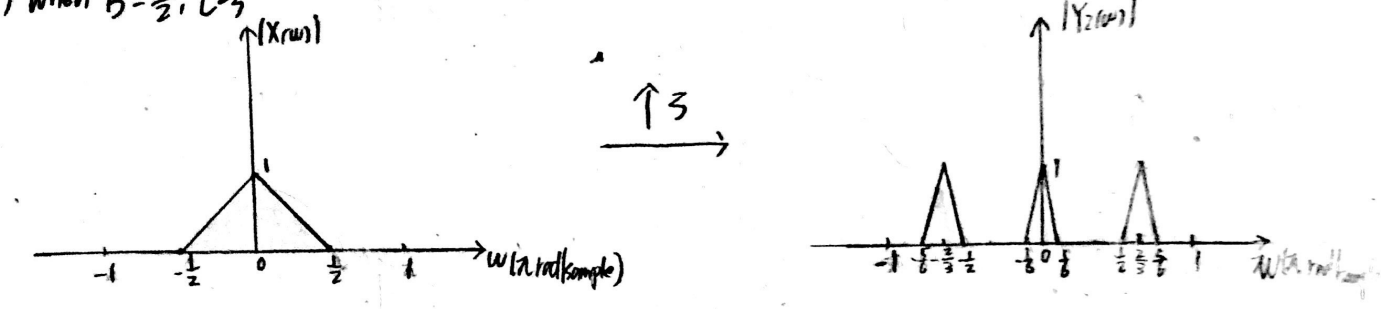
Scanned with CamScanner

(b)

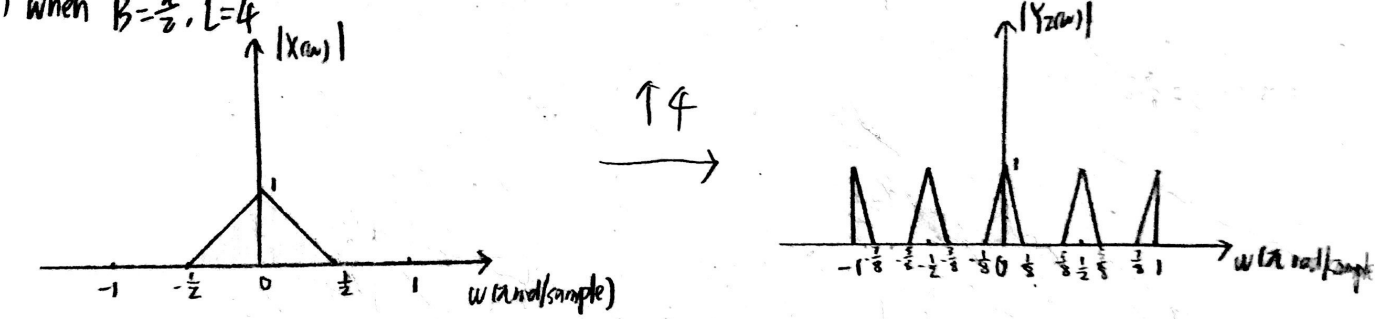
(i) when $B=\frac{\pi}{2}$, $L=2$.



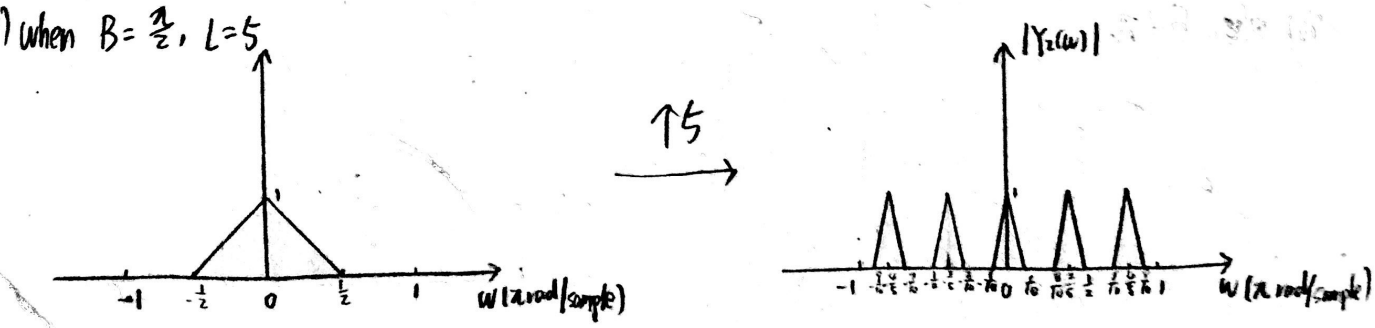
(ii) when $B=\frac{\pi}{2}$, $L=3$



(iii) when $B=\frac{\pi}{2}$, $L=4$



(iv) when $B=\frac{\pi}{2}$, $L=5$



Scanned with CamScanner

(c)

a.(i) When B=pi/5, M=2, the Matlab code is shown as follows:

```
f1=[0 0.05 0.1 0.15 0.2 0.4 0.6 0.8 1];
m1=[1 0.75 0.5 0.25 0.0 0.0 0.0 0.0 0];
% part(a)
% a.(i)
b1=fir2(100,f1,m1);   % Produce a 101 sample sequences x[n] who
has required magnitude response
w=pi*(0:0.005:1);
h1=freqz(b1,1,w);
hmag1=abs(h1);
figure(1)
plot(w/pi,hmag1);
title('Magnitude Spectrum Plot before down sampling')
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Magnitude')
num1=length(b1);
b1_new=b1(1,1:2:num1);
h1_new=freqz(b1_new,1,w);
hmag1_new=abs(h1_new);
figure(2)
plot(w/pi,hmag1_new);
title('Magnitude Spectrum Plot after down sampling')
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Magnitude')
```

The resulting figures are shown in Fig. 1 and Fig. 2 respectively:

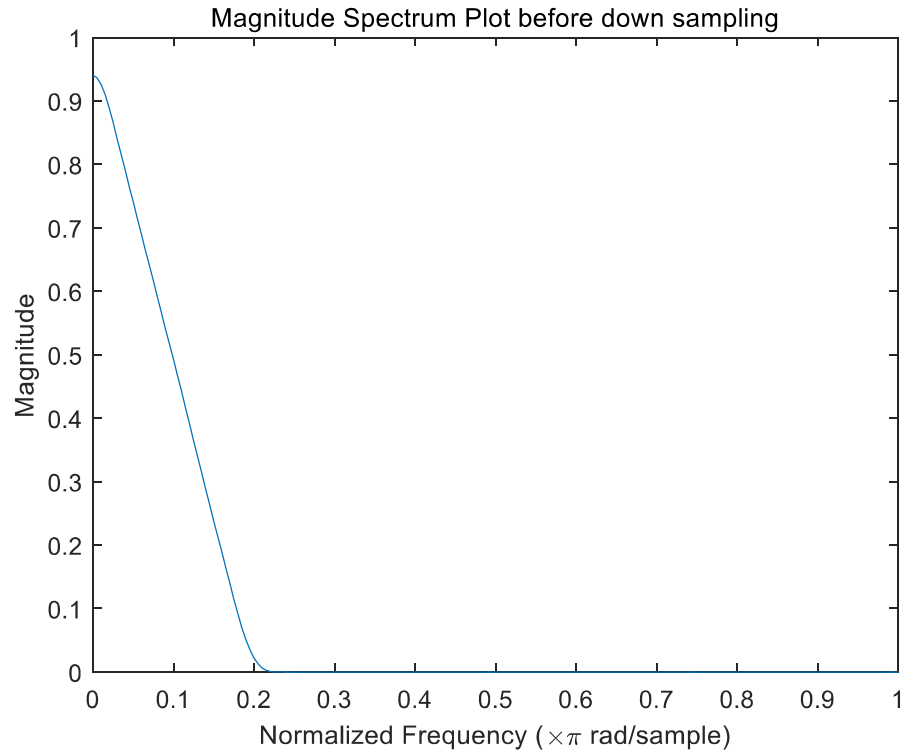


Fig. 1. The magnitude spectrum before down sampling when B=pi/5

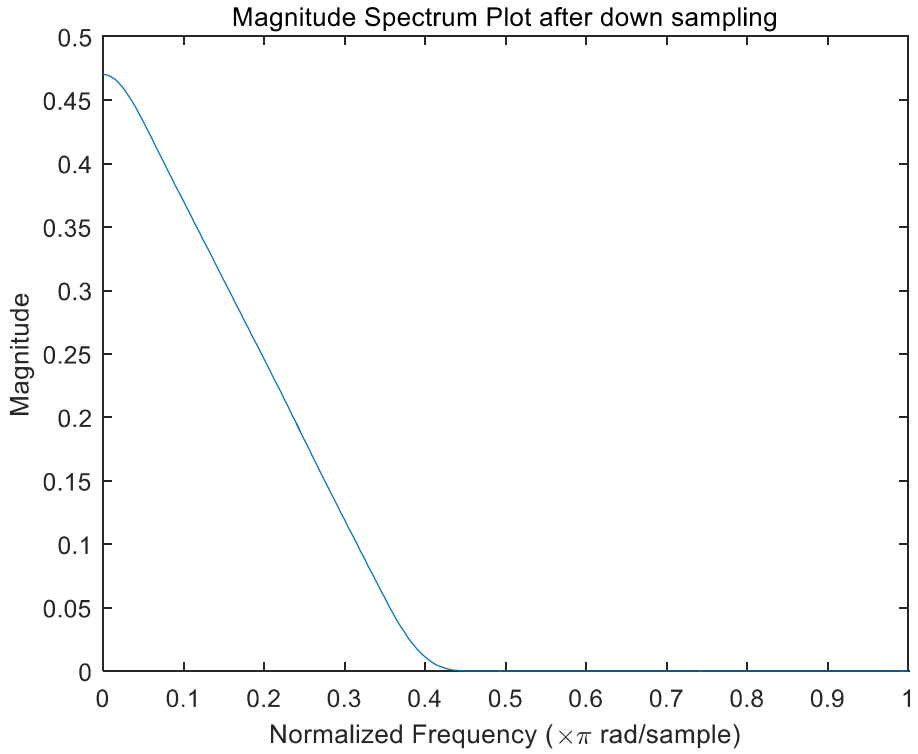


Fig. 2. The magnitude spectrum after down sampling when B=pi/5 and M=2

a.(ii) When B=pi/2, M=2, the Matlab code is shown as follows:

```
f2=[0 0.2 0.4 0.5 0.6 0.8 1];
m2=[1 0.6 0.2 0.0 0.0 0.0 0];
b2=fir2(100,f2,m2);   % Produce a 101 sample sequences x[n] who
has required magnitude response
w=pi*(0:0.005:1);
h2=freqz(b2,1,w);
hmag2=abs(h2);
figure(3)
plot(w/pi,hmag2);
title('Magnitude Spectrum Plot before down sampling')
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Magnitude')
num2=length(b2);
b2_new=b2(1,1:2:num2);
h2_new=freqz(b2_new,1,w);
hmag2_new=abs(h2_new);
figure(4)
plot(w/pi,hmag2_new);
title('Magnitude Spectrum Plot after down sampling')
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Magnitude')
```

The resulting figures are shown in Fig. 3 and Fig. 4 respectively:

