Signals and Systems

Lab 04

3190102060 黄嘉欣

Problem 1

Solutions:

(a) Since $h[n] = 2\delta[n+1] - 2\delta[n-1]$ is nonzero only on the interval $[-1,1]$, and $x[n] = \delta[n] + \delta[n-2]$ is nonzero for n in $[0,2]$, we can know the time indexing for $y[n]$ is $[-1,3]$, and the image of $y[n]$ versus n is as follows:

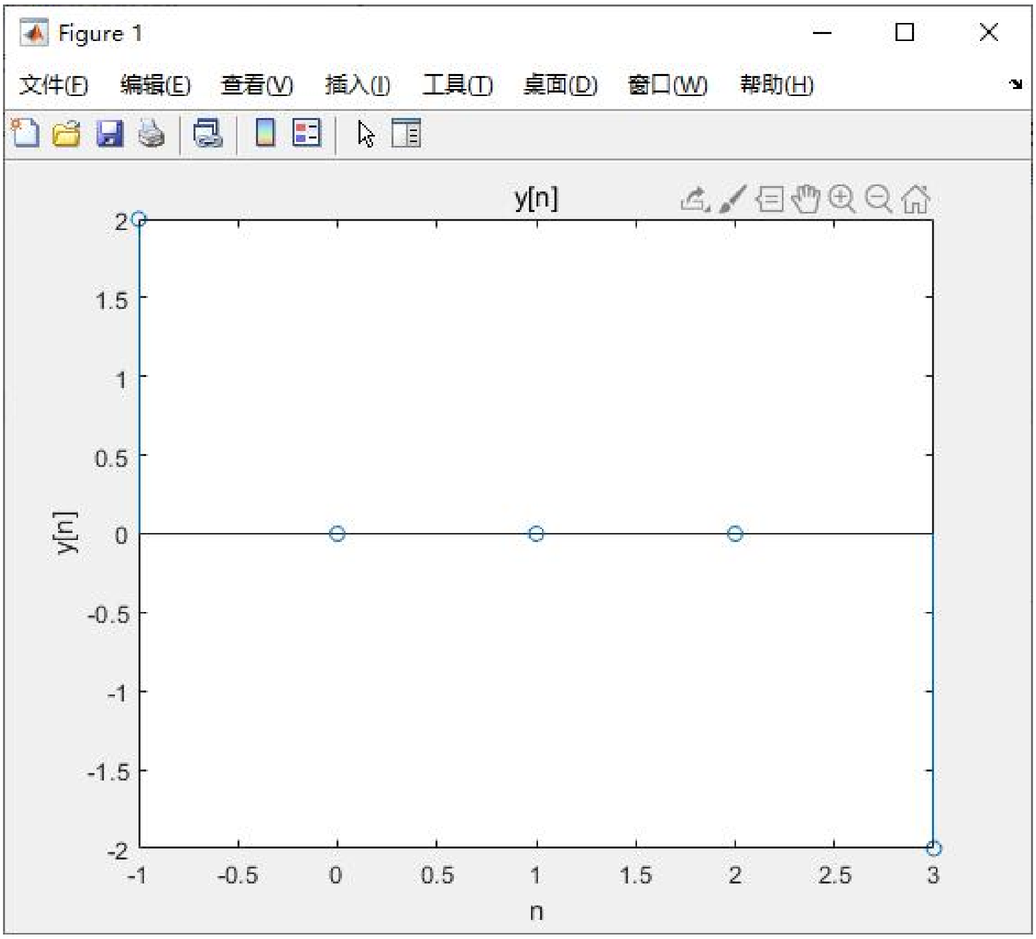

Figure 1.1 image of $y[n]$

MATLAB code:

```
% prob1a.m

clear;
clc;

x=[1 0 1]; % input and impulse response
h=[2 0 -2];
y=conv(h,x); % convolution
ny=-1:3;
stem(ny,y);
title('y[n]');
xlabel('n');
ylabel('y[n]');
```

(b) Analytically,

$$\because h[n] = \delta[n-a] + \delta[n-b], x[n] = \delta[n-c] + \delta[n-d]$$

where $a < b$ and $c < d$,

$$\therefore y[n] = h[n] * x[n] = \sum_{k=-\infty}^{\infty} h[k]x[n-k] = x[n-a] + x[n-b]$$

$$= \delta[n-a-c] + \delta[n-a-d] + \delta[n-b-c] + \delta[n-b-d]$$

It is obvious that $y[n]$ is nonzero only on the interval $[a+c, b+d]$, which means $\mathbf{ny} = [a+c: b+d]$.

When $a = 0, b = N-1, c = 0, d = M-1$, we can see that $\mathbf{ny} = [a+c: b+d] = [0: M+N-2]$, and the length of the sequence $y[n]$ is $M+N-1$, which is the same with the description of the title.

(c) Analytically, if we use the infinite sequence of $h[n]$ and $x[n]$, the output shall be

$$y[n] = h[n] * x[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] = \sum_{k=2}^{n+2} (\frac{1}{2})^{k-2} = 2 - (\frac{1}{2})^{n}$$

Since now $x[n]$ is valid only for $0 \leq n \leq 24$ and $h[n]$ is valid for $0 \leq n \leq 14$, we have(denote $y_*[n]$ as the output for truncated signals),

$$y_*[n] = h[n] * x[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] = \sum_{k=max\{2,n-14\}}^{min\{24,n\}} (\frac{1}{2})^{k-2}$$

The index for the addition may be different for some n, so it is sure that only a portion of the output is valid, and the limitation factor for valid $y[n]$'s indexing is:

$$\begin{cases} n - 14 \leq 2 \\ n \geq 24 \\ n + 2 \leq 24 \end{cases} \quad \text{or} \quad \begin{cases} n - 14 \leq 2 \\ n \leq 24 \\ n + 2 \leq n \end{cases}$$

which has no solution. So there is no value equals to the origin signal when $x[n]$ and $h[n]$ are truncated. However, as k increases, $(\frac{1}{2})^{k-2}$ becomes extremely small. For n is sufficiently large(like 8 yet no bigger than 16), the sum of $(\frac{1}{2})^{n-1}$ and $(\frac{1}{2})^{n}$ is so small that there seems no difference between the two signals. If $n > 16$, the truncated signal will surely smaller than the origin one because the sum of the previous items is very different.

From the title we know that $a = 0, b = 24, c = 0, d = 14$, so the time indices for $y[n]$ is $[0:38]$. The image of $y[n]$ and $y_*[n]$ are as follows, in which the differences between the two outputs have drawn out:

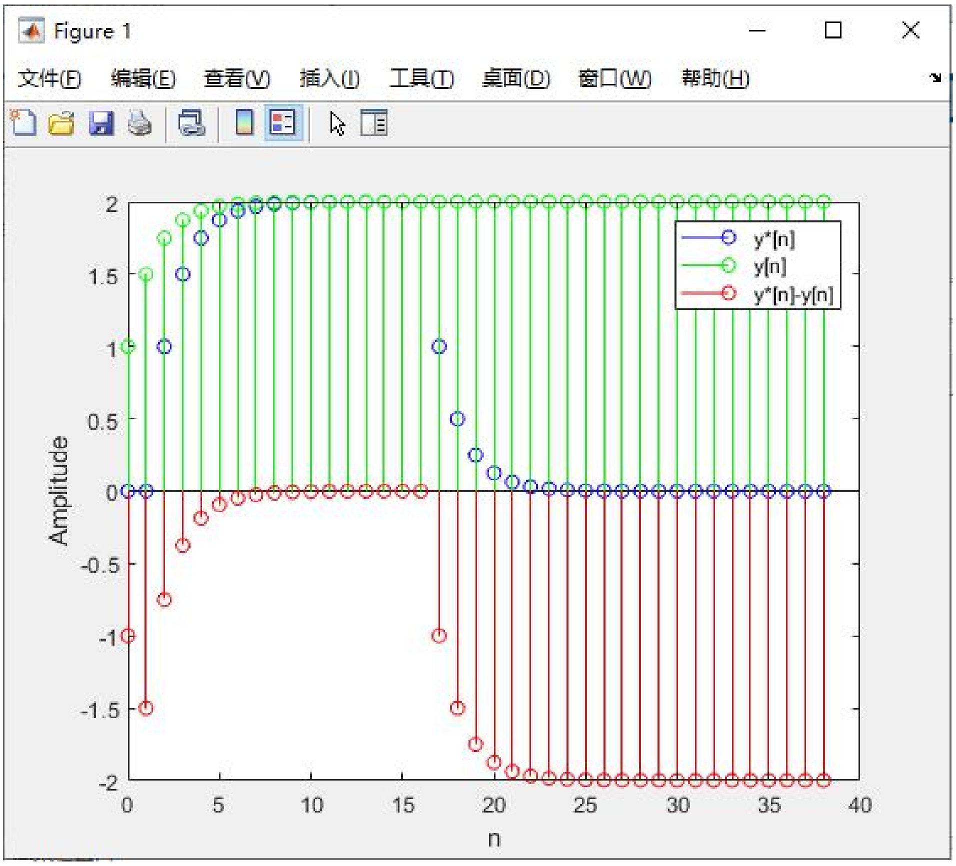

Figure 1.2 image of $y[n]$ and $y_*[n]$

From the figure we can see that for about $8 \leq n \leq 16$, the value of $y_*[n]$ are "almost" correct, while the others are invalid.