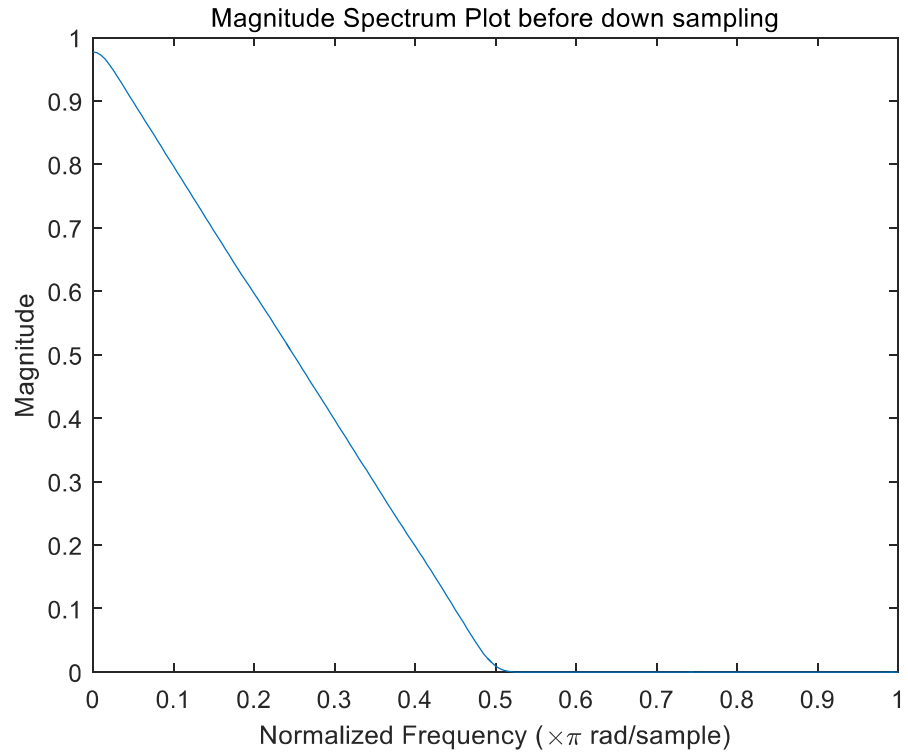


Fig. 3. The magnitude spectrum before down sampling when B=pi/2

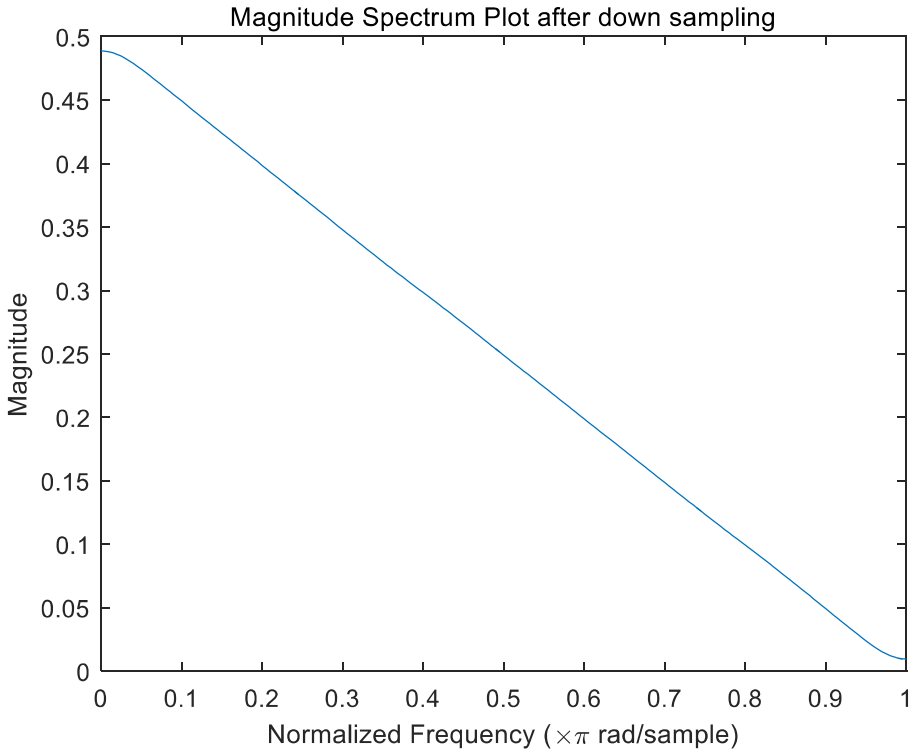


Fig. 4. The magnitude spectrum after down sampling when B=pi/2 and M=2

a.(iii) When B=3pi/4, M=2, the Matlab code is shown as follows:

```
f3=[0 0.25 0.5 0.75 1];
m3=[1 2/3  1/3 0.0  0];
b3=fir2(100,f3,m3);    % Produce a 101 sample sequences x[n]
who has required magnitude response
w=pi*(0:0.005:1);
h3=freqz(b3,1,w);
hmag3=abs(h3);
figure(5)
plot(w/pi,hmag3);
title('Magnitude Spectrum Plot before down sampling')
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Magnitude')
num3=length(b3);
b3_new=b3(1,1:2:num3);
h3_new=freqz(b3_new,1,w);
hmag3_new=abs(h3_new);
figure(6)
plot(w/pi,hmag3_new);
ylim([0 0.5])
title('Magnitude Spectrum Plot after down sampling')
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Magnitude')
```

The resulting figures are shown in Fig. 5 and Fig. 6 respectively:

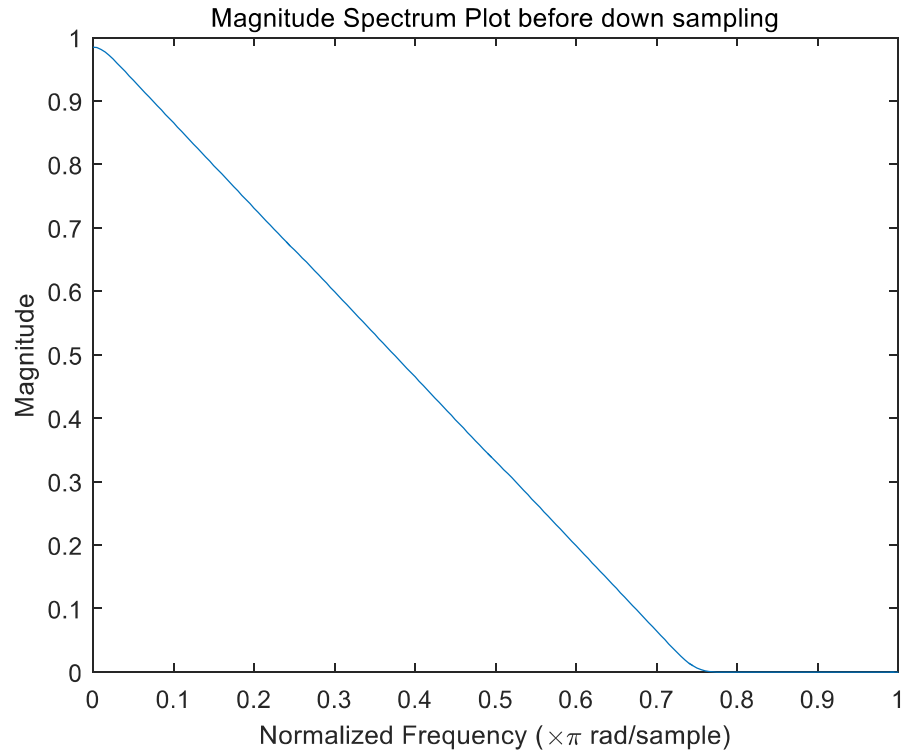


Fig. 5. The magnitude spectrum before down sampling when B=3pi/4

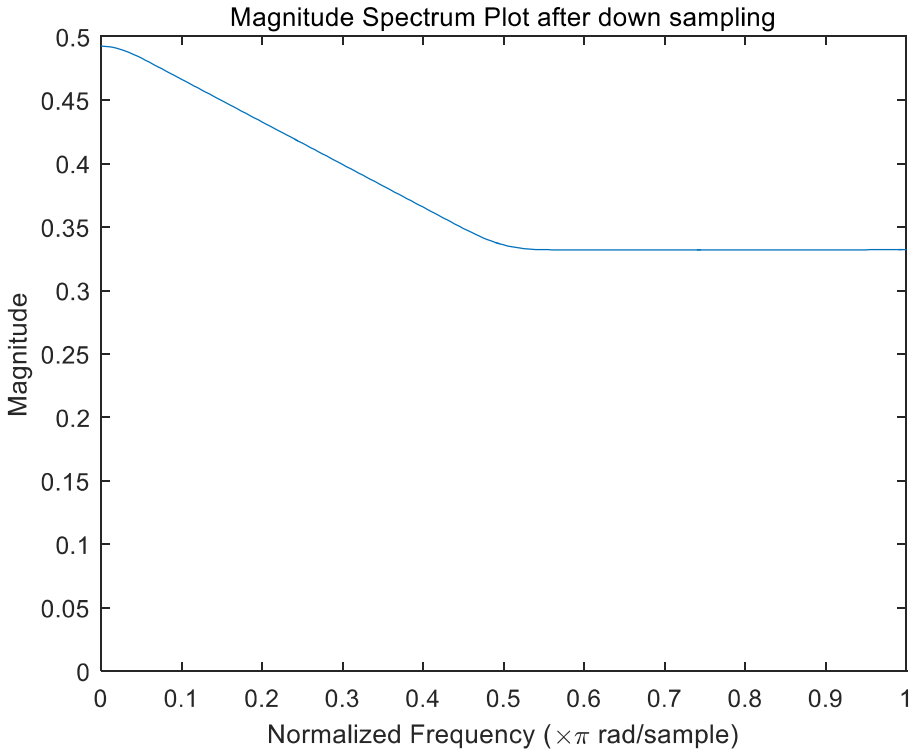


Fig. 6. The magnitude spectrum after down sampling when B=3pi/4 and M=2

a.(iv) When B=pi, M=2, the Matlab code is shown as follows:

```
f4=[0 0.2 0.4 0.6 0.8 1];
m4=[1 0.8 0.6 0.4 0.2 0];
b4=fir2(100,f4,m4);    % Produce a 101 sample sequences x[n]
who has required magnitude response
w=pi*(0:0.005:1);
h4=freqz(b4,1,w);
hmag4=abs(h4);
figure(7)
plot(w/pi,hmag4);
title('Magnitude Spectrum Plot before down sampling')
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Magnitude')
num4=length(b4);
b4_new=b4(1,1:2:num4);
h4_new=freqz(b4_new,1,w);
hmag4_new=abs(h4_new);
figure(8)
plot(w/pi,hmag4_new);
ylim([0 0.6])
title('Magnitude Spectrum Plot after down sampling')
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Magnitude')
```

The resulting figures are shown in Fig. 7 and Fig. 8 respectively:

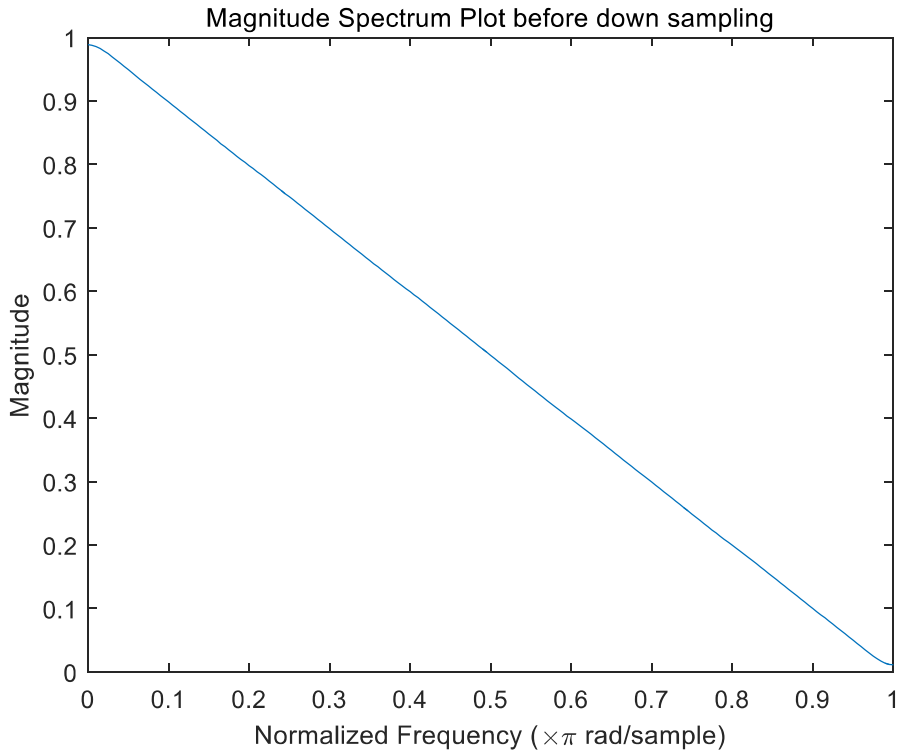


Fig. 7. The magnitude spectrum before down sampling when B=pi

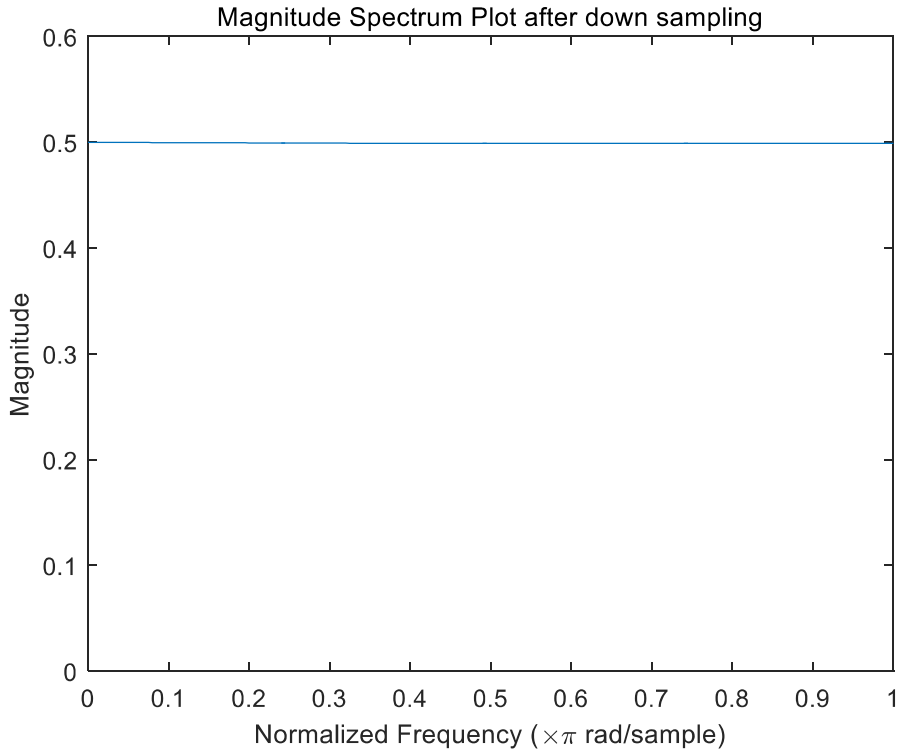


Fig. 8. The magnitude spectrum after down sampling when B=pi and M=2

b. When B=pi/2, the Matlab code is shown as follows:

```
f_b=[0 0.2 0.4 0.5 0.6 0.8 1];
m_b=[1 0.6 0.2 0.0 0.0 0.0 0];
b_b=fir2(100,f_b,m_b);
w=pi*(0:0.005:1);
h_b=freqz(b_b,1,w);
hmag_b=abs(h_b);
figure(9)
plot(w/pi,hmag_b);
title('Magnitude Spectrum Plot before up sampling')
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Magnitude')
num=length(b_b);
```

The resulting figures is shown in Fig. 9:

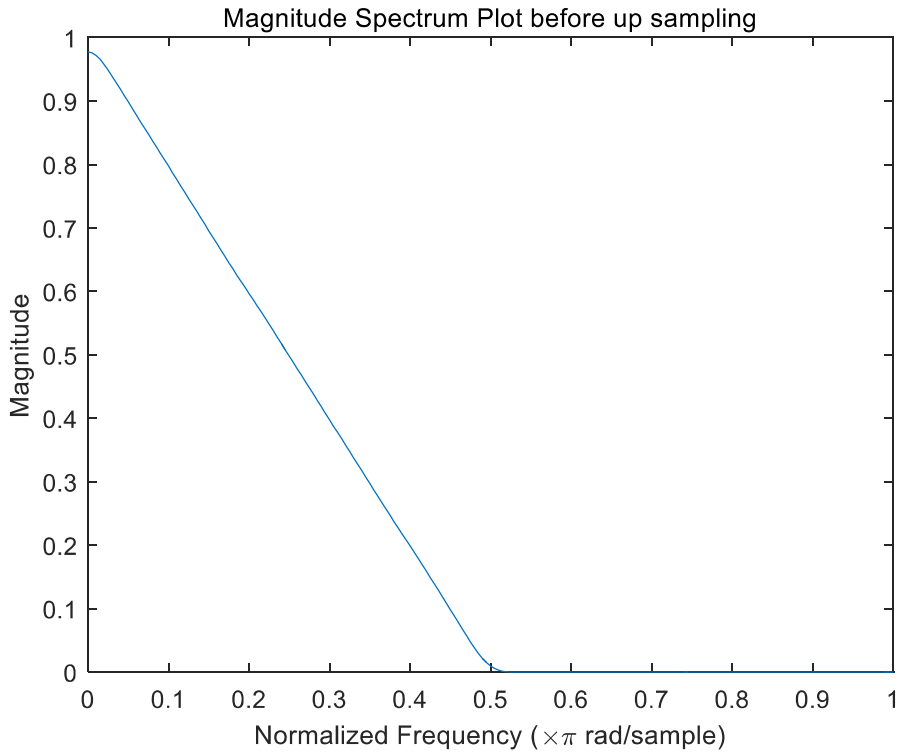


Fig. 9. The magnitude spectrum before up sampling when B=pi/2

b.(i) When B=pi/2, L=2, the Matlab code is shown as follows:

```
num0=2*num;
b_b1new=zeros(1,num0);
b_b1new(1,1:2:num0)=b_b;
h_b1new=freqz(b_b1new,1,w);
hmag_b1new=abs(h_b1new);
figure(10)
plot(w/pi,hmag_b1new);
title('Magnitude Spectrum Plot after up sampling')
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Magnitude')
```

The resulting figures is shown in Fig. 10:

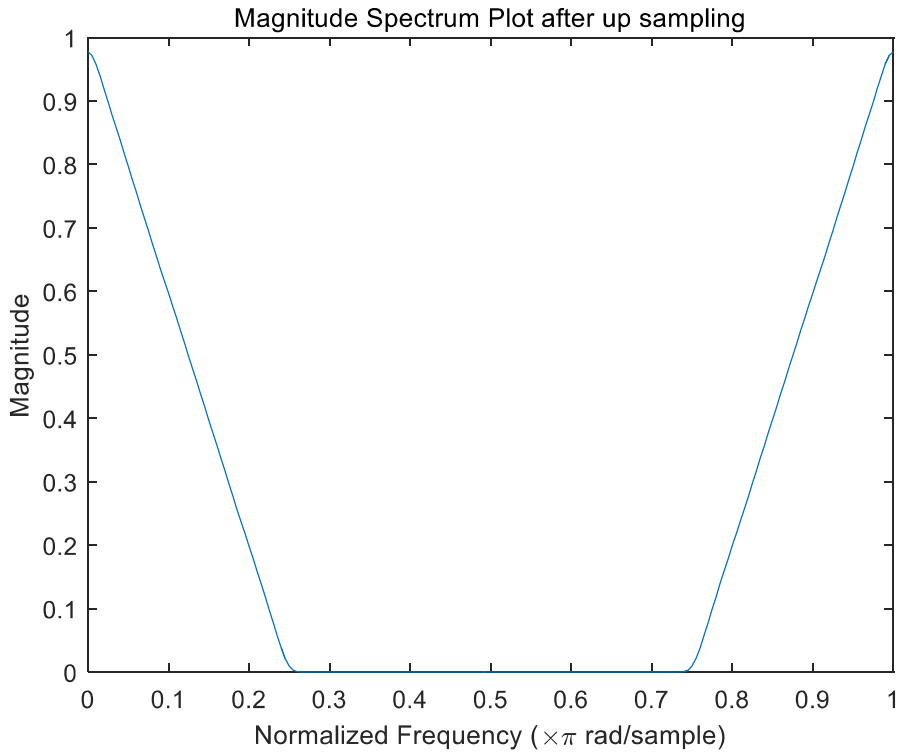


Fig. 10. The magnitude spectrum after up sampling when B=pi/2 and L=2

b.(ii) When B=pi/2, L=3, the Matlab code is shown as follows:

```
num0=3*num;
b_b2new=zeros(1,num0);
b_b2new(1,1:3:num0)=b_b;
h_b2new=freqz(b_b2new,1,w);
hmag_b2new=abs(h_b2new);
figure(11)
plot(w/pi,hmag_b2new);
title('Magnitude Spectrum Plot after up sampling')
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Magnitude')
```

The resulting figures is shown in Fig. 11:

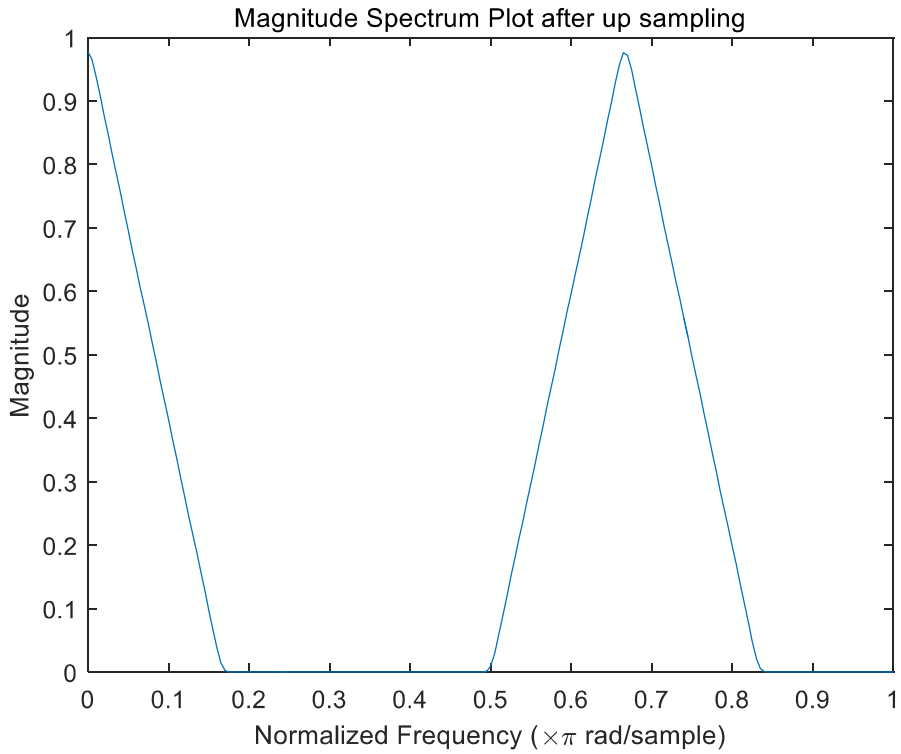


Fig. 11. The magnitude spectrum after up sampling when B=pi/2 and L=3

b.(iii) When B=pi/2, L=4, the Matlab code is shown as follows:

```
num0=4*num;
b_b3new=zeros(1,num0);
b_b3new(1,1:4:num0)=b_b;
h_b3new=freqz(b_b3new,1,w);
hmag_b3new=abs(h_b3new);
figure(12)
plot(w/pi,hmag_b3new);
title('Magnitude Spectrum Plot after up sampling')
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Magnitude')
```

The resulting figures is shown in Fig. 12:

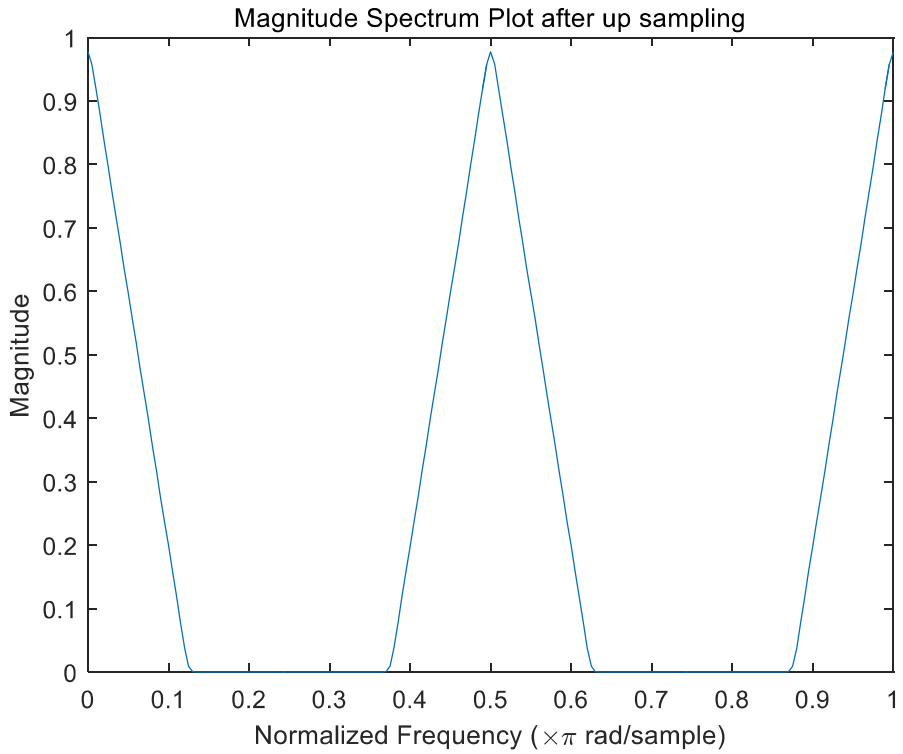


Fig. 12. The magnitude spectrum after up sampling when B=pi/2 and L=4

b.(iv) When B=pi/2, L=5, the Matlab code is shown as follows:

```
num0=5*num;
b_b4new=zeros(1,num0);
b_b4new(1,1:5:num0)=b_b;
h_b4new=freqz(b_b4new,1,w);
hmag_b4new=abs(h_b4new);
figure(13)
plot(w/pi,hmag_b4new);
title('Magnitude Spectrum Plot after up sampling')
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Magnitude')
```

The resulting figures is shown in Fig. 13:

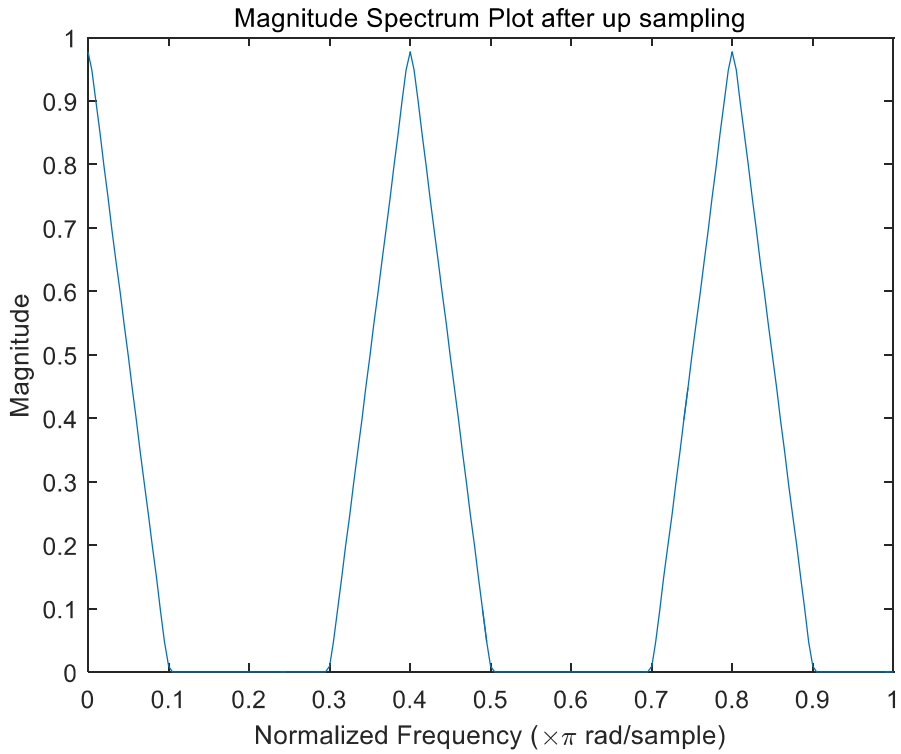


Fig. 13. The magnitude spectrum after up sampling when B=pi/2 and L=5

From the figures I have already shown, when ignoring some round-off errors, it's not hard for us to see that the drawings match the results Matlab has given.

2. Solve:

(a) Due to DTFT: $X(e^{jw}) = \sum_{n=-\infty}^{\infty} x(n) e^{-jwn}$

let $x_{int}(n) = \begin{cases} x(n), & n = 0, \pm m, \pm 2m, \cdots, \\ 0, & \text{otherwise} \end{cases}$

$$X_d(e^{jw}) = \sum_{n=-\infty}^{\infty} x_d(n) e^{-jwn} = \sum_{n=-\infty}^{\infty} x(nm) e^{-jwn} \xrightarrow{m=nm} \sum_{m=-\infty}^{\infty} x(m) e^{-jw\frac{m}{m}}$$

So we write $X_d(e^{jw}) = X_{int}(e^{\frac{jw}{m}})$ ... (1)

let $x_{int}(n) = c(n) \cdot x(n)$, $c(n) = \begin{cases} 1, & n = 0, \pm m, \pm 2m, \cdots, \\ 0, & \text{otherwise} \end{cases}$ also $c(n) = \frac{1}{m} \sum_{k=0}^{m-1} e^{j\frac{2\pi kn}{m}}$.

$$X_{int}(z) = \sum_{n=-\infty}^{\infty} c(n) x(n) z^{-n}$$

$$= \frac{1}{m} \sum_{n=-\infty}^{\infty} \left(x(n) \sum_{k=0}^{m-1} e^{j\frac{2\pi kn}{m}} \right) z^{-n}$$

$$= \frac{1}{m} \sum_{k=0}^{m-1} \sum_{n=-\infty}^{\infty} x(n) \left(z e^{-j\frac{2\pi k}{m}} \right)^{-n} = \frac{1}{m} \sum_{k=0}^{m-1} X\left(z e^{-j\frac{2\pi k}{m}} \right) \quad \cdots (2)$$

From (1), we know $X_d(z) = X_{int}(z^{\frac{1}{m}})$ ... (3)

Substitute (2) into (1): $X_d(z) = \frac{1}{m} \sum_{k=0}^{m-1} X\left(z^{\frac{1}{m}} e^{-j\frac{2\pi k}{m}} \right)$

$z = e^{jw}$

Therefore, $X_d(e^{jw}) = \frac{1}{m} \sum_{k=0}^{m-1} X\left(e^{j\frac{w-2\pi k}{m}} \right) \xrightarrow{k=i} \frac{1}{m} \sum_{i=0}^{m-1} X\left(e^{j\frac{w-2\pi i}{m}} \right)$

(b) $X_a(z) = X(z) \cdot H(z^m)$

$$Y_1(z) = \frac{1}{m} \sum_{k=0}^{m-1} X_a\left(z^{\frac{1}{m}} e^{-j\frac{2\pi k}{m}} \right) = \frac{1}{m} \sum_{k=0}^{m-1} \left[X\left(z^{\frac{1}{m}} \cdot e^{-j\frac{2\pi k}{m}} \right) \cdot H\left(z^{\frac{m}{m}} \cdot e^{-j\frac{2\pi km}{m}} \right) \right]$$

due to that $e^{-j2\pi k} = 1$, So $Y_1(z) = \frac{1}{m} \sum_{k=0}^{m-1} \left[X\left(z^{\frac{1}{m}} e^{-j\frac{2\pi k}{m}} \right) \cdot H(z) \right]$

$z = e^{jw}$, $Y_1(e^{jw}) = \frac{1}{m} H(e^{jw}) \sum_{k=0}^{m-1} \left[X\left(e^{j\frac{w-2\pi k}{m}} \right) \right]$ ... (4)

$$X_b(z) = \frac{1}{m} \sum_{k=0}^{m-1} X\left(z^{\frac{1}{m}} \cdot e^{-j\frac{2\pi k}{m}} \right)$$

$$Y_2(z) = X_b(z) \cdot H(z)$$

$$= \frac{1}{m} H(z) \sum_{k=0}^{m-1} X\left(z^{\frac{1}{m}} e^{-j\frac{2\pi k}{m}} \right)$$

$z = e^{jw}$, $Y_2(e^{jw}) = \frac{1}{m} H(z) \sum_{k=0}^{m-1} \left[X\left(e^{j\frac{w-2\pi k}{m}} \right) \right]$ ... (5)

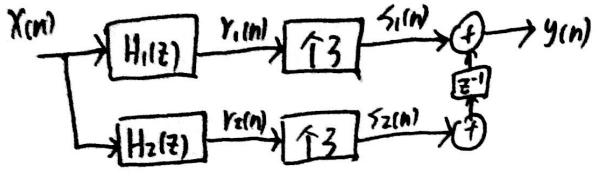
From (4) and (5),

So $Y_1(e^{jw}) = Y_2(e^{jw}) \Rightarrow y_1(n) = y_2(n)$

Two systems are equivalent.

Scanned with CamScanner

(c) The efficient polyphase structure with two polyphase components is:



$$h(n) = \delta(n) + 0.5\delta(n-1) + 0.25\delta(n-2) + 0.125\delta(n-3) + 0.0625\delta(n-4) + 0.03125\delta(n-5) + 0.015625\delta(n-6) + 0.0078125\delta(n-7)$$

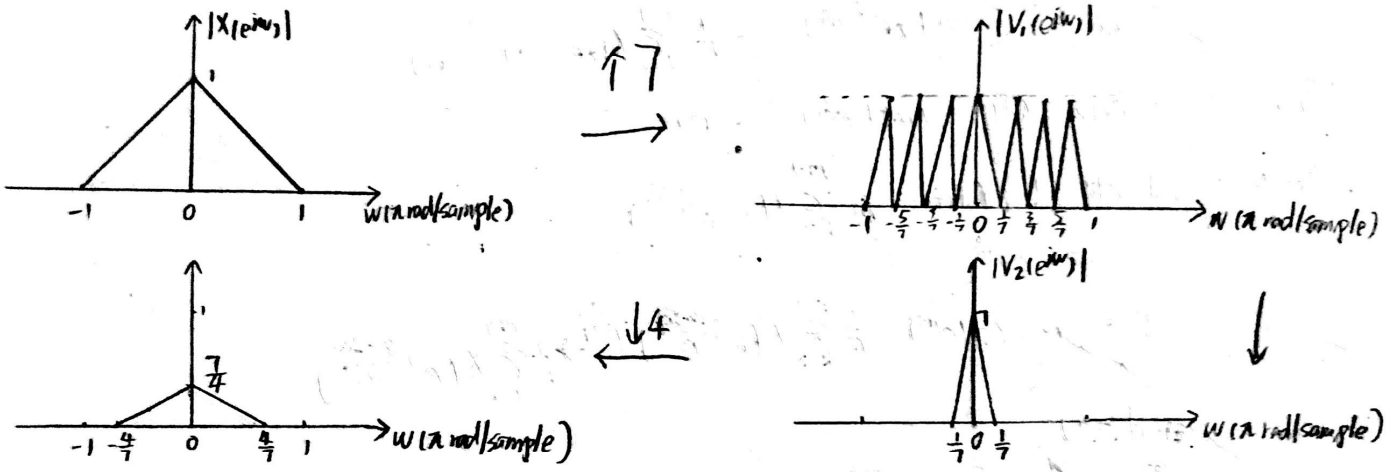
So the filter impulse response for each polyphase component is:

$$h_1(n) = \delta(n) + 0.25\delta(n-2) + 0.0625\delta(n-4) + 0.015625\delta(n-6)$$