MATLAB code:

```
% prob1c.m

clear;
clc;

nx=0:24;
x=(0.5.^(nx-2)).*(nx>=2); % x[n]

nh=0:14;
h=ones(1,15); % h[n]

y=conv(h,x);
```

```
ny=0:38;
stem(ny,y,'b'); % truncated output

hold on;
stem(ny,2-2.^(-ny),'g'); % analytical output

hold on;
stem(ny,y-2+2.^(-ny),'r');  % difference between the two
                            % outputs
xlabel('n');
ylabel('Amplitude');
legend('y*[n]','y[n]','y*[n]-y[n]');
```

(d) Using MATLAB, the image of $y[n]$ is as follows:

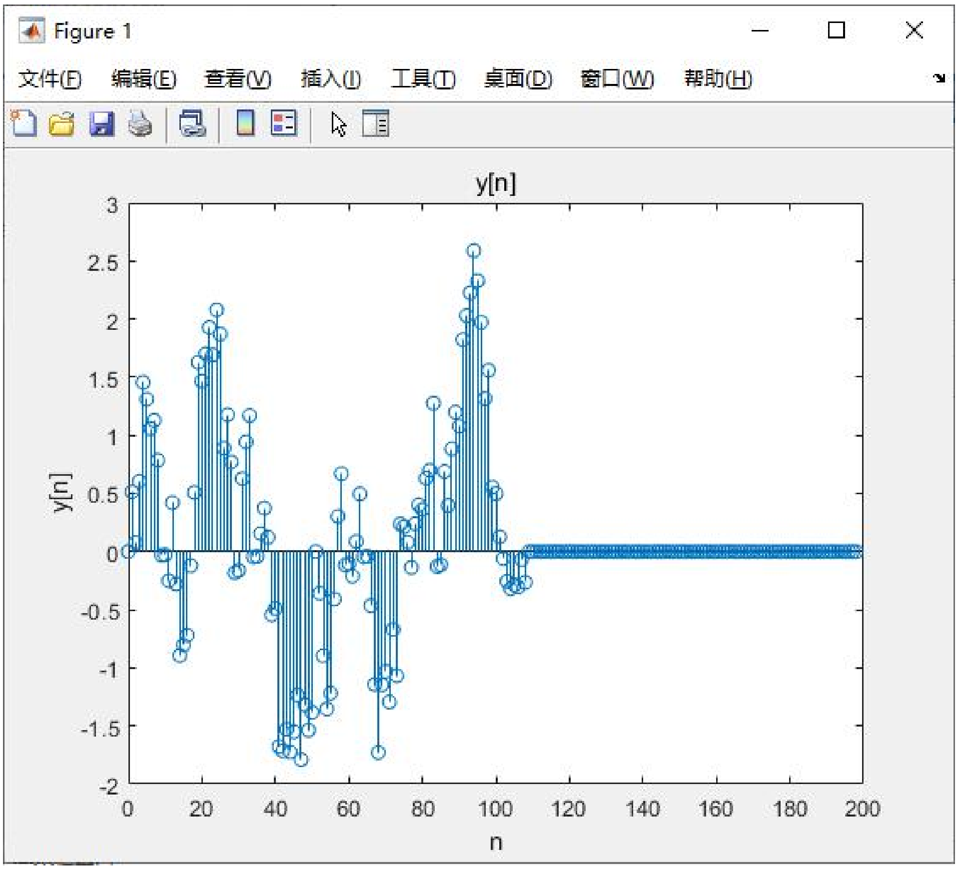

Figure 1.3 image of $y[n]$

MATLAB code:

```
% prob1d.m
```

```
clear;
clc;

nx=0:99;
x=cos(nx.^2).*sin(2*pi/5.*nx); % x[n]
nh=0:99;
h=0.9.^nh.*(nh>=0 & nh<=9); % h[n]

y=conv(h,x);
ny=0:198; % range for y[n]
stem(ny,y);
xlabel('n');
ylabel('y[n]');
title('y[n]');
```

(e) Since $L = 50$, we can know that $k = L = 50$, and the image of $y[n]$ by using the overlap-add method is as follows, which is the same as the result we got in part (d) over the range $0 \leq n \leq 99$:

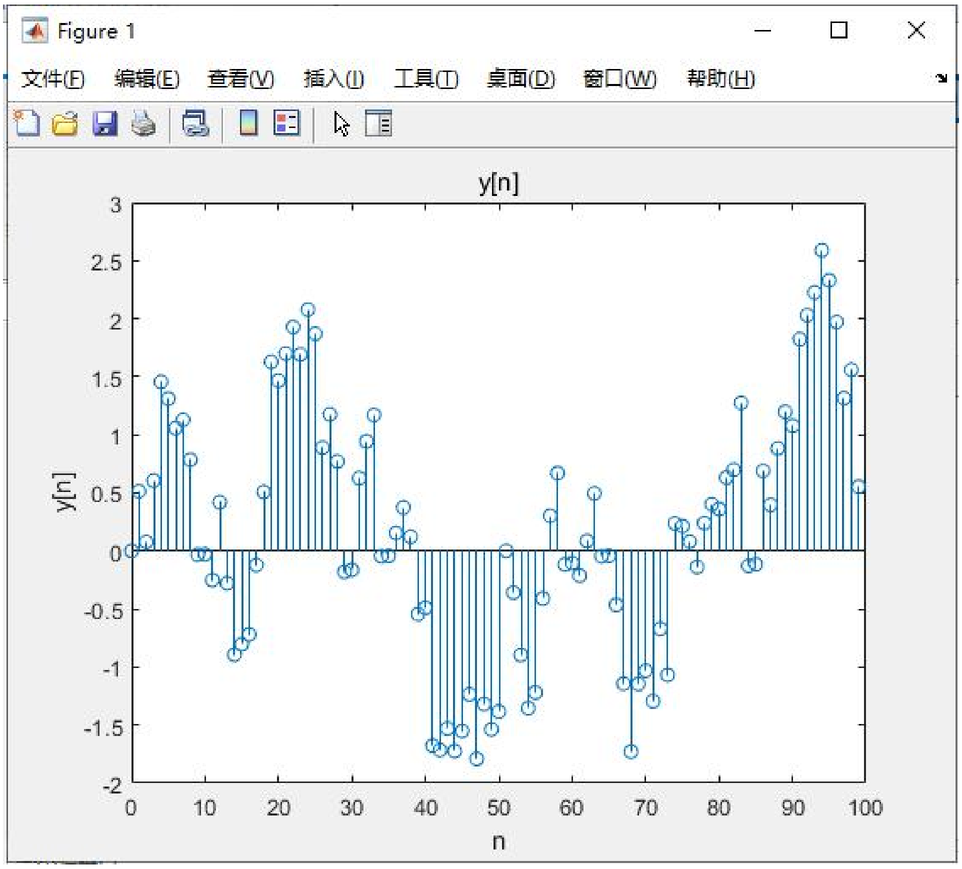

Figure 1.4 image of $y[n]$

MATLAB code:

```
% prob1e.m

clear;
clc;

n=0:99;
h=0.9.^n.*(n>=0&n<10); % h[n]

nx0=0:49;
x0=cos(nx0.^2).*sin(2*pi/5.*nx0); % x0[n]
nx1=nx0+50;
x1=cos(nx1.^2).*sin(2*pi/5.*nx1); % x1[n]

y0=conv(h,x0); % y0[n]
y1=conv(h,x1); % y1[n]
y=[y0 zeros(1,50)]+[zeros(1,50) y1]; % y0[n]+y1[n-50]
stem(n,y(1:100));
xlabel('n');
ylabel('y[n]');
title('y[n]');
```

Tutorial

Solutions:

(a) Since $y[n]-0.8y[n-1]=2x[n]-x[n-2]$,

we have $a_1=[1\ \ -0.8]$ and $b_1=[2\ 0\ \ -1]$.

(b) Using MATLAB, we get the values of **H1** and **omega1** as follows:

```
H1 =                    omega1 =

   5.0000 + 0.0000i          0
   2.8200 - 1.3705i     0.7854
   1.8293 - 1.4634i     1.5708
   0.9258 - 0.9732i     2.3562
```

Figure T.1 values of H1 and omega1

which is the same with the result provided by the title.

MATLAB code:

```
% tutorial_b.m

clear;
clc;

a1=[1 -0.8];
b1=[2 0 -1]; % the coefficients
[H1 omega1]=freqz(b1,a1,4) % use the freqz
```

(c) Similarly, the values of **H2** and **omega2** are:

H2 =	omega2 =
5.0000 + 0.0000i	0
1.8293 - 1.4634i	1.5708
0.5556 + 0.0000i	3.1416
1.8293 + 1.4634i	4.7124

Figure T.2 values of H2 and omega2

which are correct.