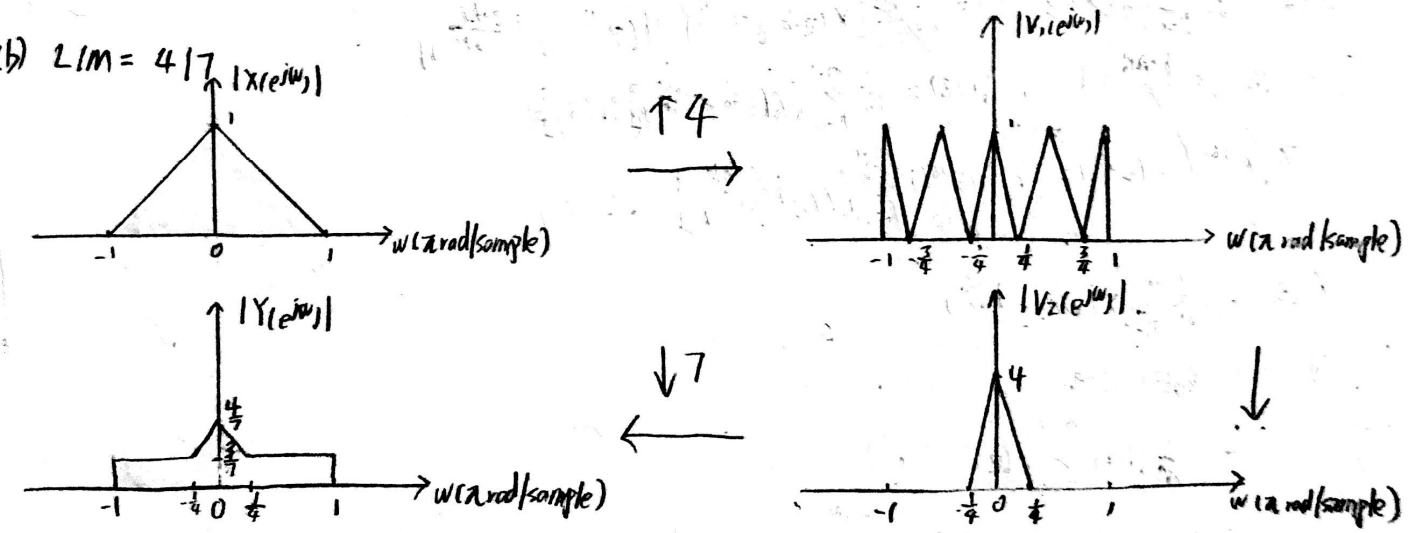
$$h_2(n) = 0.5\delta(n) + 0.125\delta(n-2) + 0.03125\delta(n-4) + 0.0078125\delta(n-6)$$

3. Solve:

(a) $L/M = 7/4$



(b) $L/M = 4/7$



The new maximum magnitude becomes L/M

Scanned with CamScanner

4. Solve:

(a)

The Matlab code is shown as follows:

```
f=[0 0.1 0.2 0.3 0.4 0.5 0.6 0.8 1];
m=[1 0.8 0.6 0.4 0.2 0.0 0.0 0.0 0];
b=fir2(128,f,m);          % Use the Matlab routine fir2
w=pi*(0:0.005:1);
h=freqz(b,1,w);
hmag=abs(h);
figure(1)                 %  Plot the magnitude response of the
                             resulting signal
plot(w/pi,hmag);
title('Magnitude Spectrum Plot')
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Magnitude')
```

The figure of the magnitude response of the resulting signal is shown in Fig. 14:

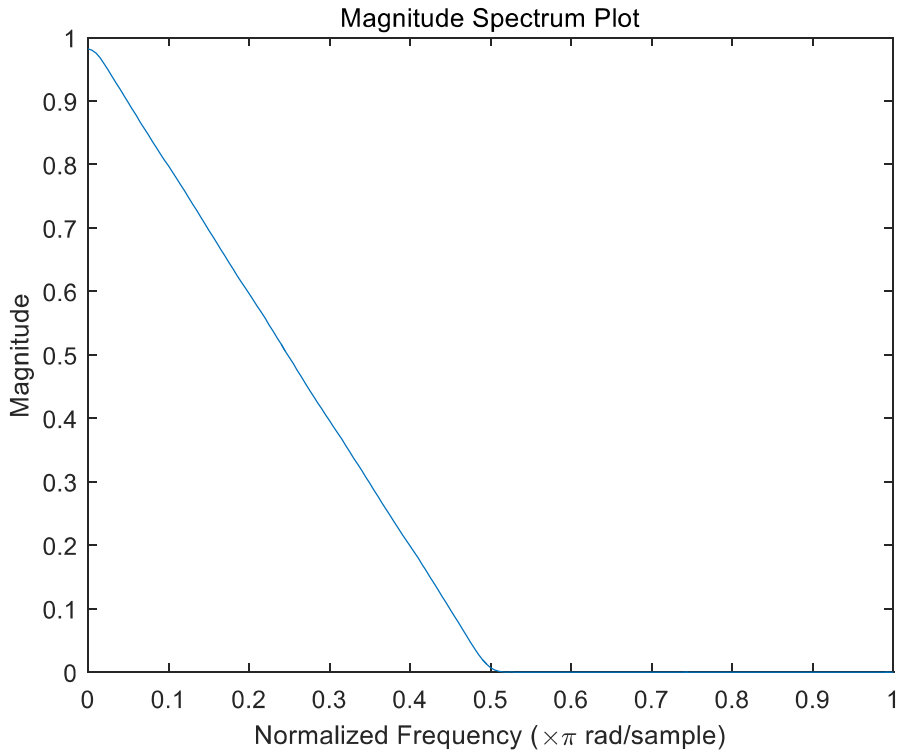


Fig. 14. The magnitude response of the resulting signal

(b)

The Matlab code of my own function, aiming at implementing the up-sampling process, is shown as follows:

```
function b_new=up_sample(b,L,w)
num=length(b);
```

```
num1=L*num;
b_new=zeros(1,num1);
b_new(1,1:L:num1)=b;     % Pad zeros accordingly
h_new=freqz(b_new,1,w);
hmag_new=abs(h_new);
figure(L)
plot(w/pi,hmag_new);
title('Magnitude Spectrum Plot')
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Magnitude')
end
b1_new=up_sample(b,2,w);      % L=2
b2_new=up_sample(b,3,w);      % L=3
b3_new=up_sample(b,4,w);      % L=4
```

The figures of the magnitude response of the resulting up-sampled signal for L=2, 3, and 4 are shown in Fig. 15, Fig. 16, and Fig. 17 respectively:

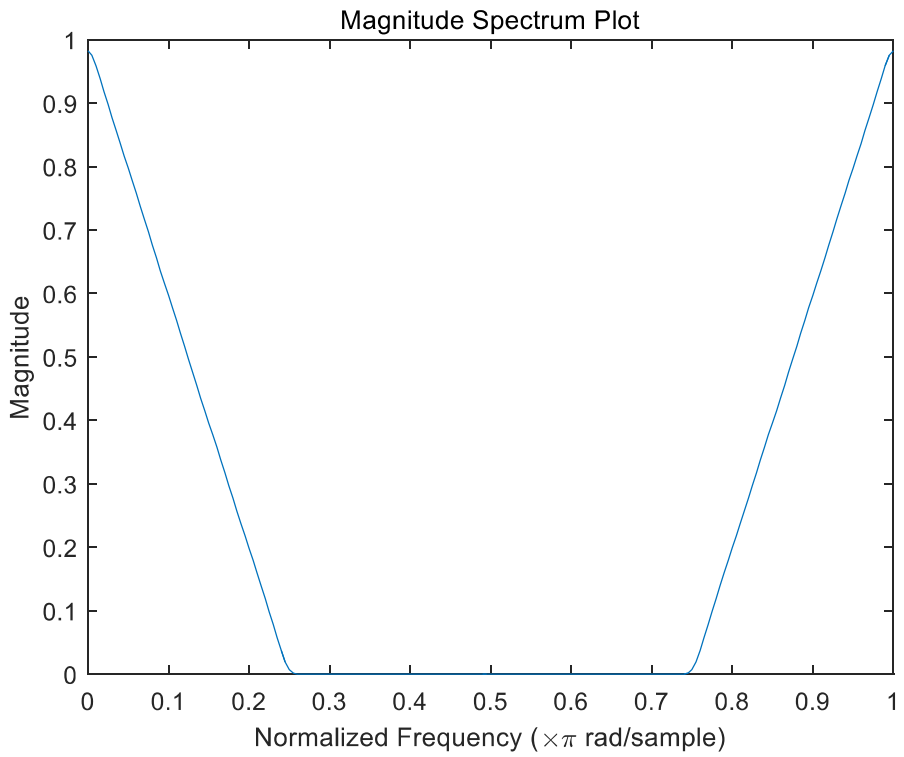


Fig. 15. The magnitude response of the resulting up-sampled signal for L=2

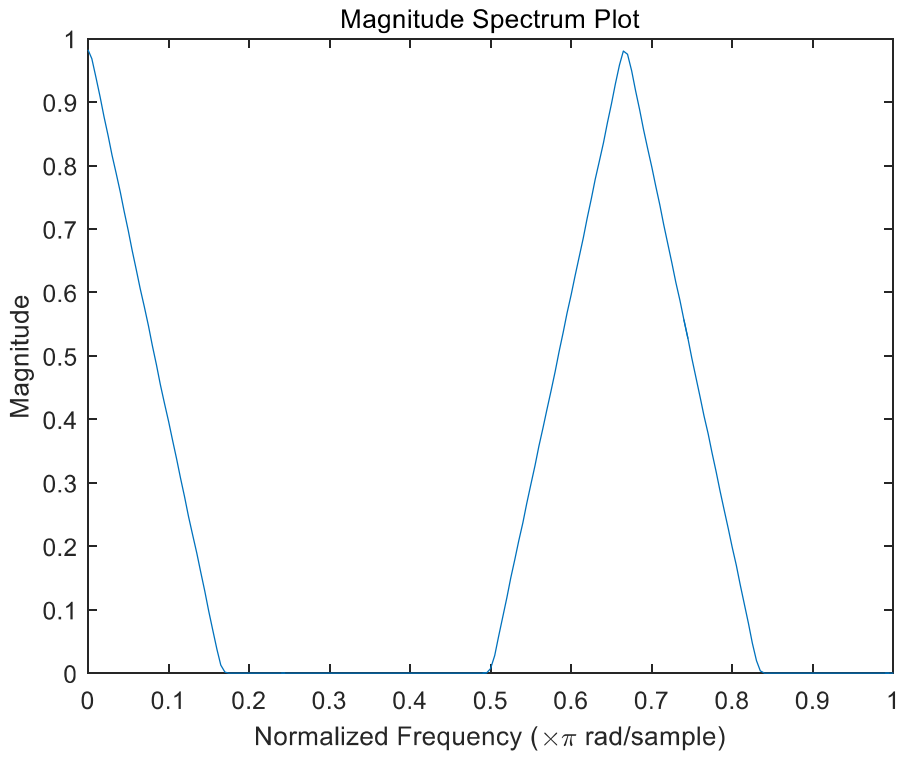


Fig. 16. The magnitude response of the resulting up-sampled signal for L=3

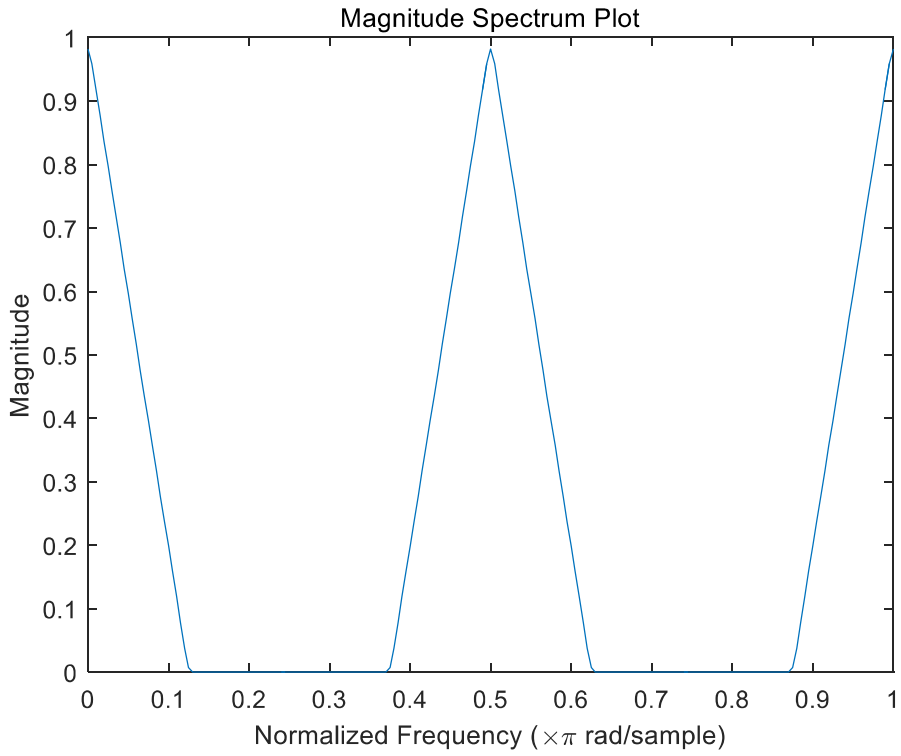


Fig. 17. The magnitude response of the resulting up-sampled signal for L=4

As the value of L increases, the spectrum will become narrower and never have aliasing.