MATLAB code:

```
% tutorial_c.m

clear;
clc;

a1=[1 -0.8];
b1=[2 0 -1]; % the coefficients
[H2 omega2]=freqz(b1,a1,4,'whole') % use the freqz
```

Problem 2

Solutions:

(a) MATLAB code:

```
% prob2a.m

clear;
clc;

wc=0.4; % cutoff frequency
n1=10;n2=4;n3=12; % orders
[b1,a1]=butter(n1,wc); % coefficients
a2=1;b2=remez(n2,[0 wc-0.04 wc+0.04 1],[1 1 0 0]);
a3=1;b3=remez(n3,[0 wc-0.04 wc+0.04 1],[1 1 0 0]);
```

(b) Using MATLAB, we obtained the magnitude and phase of the three filters as follows:

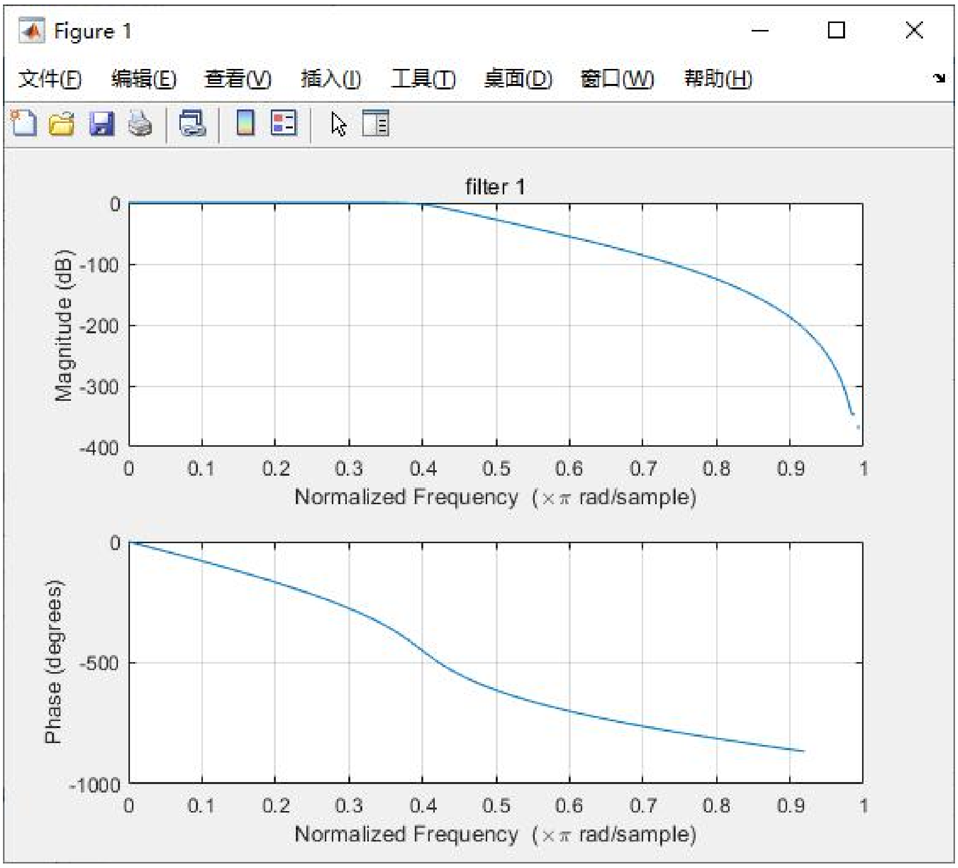

Figure 2.2.1 magnitude and phase of filter 1

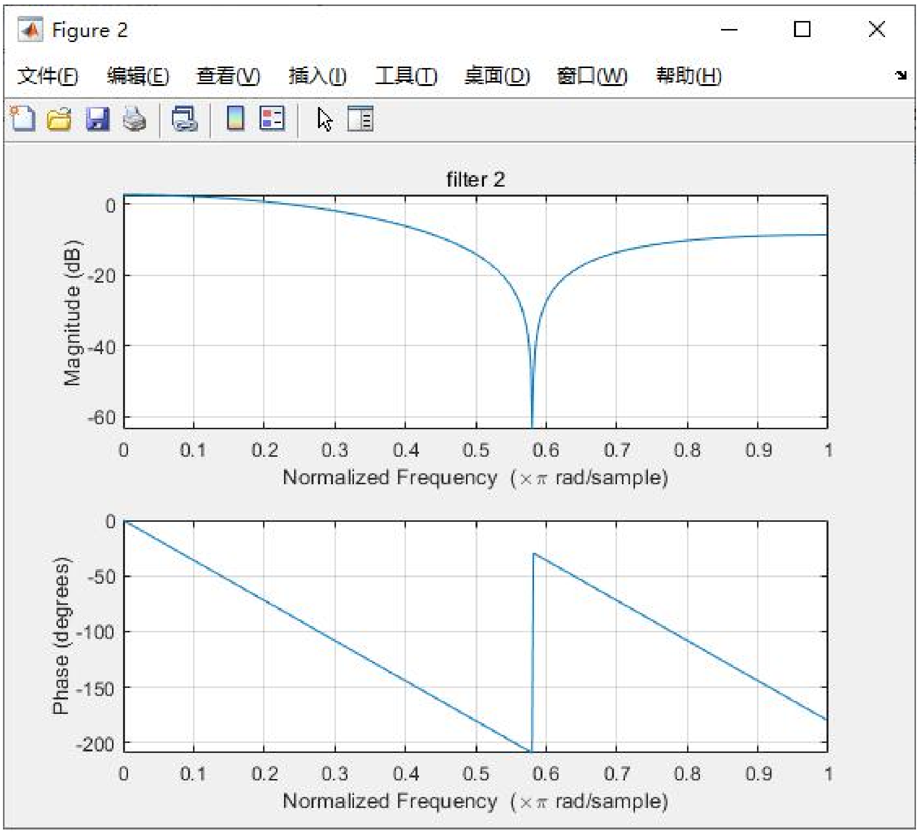

Figure 2.2.2 magnitude and phase of filter 2

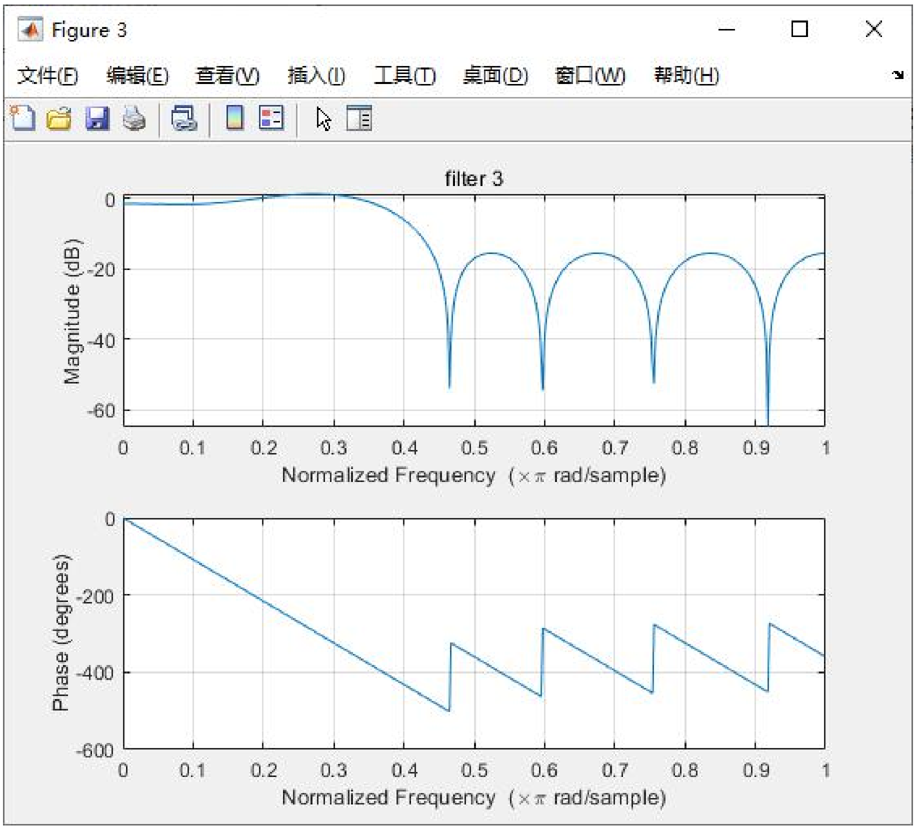

Figure 2.2.3 magnitude and phase of filter 3

Obviously, we can see that **wc** is the approximate cutoff frequency of each filter from the magnitude plot(for filter 2 and 3, since they are finite-length, the cutoff frequencies are not exactly **wc**). And filter 2 and 3 have linear phase.