(c)

Use the Matlab routines **firpmord** and **firpm** to generate the appropriate filter for interpolation by a factor of L = 2. The Matlab code is shown as follows:

```
fs=10000;           % The sample rate
ws=0.25;             % The stop band edge frequency
wc=0.25-1500/fs;    % The cut-off frequency
delp=0.01;
dels=0.001;
fedge=[wc ws];
mval=[1 0];
dev=[delp dels];
[A,Fo,Ao,W]=firpmord(fedge,mval,dev);
b_b=firpm(A,Fo,Ao,W);
w=pi*(0:0.005:1.0);
h_b=freqz(b_b,1,w);
hmag_b=abs(h_b);
hphase_b=angle(h_b);
tol=0.95*pi;
figure(5)
subplot(2,1,1)      % Plot the magnitude response of the
                    resulting filter
hmag_b=unwrap(hmag_b,tol);
plot(w/pi,hmag_b)
title('Magnitude Response Plot (firpm FIR filter)')
ylabel('Magnitude ')
xlabel('Frequency (\times\pi radians)')
subplot(2,1,2)      % Plot the phase response of the resulting
filter
hphase_b=unwrap(hphase_b,tol);
plot(w/pi,hphase_b)
title('Phase Response Plot (firpm FIR filter)')
ylabel('Phase (radians)')
xlabel('Frequency (\times\pi radians)')
```

The figure of the magnitude and phase response of the resulting filter is shown in Fig. 18

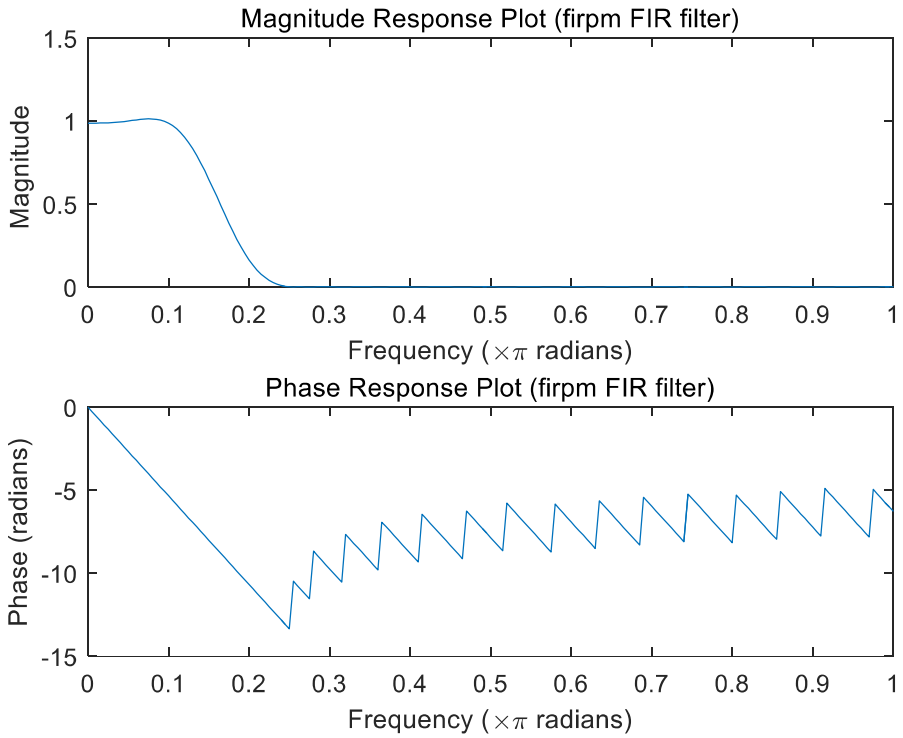


Fig. 18. The magnitude and phase response of the resulting filter

(d)

Apply the filter generated in (c) to the up-sampled sequence produced for L = 2 in (b). The Matlab code is shown as follows:

```
y2_new= filter(b_b,1,b1_new);
b0_new=zeros(1,length(y2_new));
b0_new(1,1:2:length(y2_new))=b;     % Align the sequences
                                      accordingly
stem(0:length(y2_new)-1,abs(y2_new-b0_new))
title('The absolute error between y2[n] and x[n/2]')
xlabel('Sample')
ylabel('Amplitude')
```

The figure of the absolute error between the up-sampled sequence y2[n] and the original sequence x[n/2] is shown in Fig. 19:

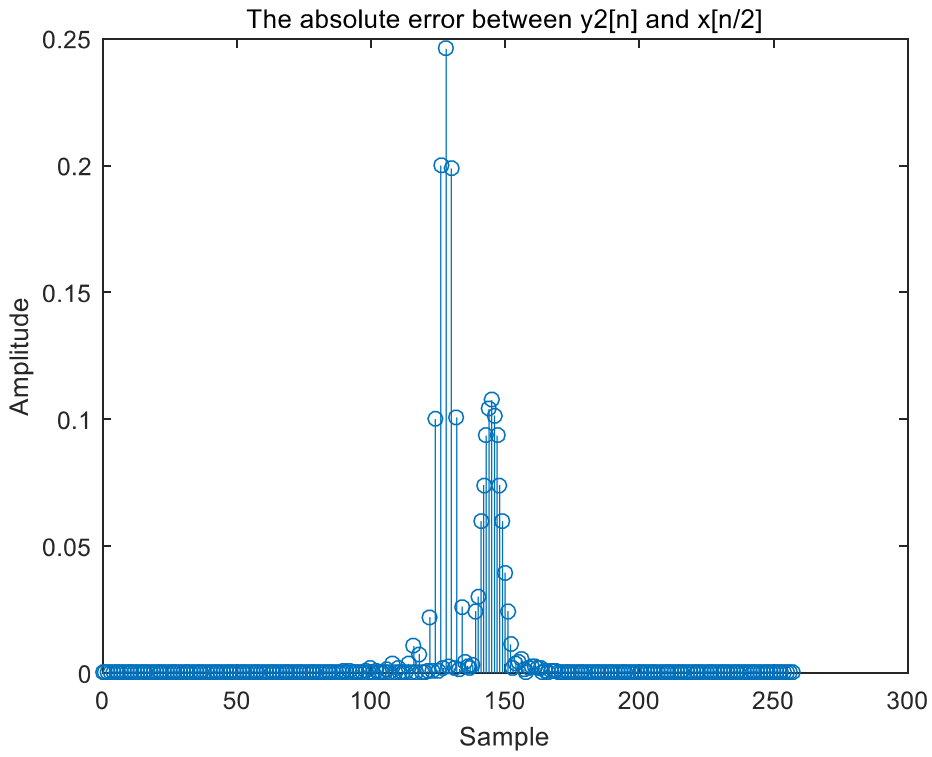


Fig. 19. the absolute error between the up-sampled sequence and the original sequence

As is clearly demonstrated, the absolute error is very small so that people can actually ignore it while conducting research. It's okay for us to draw a conclusion that the two samples are equivalent.

5. Solve:

(a)

The Matlab code to design a filter meeting required specifications is shown as follows:

```
Fs=10000;
Fpass=45;
Fstop=50;
delp=0.01;
dels=0.001;
mval=[1 0];
dev=[delp dels];
[A,Fo,Ao,W]=firpmord([Fpass Fstop],mval,dev,Fs);
b1=firpm(A,Fo,Ao,W);
```

Moreover, I can get the number of multiplications per second needed to implement this single stage decimator as follows:

Number of filter coefficients: **5084**

Number of samples generated per second: **10000**

Number of samples processed at a decimation if 100: **100**

Number of multiplications per second: **508400**

(b)

The Matlab code to implement a 2-stage decimator is shown as follows:

```
M_b1=10;M_b2=10;
F_b0=Fs;
F_b1=F_b0/M_b1;
F_b2=F_b1/M_b2;
% The first filter
Fpass_b1=Fpass;
Fstop_b1=F_b1-Fstop;
delp_b1=delp/2;
dels_b1=dels;
mval=[1 0];
dev=[delp_b1 dels_b1];
[A,Fo,Ao,W]=firpmord([Fpass_b1 Fstop_b1],mval,dev,F_b0);
b_b1=firpm(A,Fo,Ao,W);
% The second filter
Fpass_b2=Fpass;
Fstop_b2=F_b2-Fstop;
delp_b2=delp/2;
dels_b2=dels;
mval=[1 0];
dev=[delp_b2 dels_b2];
[A,Fo,Ao,W]=firpmord([Fpass_b2 Fstop_b2],mval,dev,F_b1);
```

```
b_b2=firpm(A,Fo,Ao,W);
```

Moreover, I can get the number of multiplications per second needed to implement this 2-stage decimator as follows:

Number of filter coefficients for first stage: **31**

Number of filter coefficients for second stage: **553**

Number of samples per sec possessed at 1st stage M1=10: **1000**

Number of multiplications per second at first stage: **31000**

Number of samples per sec possessed at 2nd stage M2=10: **100**

Number of multiplications per second at second stage: **55300**

Number of multiplications per second: **86300**

This results in a saving of

Ratio=508400/86300=5.89,

which highly increases the efficiency of the operation.

(c)

The Matlab code to implement a 3-stage decimator is shown as follows:

```
M_c1=25;M_c2=2;M_c3=2;
F_c0=Fs;
F_c1=F_c0/M_c1;
F_c2=F_c1/M_c2;
F_c3=F_c2/M_c3;
% The first filter
Fpass_c1=Fpass;
Fstop_c1=F_c1-Fstop;
delp_c1=delp/2;
dels_c1=dels;
mval=[1 0];
dev=[delp_c1 dels_c1];
[A,Fo,Ao,W]=firpmord([Fpass_c1 Fstop_c1],mval,dev,F_c0);
b_c1=firpm(A,Fo,Ao,W);
% The second filter
Fpass_c2=Fpass;
Fstop_c2=F_c2-Fstop;
delp_c2=delp/2;
dels_c2=dels;
mval=[1 0];
dev=[delp_c2 dels_c2];
[A,Fo,Ao,W]=firpmord([Fpass_c2 Fstop_c2],mval,dev,F_c1);
b_c2=firpm(A,Fo,Ao,W);
% The third filter
Fpass_c3=Fpass;
Fstop_c3=F_c3-Fstop;
delp_c3=delp/2;
dels_c3=dels;
```

```
mval=[1 0];
dev=[delp_c3 dels_c3];
[A,Fo,Ao,W]=firpmord([Fpass_c3 Fstop_c3],mval,dev,F_c2);
b_c3=firpm(A,Fo,Ao,W);
```

Moreover, I can get the number of multiplications per second needed to implement this 2-stage decimator as follows:

Number of filter coefficients for first stage: **92**
Number of filter coefficients for second stage: **9**
Number of filter coefficients for third stage: **112**
Number of samples per sec possessed at 1st stage M1=25: **400**
Number of multiplications per second at first stage: **36800**
Number of samples per sec possessed at 2nd stage M2=2: **200**
Number of multiplications per second at second stage: **1800**
Number of samples per sec possessed at 3rd stage M3=2: **100**
Number of multiplications per second at second stage: **11200**
Number of multiplications per second: **49800**
This results in a saving of

Ratio1=508400/49800=10.21,
Ratio2=86300/49800=1.73,

which highly increases the efficiency of the operation compared with the previous two methods.