MATLAB code:

```
% prob2b.m

clear;
clc;

wc=0.4;
n1=10;n2=4;n3=12;
[b1,a1]=butter(n1,wc);
a2=1;b2=remez(n2,[0 wc-0.04 wc+0.04 1],[1 1 0 0]);
a3=1;b3=remez(n3,[0 wc-0.04 wc+0.04 1],[1 1 0 0]);

figure; % filter_1
freqz(b1,a1);
title('filter 1');

figure; % filter_2
freqz(b2,a2);
title('filter 2');

figure;% filter_3
freqz(b3,a3);
title('filter 3');
```

(c) Using MATLAB, we got the step response of the filters as follows:

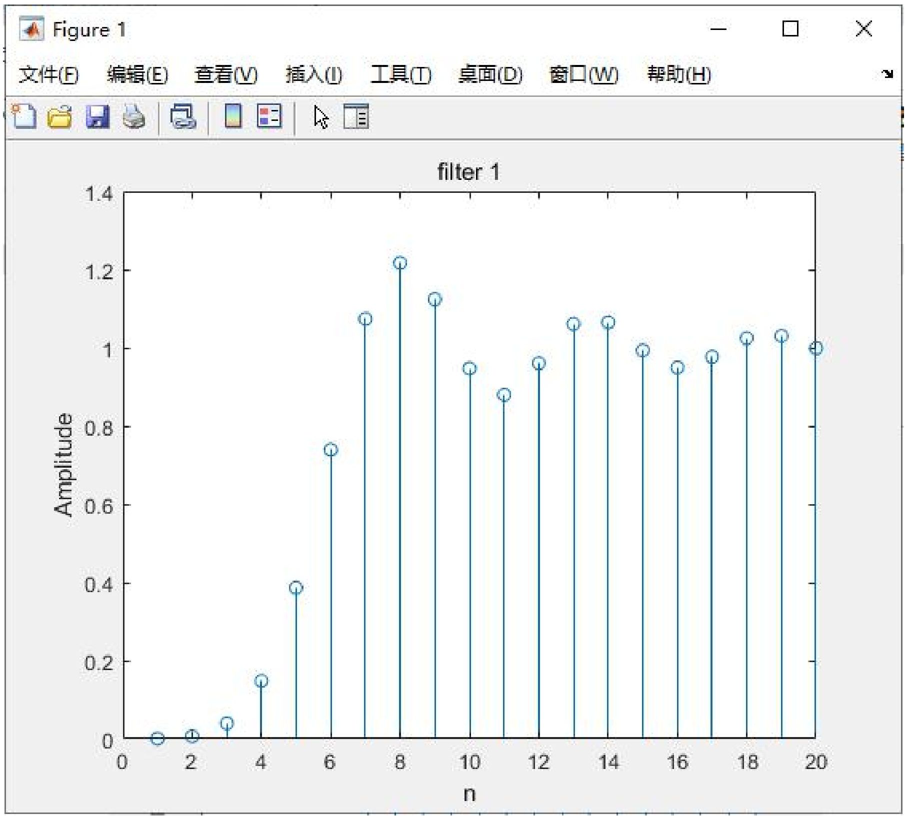

Figure 2.3.1 impulse response of filter 1

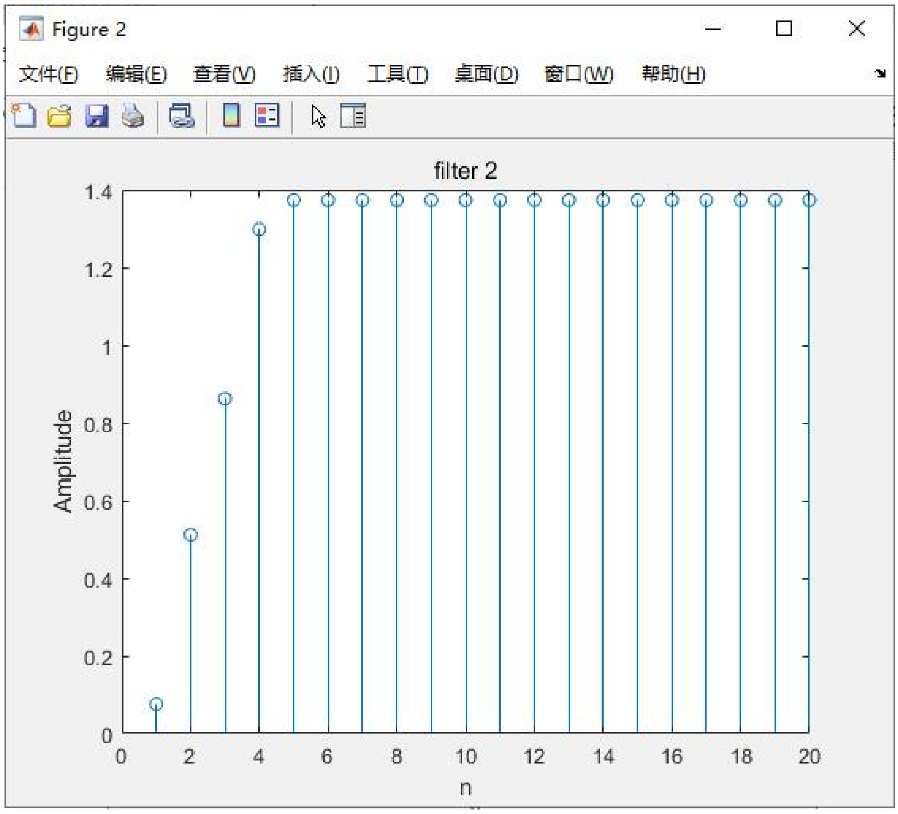

Figure 2.3.2 impulse response of filter 2

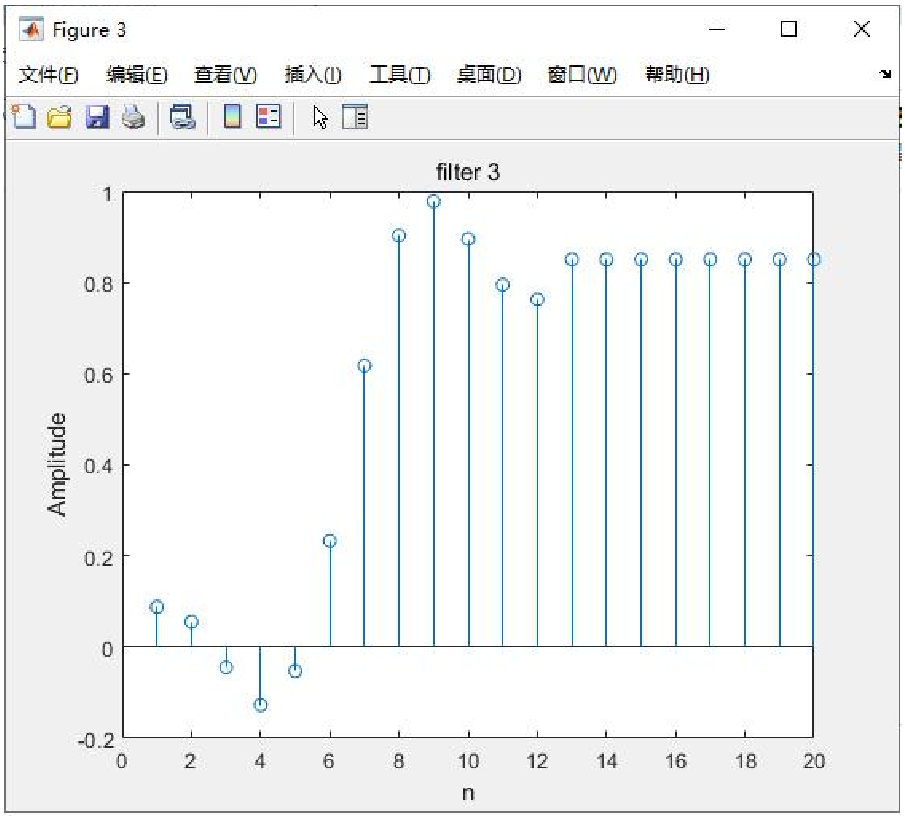

Figure 2.3.3 impulse response of filter 3

It is clearly that filter 1 has the largest overshoot larger than 0.2 while the others not.

MATLAB code:

```
% prob2c.m

clear;
clc;

wc=0.4;
n1=10;n2=4;n3=12;
[b1,a1]=butter(n1,wc);
a2=1;b2=remez(n2,[0 wc-0.04 wc+0.04 1],[1 1 0 0]);
a3=1;b3=remez(n3,[0 wc-0.04 wc+0.04 1],[1 1 0 0]);

n=1:20; % the input signal
x=ones(1,20);
h1=filter(b1,a1,x); % the step responses
```

```
h2=filter(b2,a2,x);
h3=filter(b3,a3,x);

figure;
stem(n,h1);
xlabel('n');
ylabel('Amplitude');
title('filter 1');

figure;
stem(n,h2);
xlabel('n');
ylabel('Amplitude');
title('filter 2');

figure;
stem(n,h3);
xlabel('n');
ylabel('Amplitude');
title('filter 3');
```

(d) Using MATLAB, we can obtain the image as follows:

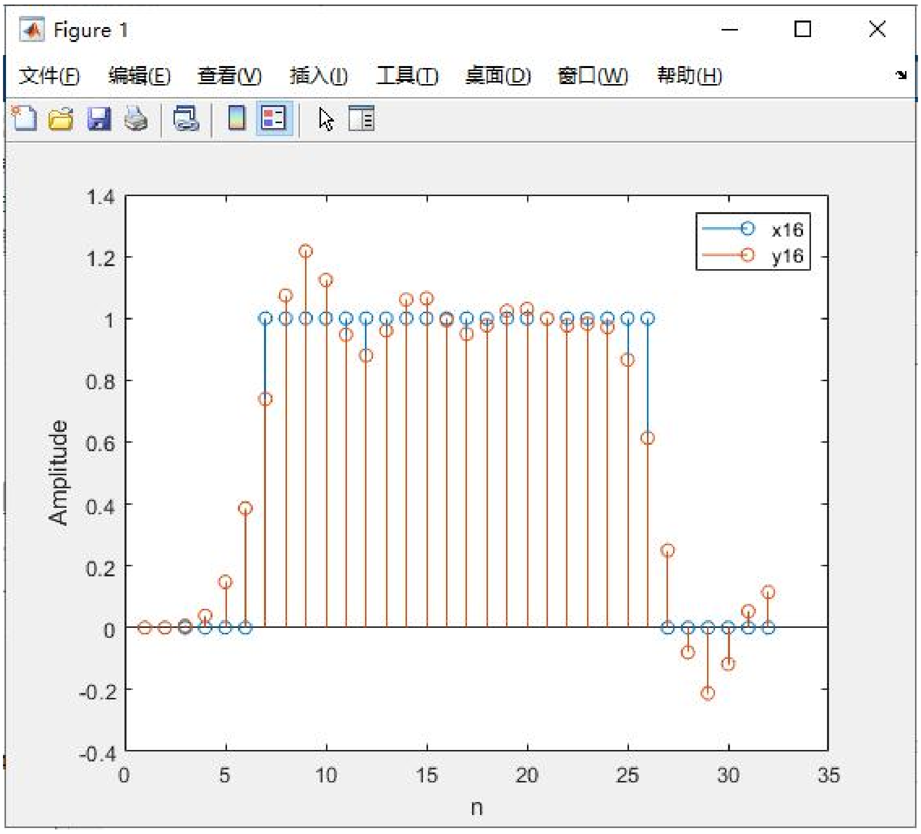

Figure 2.4 image of **x16** and **y16**

And of course the discontinuities in **x16** line up with the “smoothed” discontinuities in **y16**.