(d)

The Matlab code to plot three sorts of overall frequency response is shown as follows:

```
% Frequency response of single stage implementation in part(a)
w=pi*(0:0.005:1.0);
h1=freqz(b1,1,w);
hphase=angle(h1);
hmag=20*log10(abs(h1));
tol=0.5*pi;
figure(1)
subplot(2,1,1)    % Plot the magnitude response of the
resulting filter
plot(w/pi,hmag)
ylim([-400 0])
title('Magnitude Response Plot (firpm FIR filter)')
ylabel('Magnitude ')
xlabel('Frequency (\times\pi radians)')
subplot(2,1,2)    % Plot the phase response of the resulting
filter
hphase=unwrap(hphase,tol);
plot(w/pi,hphase)
title('Phase Response Plot (firpm FIR filter)')
ylabel('Phase (radians)')
```

```
xlabel('Frequency (\times\pi radians)')
% Frequency response of multistage implementation in part(b)
b_b2_up=upsample(b_b2,M_b1);
h_casc_b=conv(b_b1,b_b2_up);
h2=freqz(h_casc_b,1,w);
hphase=angle(h2);
hmag=20*log10(abs(h2));
figure(2)
subplot(2,1,1)    % Plot the magnitude response of the
resulting filter
plot(w/pi,hmag)
title('Magnitude Response Plot (firpm FIR filter)')
ylabel('Magnitude ')
xlabel('Frequency (\times\pi radians)')
subplot(2,1,2)    % Plot the phase response of the resulting
filter
hphase=unwrap(hphase,tol);
plot(w/pi,hphase)
title('Phase Response Plot (firpm FIR filter)')
ylabel('Phase (radians)')
xlabel('Frequency (\times\pi radians)')
% Frequency response of multistage implementation in part(c)
b_c2_up=upsample(b_c2,M_c1);
h_casc_c=conv(b_c1,b_c2_up);
b_c3_up=upsample(b_c3,M_c1*M_c2);
h_casc_c=conv(h_casc_c,b_c3_up);
h3=freqz(h_casc_c,1,w);
hphase=angle(h3);
hmag=20*log10(abs(h3));
figure(3)
subplot(2,1,1)    % Plot the magnitude response of the
resulting filter
plot(w/pi,hmag)
ylim([-400 0])
title('Magnitude Response Plot (firpm FIR filter)')
ylabel('Magnitude ')
xlabel('Frequency (\times\pi radians)')
subplot(2,1,2)    % Plot the phase response of the resulting
filter
hphase=unwrap(hphase,tol);
plot(w/pi,hphase)
title('Phase Response Plot (firpm FIR filter)')
ylabel('Phase (radians)')
xlabel('Frequency (\times\pi radians)')
```

The resulting figures are shown in Fig. 20, Fig. 21, and Fig. 22 respectively.

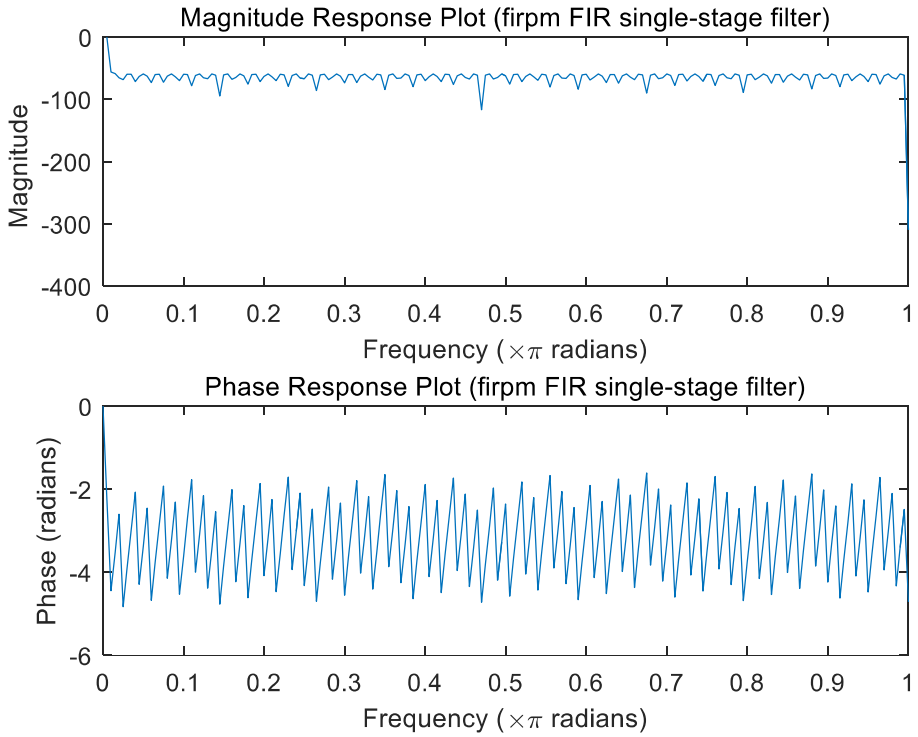


Fig. 20. The magnitude and phase response for the single stage decimator

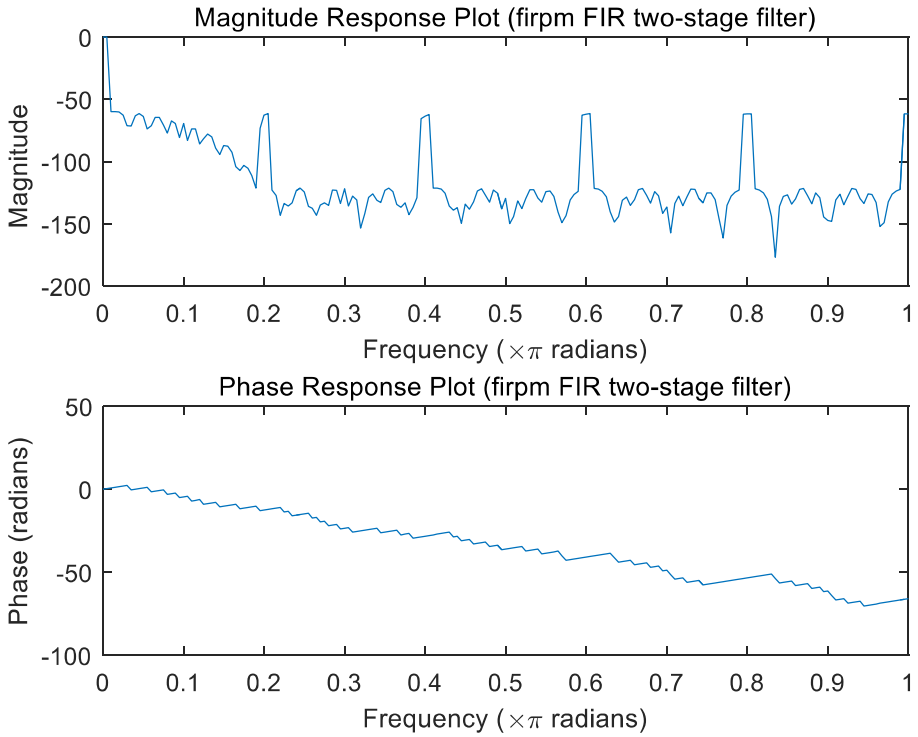


Fig. 21. The magnitude and phase response for the 2-stage decimator

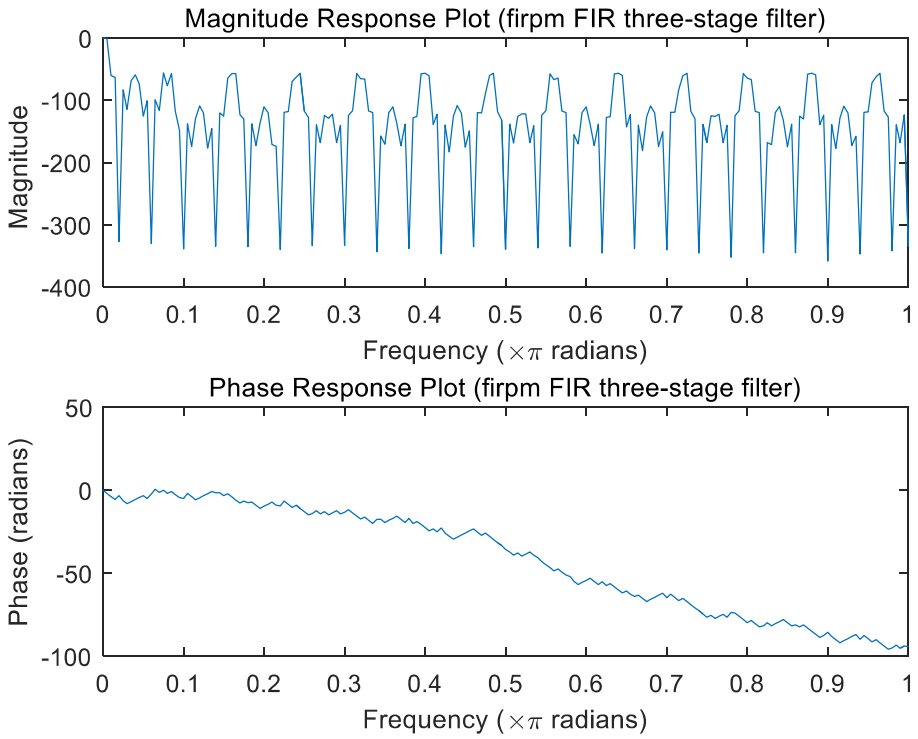


Fig. 22. The magnitude and phase response for the 3-stage decimator

From three figures, within the crucial frequency contend (required passband and transition band), the magnitude and phase response of three figures are almost similar to each other.

6. Solve:

(a)

The Matlab code to design a lowpass filter with required specifications is shown as follows:

```
b0=fir1(20,0.413);
w=pi*(0:0.005:1.0);
h1=freqz(b0,1,w);
hmag1=abs(h1);
hphase1=angle(h1);
tol=0.95*pi;
figure(1)
subplot(2,1,1)   % Plot the magnitude response of the resulting
filter
hmag1=unwrap(hmag1,tol);
plot(w/pi,hmag1)
title('Magnitude Response Plot (firpm FIR filter)')
ylabel('Magnitude ')
xlabel('Frequency (\times\pi radians)')
subplot(2,1,2)    % Plot the phase response of the resulting
filter
hphase1=unwrap(hphase1,tol);
plot(w/pi,hphase1)
title('Phase Response Plot (firpm FIR filter)')
ylabel('Phase (radians)')
xlabel('Frequency (\times\pi radians)')
```

The figure of the magnitude and phase response of the resulting filter is shown in Fig. 23

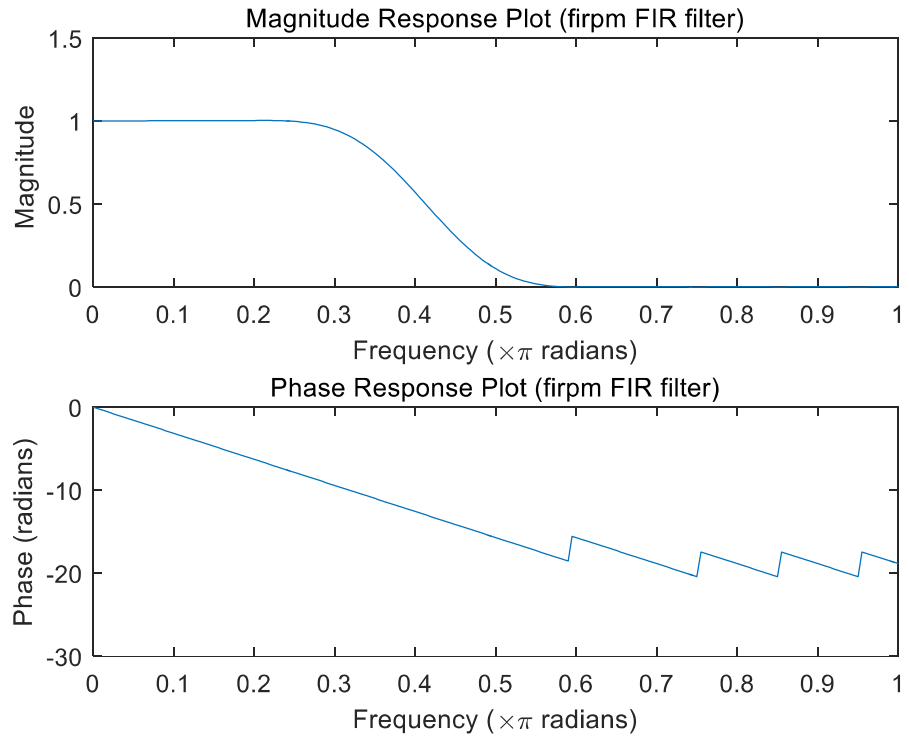


Fig. 23. The magnitude and phase response of the resulting filter

(b)