MATLAB code:

```
% prob2d.m

clear;
clc;

load plus;

wc=0.4;
n1=10;n2=4;n3=12;
[b1,a1]=butter(n1,wc);
a2=1;b2=remez(n2,[0 wc-0.04 wc+0.04 1],[1 1 0 0]);
a3=1;b3=remez(n3,[0 wc-0.04 wc+0.04 1],[1 1 0 0]);

x16=x(:,16); % the column input
y16_1=filter(b1,a1,[x16;zeros(n1/2,1)]); % filter
                                            response

figure;
stem(x16);
hold on;
stem(y16_1(n1/2+1:end));
xlabel('n');
ylabel('Amplitude');
legend('x16','y16');
```

(e) Similarly, using the method introduced in part (d), we obtained the filter responses of filter 2 and 3 as follows, whose discontinuities are aligned and no delay or advance appears at all:

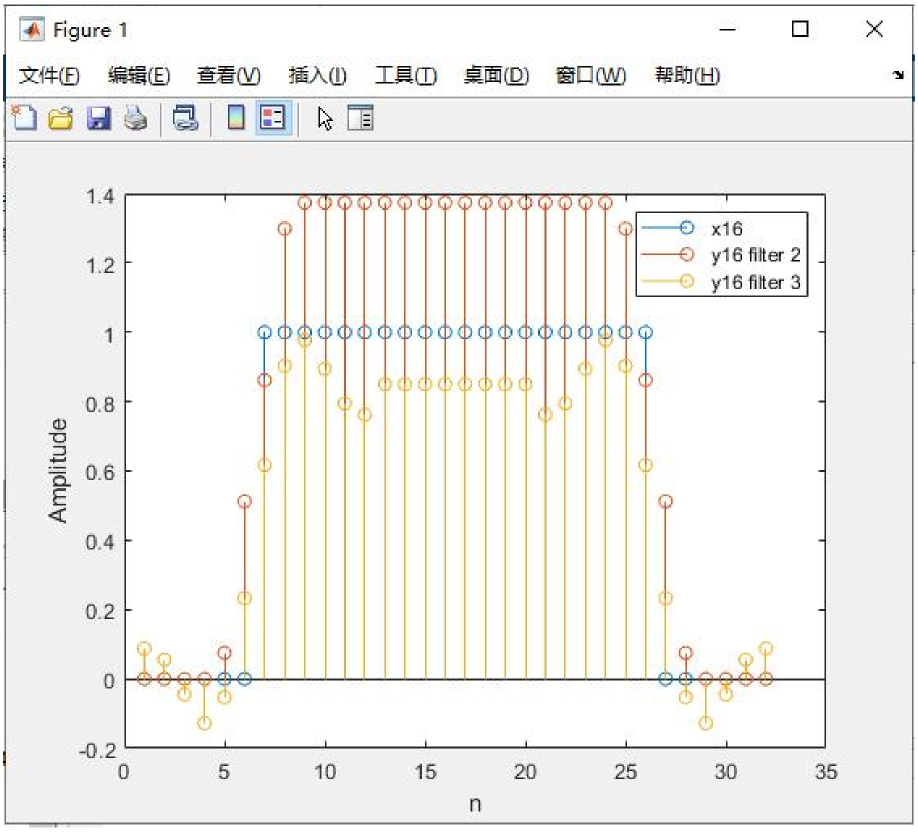

Figure 2.5 filter responses of filter 2 and 3

MATLAB code:

```
% prob2d.m

clear;
clc;

load plus;

wc=0.4;
n1=10;n2=4;n3=12;
[b1,a1]=butter(n1,wc);
a2=1;b2=remez(n2,[0 wc-0.04 wc+0.04 1],[1 1 0 0]);
a3=1;b3=remez(n3,[0 wc-0.04 wc+0.04 1],[1 1 0 0]);

x16=x(:,16);
% filter 2's response
y16_2=filter(b2,a2,[x16;zeros(n2/2,1)]);
% filter 3's response
y16_3=filter(b3,a3,[x16;zeros(n3/2,1)]);

figure;
```

```
stem(x16);
hold on;
stem(y16_2(n2/2+1:end));
hold on;
stem(y16_3(n3/2+1:end));
xlabel('n');
ylabel('Amplitude');
legend('x16','y16 filter 2','y16 filter 3');
```

(f) MATLAB code:

```
% filt2d.m

function y=filt2d(b,a,d,x)
    z=zeros(32,32); % preset memory
    y=zeros(32,32);
    for i=1:32
        xi=x(:,i); % each column
        z1=filter(b,a,[xi;zeros(d,1)]); % filter response
        z(:,i)=z1(d+1:end); % store the result
    end
    for i=1:32
        zi=z(i,:); % each row
        y1=filter(b,a,[zi zeros(1,d)]); % filter response
        y(i,:)=y1(d+1:end); % store the result
    end
```

(g) After filtering, the filtered image for the three filters are respectively as follows, and it is obvious that there is no delay in the image:

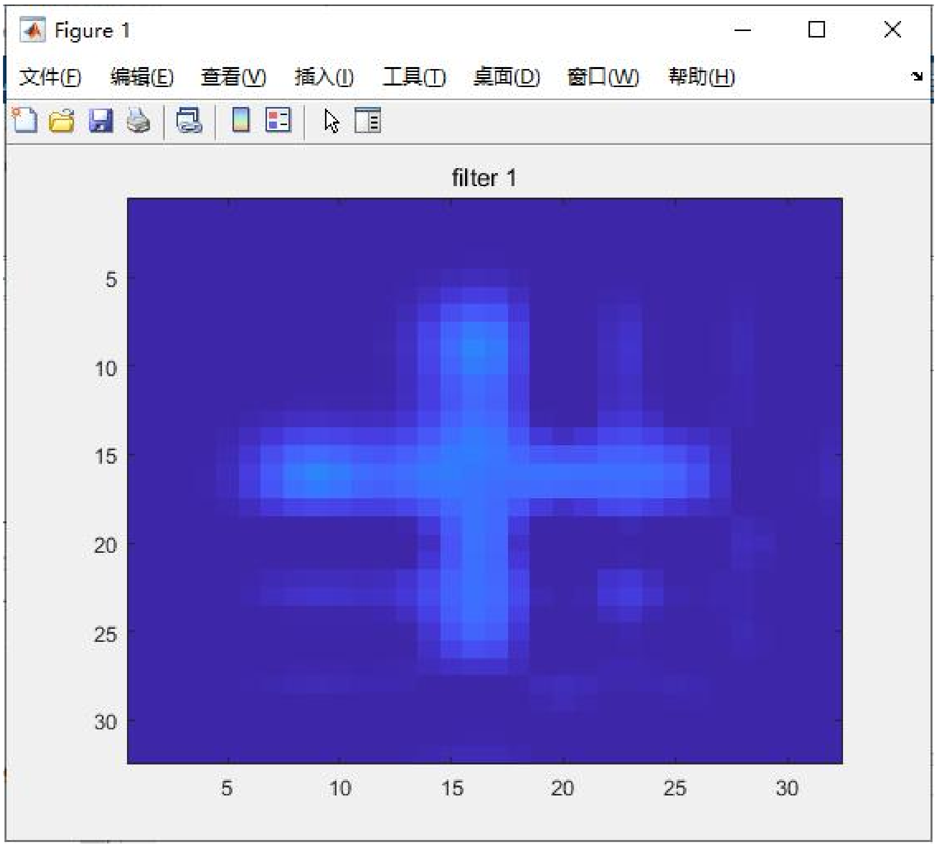

Figure 2.7.1 plus of filter 1

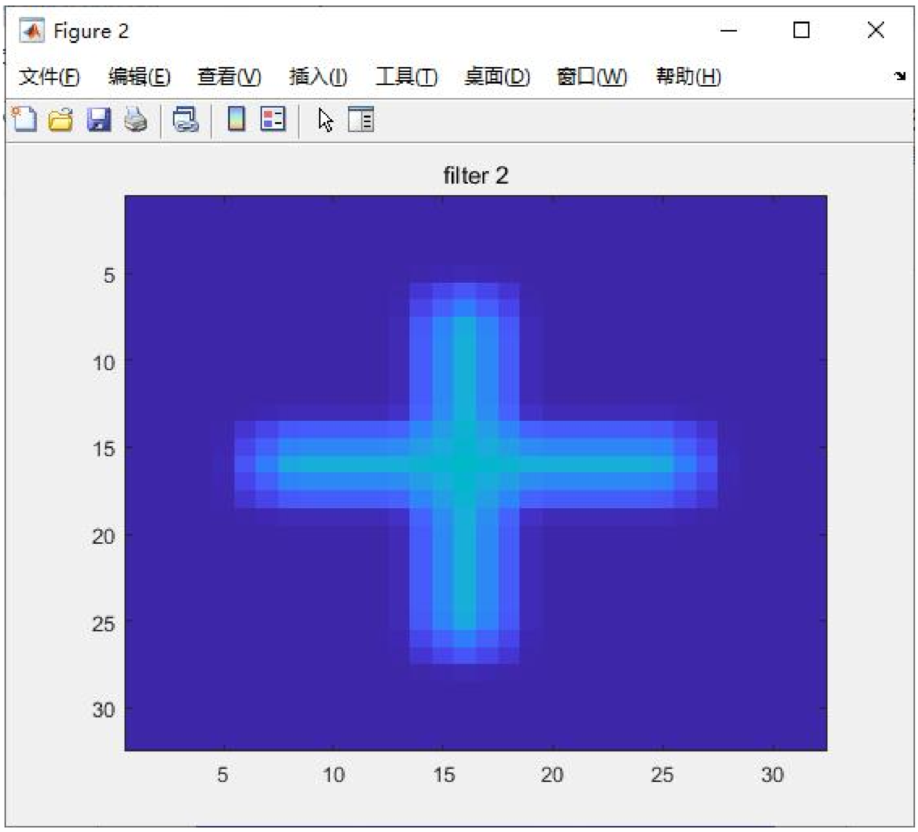

Figure 2.7.2 plus of filter 2

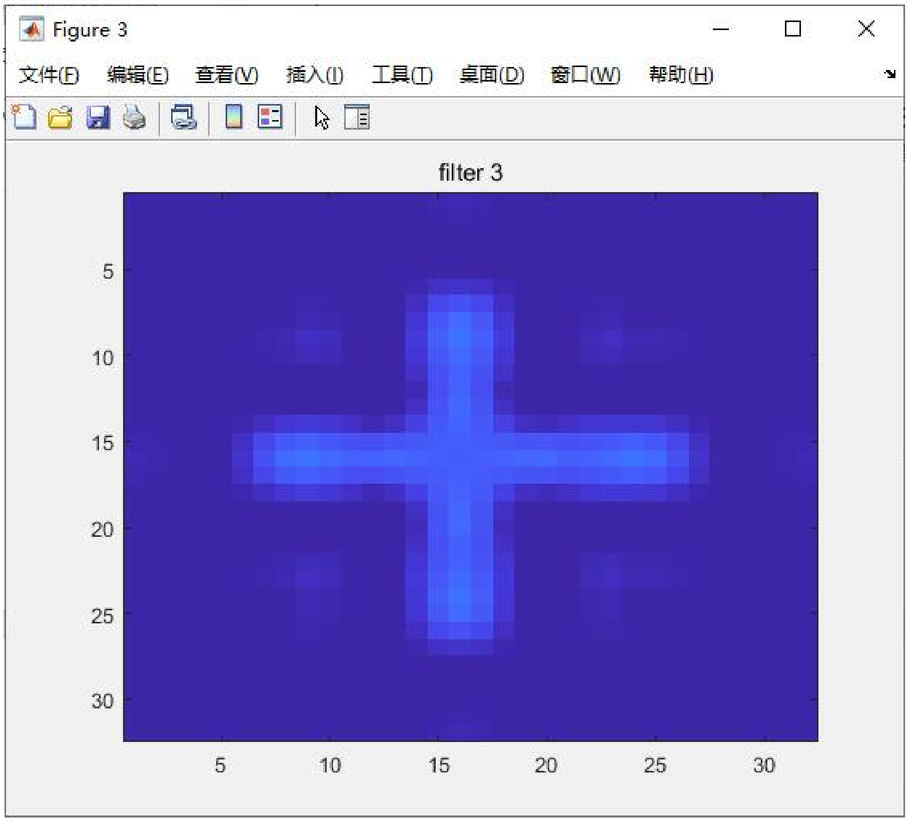

Figure 2.7.3 plus of filter 3

MATLAB code:

```
% prob2g.m

clear;
clc;

load plus;

wc=0.4;
n1=10;n2=4;n3=12;
[b1,a1]=butter(n1,wc);
a2=1;b2=remez(n2,[0 wc-0.04 wc+0.04 1],[1 1 0 0]);
a3=1;b3=remez(n3,[0 wc-0.04 wc+0.04 1],[1 1 0 0]);

y1=filt2d(b1,a1,n1/2,x); % use function filt2d to filter
y2=filt2d(b2,a2,n2/2,x); % filter 2
y3=filt2d(b3,a3,n3/2,x); % filter 3

figure;
image(y1*64);
```

```
title('filter 1');
figure;
image(y2*64);
title('filter 2');
figure;
image(y3*64);
title('filter 3');
```

(h) From the results in (g), filter 1 leads to more distortion, whose image is not symmetry at all. And in part (b) we know that filter 1 doesn't have linear phase, which confirms what the title has told us.

Problem 3

Solutions:

(a) Analytically, $x[n]$ is purely real, because the DTFS coefficients of $x[n]$ is conjugation symmetric, which means $x[n] = x^*[n]$.

(b) Since the DTFS coefficients is also periodic with period $N = 5$, we have,

$$a_0 = 1, a_1 = a_{-4} = 2e^{-j\frac{\pi}{3}}, a_2 = e^{j\frac{\pi}{4}}, a_3 = a_{-2} = e^{-j\frac{\pi}{4}}, a_4 = 2e^{j\frac{\pi}{3}}$$

and the vector:

```
a=[1 2*exp(-1j*pi/3) exp(1j*pi/4) exp(-1j*pi/4) 2*exp(1j*pi/3)];
```

(c) Using the **for** loops in MATLAB, we obtained the plot of $x[n]$'s real part and imaginary part as follows:

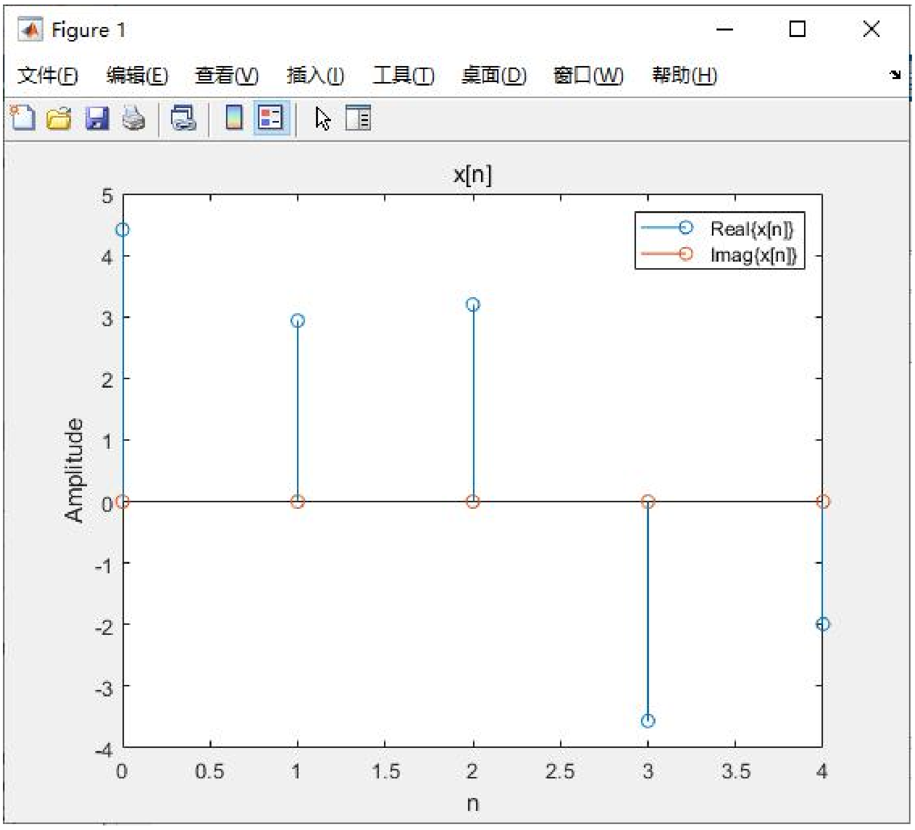

Figure 3.3 image of $x[n]$

Because of the roundoff errors, there exits a very small nonzero imaginary part in $x[n]$, yet it is obvious that the signal is purely real ignoring the errors, which shows that our prediction in part (a) is correct.