Use the following Matlab code to determine the coefficients for H1(z), H2(z), and H3(z) for the polyphase implementation of the filter.

```
b1=b0(1,1:3:end);
b2=b0(1,2:3:end);
b3=b0(1,3:3:end);
```

(c)

My own Matlab function to implement a filter using the three level polyphase decomposition of the filter is shown as follows (‘b’ denotes the filter’s coefficients, the ‘x’ denotes the input sequence, the ‘level’ denotes which layer of the polyphase, and the ‘n’ denotes the length of the output sequence):

```
function s=level3_filter(b,x,level,n)
r=conv(b,x);
n1=length(r);
s=zeros(1,n);
n2=3*n1;
s(1,level:3:n2)=r;
end
```

(d)

Use my routine for the polyphase implementation to filter and interpolate the sample sequence. The Matlab code is shown as follows:

```
x_n=sampdata;
figure(2)
stem(0:length(x_n)-1,x_n)
title('x(n) The input sequence')
xlabel('Sample number')
ylabel('Amplitude')
n0=length(b0)+3*length(x_n)-1;
% Implement the polyphase decomposition
s1=level3_filter(b1,x_n,1,n0);
s2=level3_filter(b2,x_n,2,n0);
s3=level3_filter(b3,x_n,3,n0);
y1_n=s1+s2+s3;
figure(3)
stem(0:length(y1_n)-1,y1_n)
title('Output using polyphase with up sampling')
xlabel('Sample number')
ylabel('Amplitude')
```

The figure of the input sequence x(n) is shown in Fig. 24:

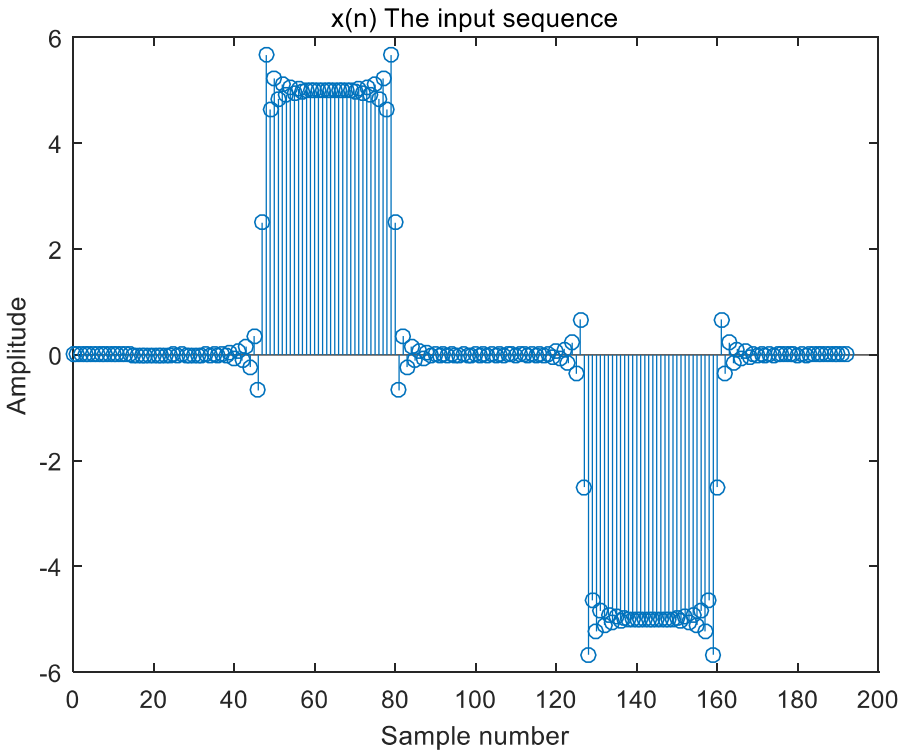


Fig. 24. The input sequence x(n)

The figure of the output sequence y1(n) using polyphase with up sampling is shown in Fig. 25:

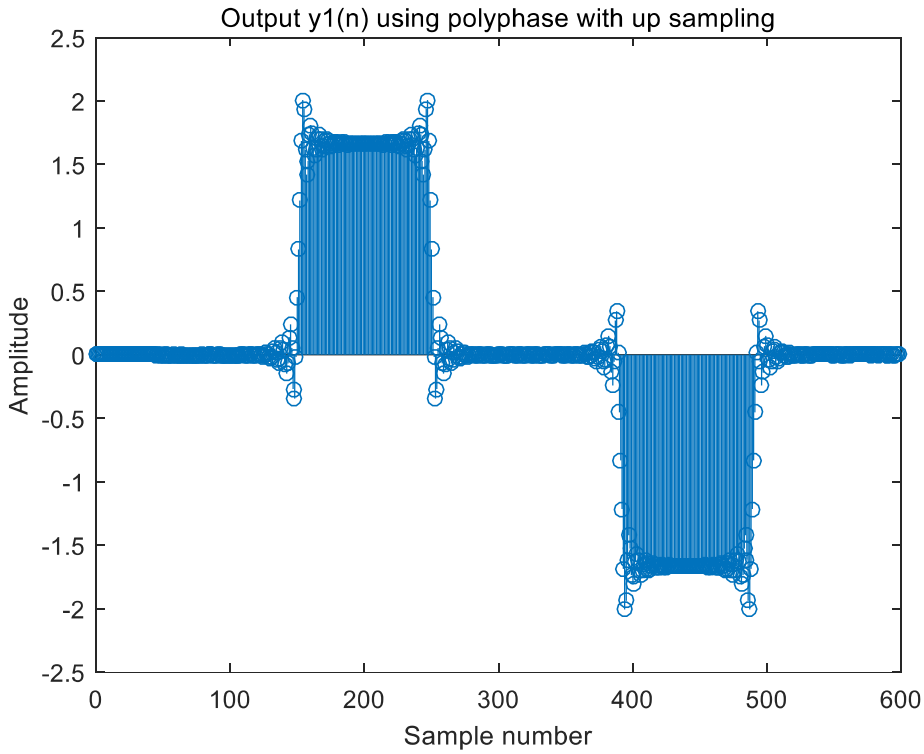


Fig. 25. The output sequence y1(n) using polyphase with up sampling

(e)

Up-sample the sample sequence and then apply the filter designed in (a) directly to the up-sampled sequence using the **conv** function The Matlab code is shown as follows:

```
v=zeros(1,3*length(x_n));
v(1,1:3:end)=x_n;
y2_n=conv(b0,v);
figure(4)
stem(0:length(y2_n)-1,y2_n)
title('Output using up sampling before filtering')
xlabel('Sample number')
ylabel('Amplitude')
```

The figure of the output sequence y2(n) using up sampling before filtering is shown in Fig. 26:

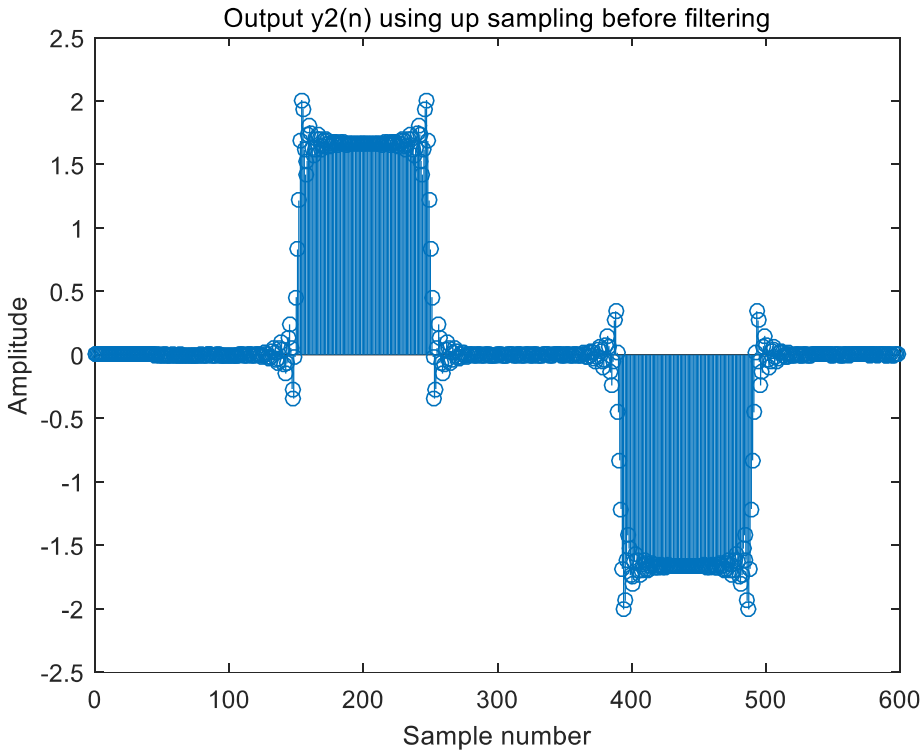


Fig. 26. The output sequence y2(n) using up sampling before filtering

(f)

Compute the absolute error between the computed signals y1(n) and y2(n). The Matlab code is shown as follows:

```
stem(0:length(y1_n)-1,abs(y1_n-y2_n))
title('The absolute error between two methods')
xlabel('Sample number')
ylabel('Absolute error')
```

The figure of the absolute error between y1(n) and y2(n) is shown in Fig. 27:

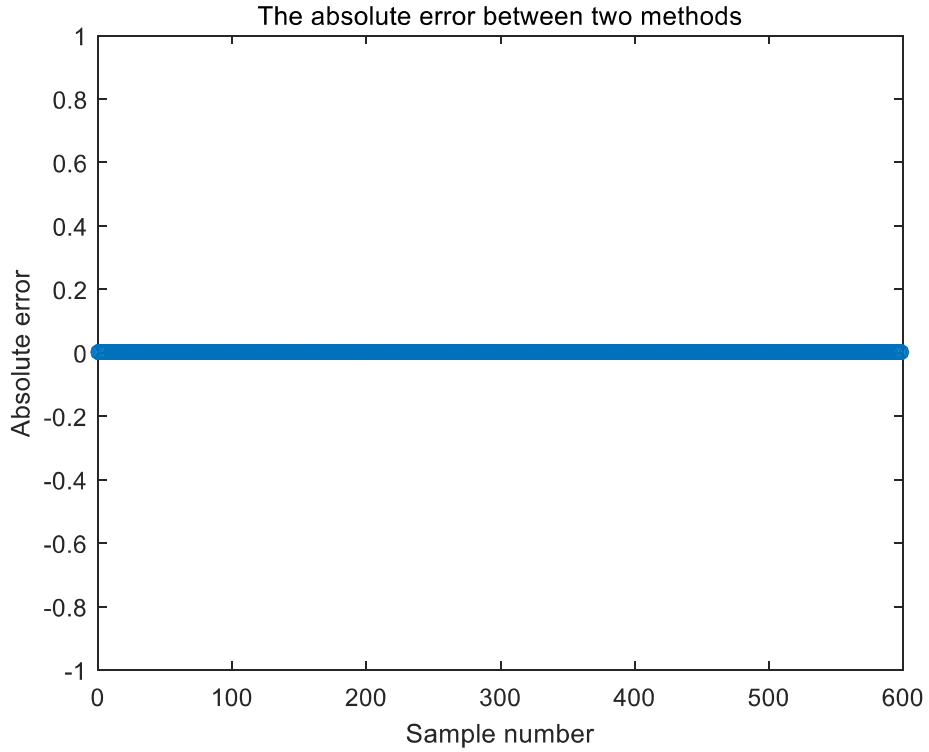


Fig. 27. Absolute Error between y1(n) and y2(n)

As is clearly demonstrated, the absolute error is nearly zero so that people can actually ignore it while conducting research. It's okay for us to draw a conclusion that the two samples are equivalent.

(g)

Compute the number of multiplications and the number of additions required to filter the sequence directly and the number of multiplications and number of additions required to filter the sequence using the polyphaser implementation.

Methods	Additions	Multiplications
Conventional method	(21-1)*193*3=11580	21*193*3=12159
Polyphaser filtering	(7-1)*193*3=3474	7*193*3=4053