MATLAB code:

```
% prob3c.m

clear;
clc;

a=[1 2*exp(-1j*pi/3) exp(1j*pi/4) exp(-1j*pi/4)  ...
                                      2*exp(1j*pi/3)];
x=zeros(1,5);
nx=0:4;
for n=1:5
```

```
    for k=1:5
        % compute x[n]
        x(n)=x(n)+a(k)*exp(1j*(k-1)*2*pi/5*(n-1));
    end
end
xr=real(x); % real and imaginary part
xi=imag(x);

stem(nx,xr);
hold on;
stem(nx,xi);
xlabel('n');
ylabel('Amplitude');
title('x[n]');
legend('Real\{x[n]\}','Imag\{x[n]\}');
```

(d) Using MATLAB, the periodic signals over the range $0 \le n \le 63$ are as:

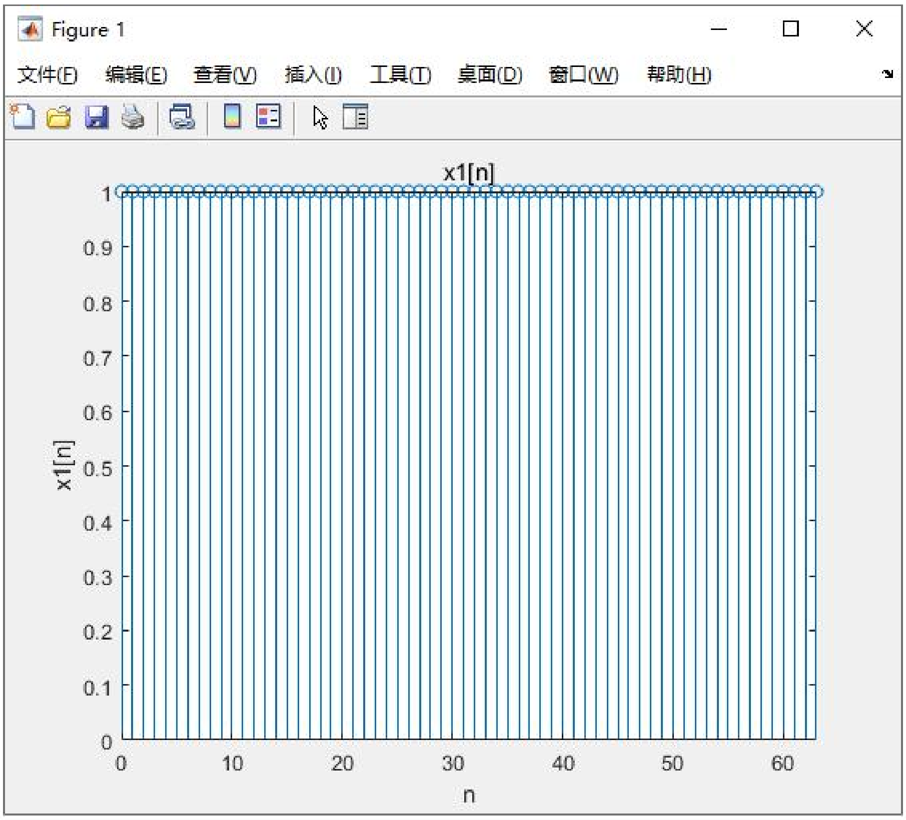

Figure 3.4.1 image of $x_1[n]$

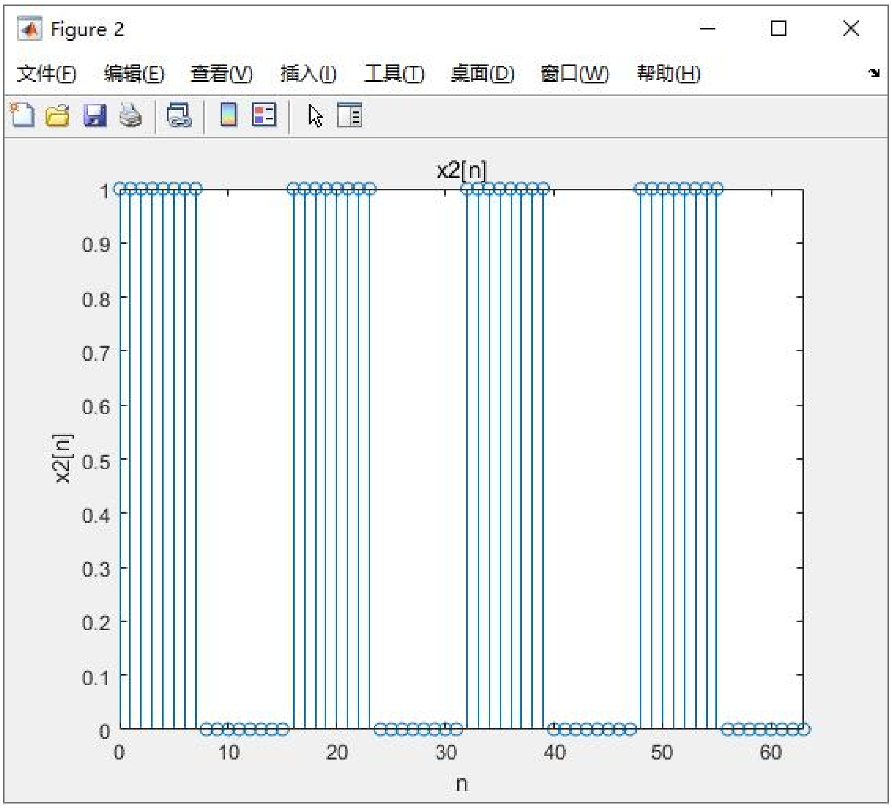

Figure 3.4.2 image of $x_2[n]$

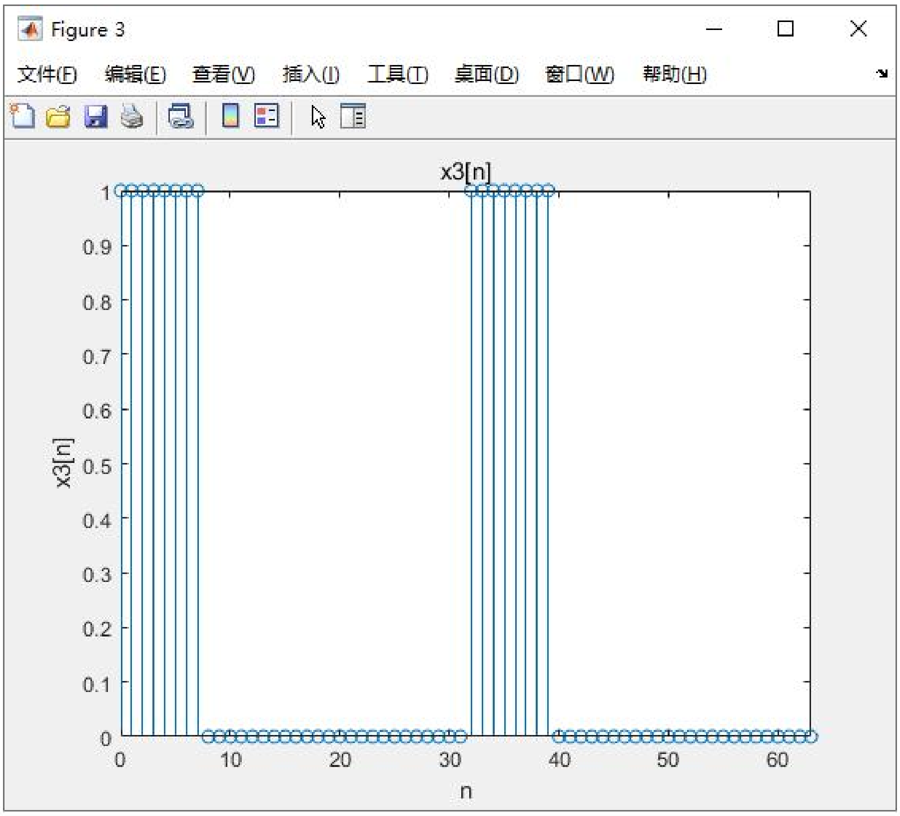

Figure 3.4.3 image of $x_3[n]$

MATLAB code:

```
% prob3d.m

clear;
clc;

x1=ones(1,8); % each periodic signal
x2=[ones(1,8),zeros(1,8)];
x3=[ones(1,8),zeros(1,24)];
n=0:63;

x1_0=[x1 x1 x1 x1 x1 x1 x1 x1];
x2_0=[x2 x2 x2 x2];
x3_0=[x3 x3]; % repeat the vectors

figure;
stem(n,x1_0);
xlabel('n');
ylabel('x1[n]');
title('x1[n]');
axis([0 63 -inf inf]);

figure;
stem(n,x2_0);
xlabel('n');
ylabel('x2[n]');
title('x2[n]');
axis([0 63 -inf inf]);

figure;
stem(n,x3_0);
xlabel('n');
ylabel('x3[n]');
title('x3[n]');
axis([0 63 -inf inf]);
```

(e) As is known to us, the DC component of a signal is the average of its one period, so

$$a1(1) = 1,$$

$$a2(1) = \frac{8}{16} = \frac{1}{2},$$

$$a3(1) = \frac{8}{32} = \frac{1}{4}$$

While the plots of the magnitude of each of the DTFS coefficients are as follows, from which we can see that $a1(1) = 1$, $a2(1) = \frac{1}{2}$ and $a3(1) = \frac{1}{4}$, matching the results we have predicted above:

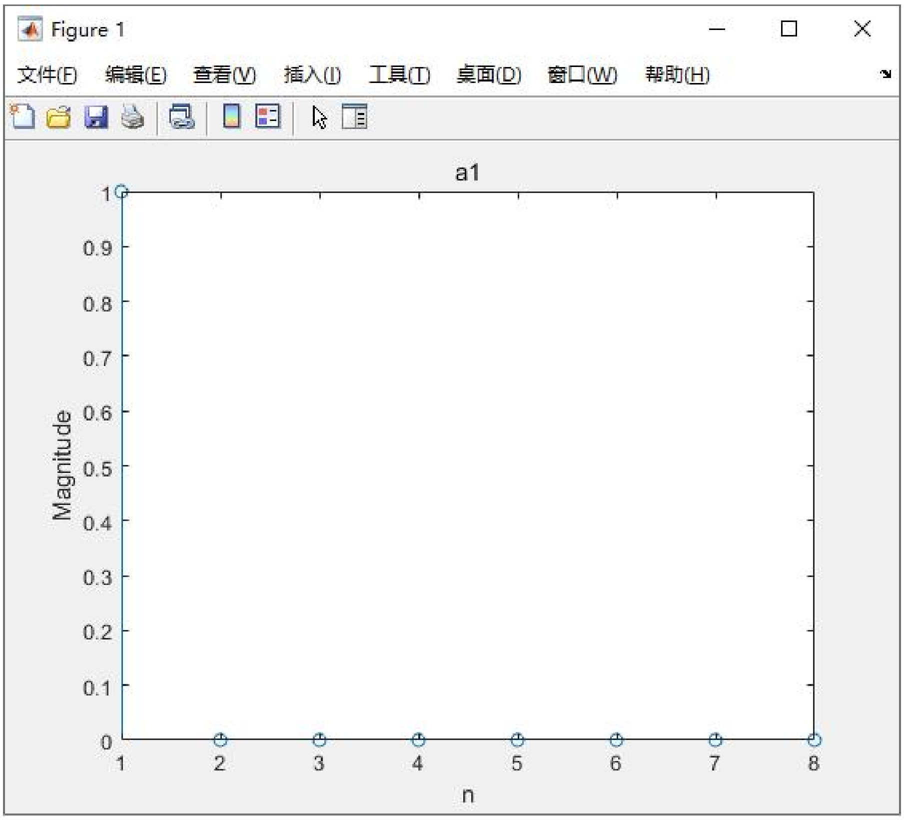

Figure 3.5.1 magnitude of a1

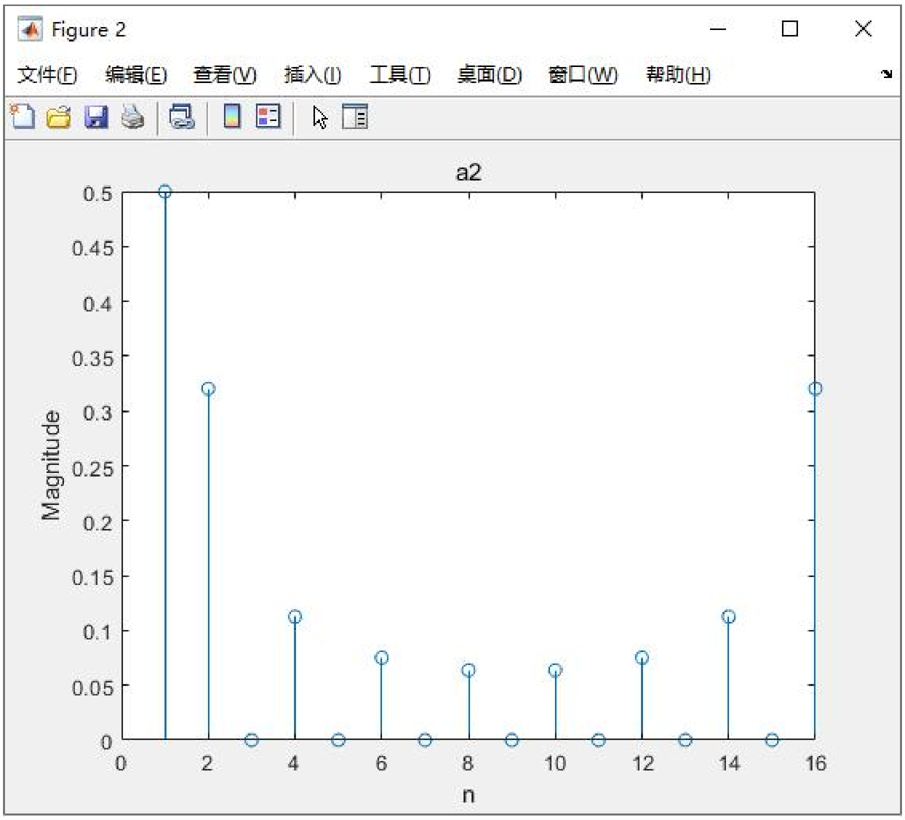

Figure 3.5.2 magnitude of a2

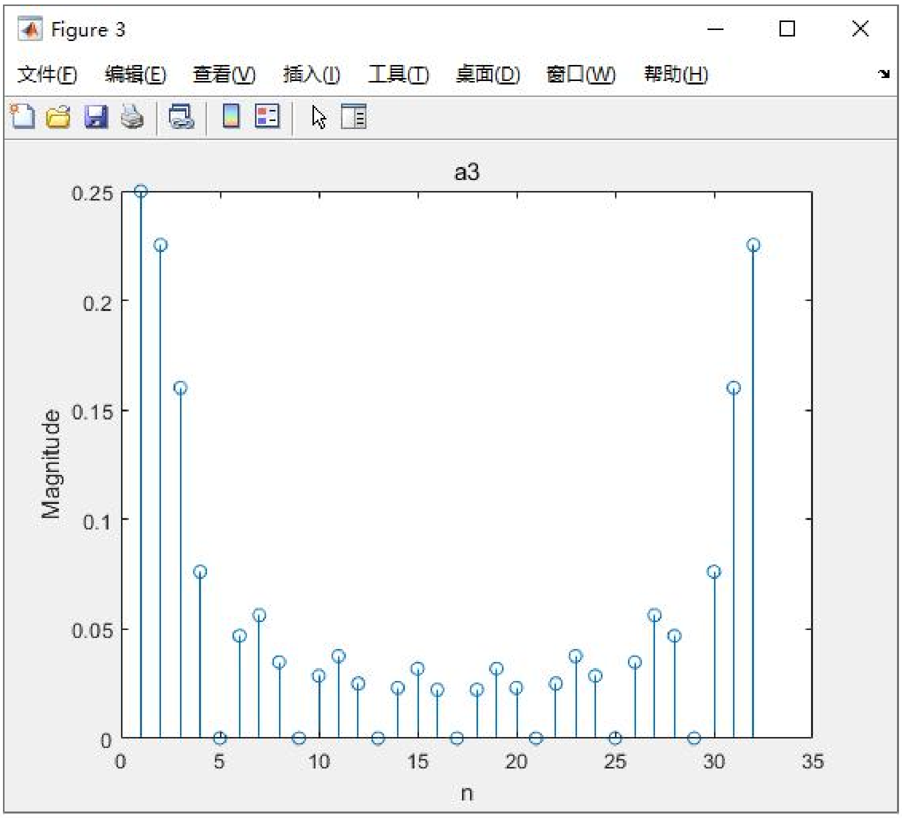

Figure 3.5.3 magnitude of a3

MATLAB code:

```
% prob3e.m

clear;
clc;

x1=ones(1,8);
x2=[ones(1,8),zeros(1,8)];
x3=[ones(1,8),zeros(1,24)];

% use fft to compute coefficients
a1=1/8*fft(x1);
a2=1/16*fft(x2);
a3=1/32*fft(x3);

a1_s=abs(a1); % the magnitude
a2_s=abs(a2);
a3_s=abs(a3);

figure;
stem(a1_s);
xlabel('n');
ylabel('Magnitude');
title('a1');

figure;
stem(a2_s);
xlabel('n');
ylabel('Magnitude');
title('a2');

figure;
stem(a3_s);
xlabel('n');
ylabel('Magnitude');
title('a3');
```

(f) Since the DTFS coefficients are conjugation symmetric, we have, for $k < 0$, $a_k = a^*_{-k}$. On the other hand, because the coefficients are periodic, we can also obtain the elements of a corresponding to the negative k by $a_k = a_{k+32}$, from which we know that a_{-15}, $a_{-14}, \ldots\ldots, a_{-1}$ equals to $a_{17}, a_{18}, \ldots\ldots, a_{31}$ respectively (however, in MATLAB we shall add the latter index 1 because it starts with 1). The MATLAB code can be seen in part (h).

(g) According to the title,

$$x_{3\_all}[n] = \sum_{k=-15}^{16} a_k e^{jk\frac{2\pi}{32}n}$$

then its conjugation is

$$x_{3\_all}^*[n] = \sum_{k=-15}^{16} a_k^* e^{-jk\frac{2\pi}{32}n}$$

since $x[n]$ is purely real, the FS coefficients are conjugation symmetric. Substituting k with $-k$ in the equation above, we get,

$$x_{3\_all}^*[n] = \sum_{k=-16}^{15} a_{-k}^* e^{jk\frac{2\pi}{32}n} = \sum_{k=-16}^{15} a_k e^{jk\frac{2\pi}{32}n}$$

Furthermore, the coefficients are periodic with period $N = 32$, so $a_{-16} = a_{16}$ while

$$e^{j(-16)\frac{2\pi}{32}n} = e^{-jn\pi} = e^{jn\pi} = e^{j16\frac{2\pi}{32}n}$$

then

$$x_{3\_all}^*[n] = \sum_{k=-16}^{15} a_k e^{jk\frac{2\pi}{32}n} = \sum_{k=-15}^{16} a_k e^{jk\frac{2\pi}{32}n} = x_{3\_all}[n]$$

which means that $x_{3\_all}[n]$'s conjugation equals to itself, so $x_{3\_all}[n]$

must be real signal.

(h) Using what we have discussed in part (f), the plots of the signals as well as $x_3[n]$ is drawn as follows:

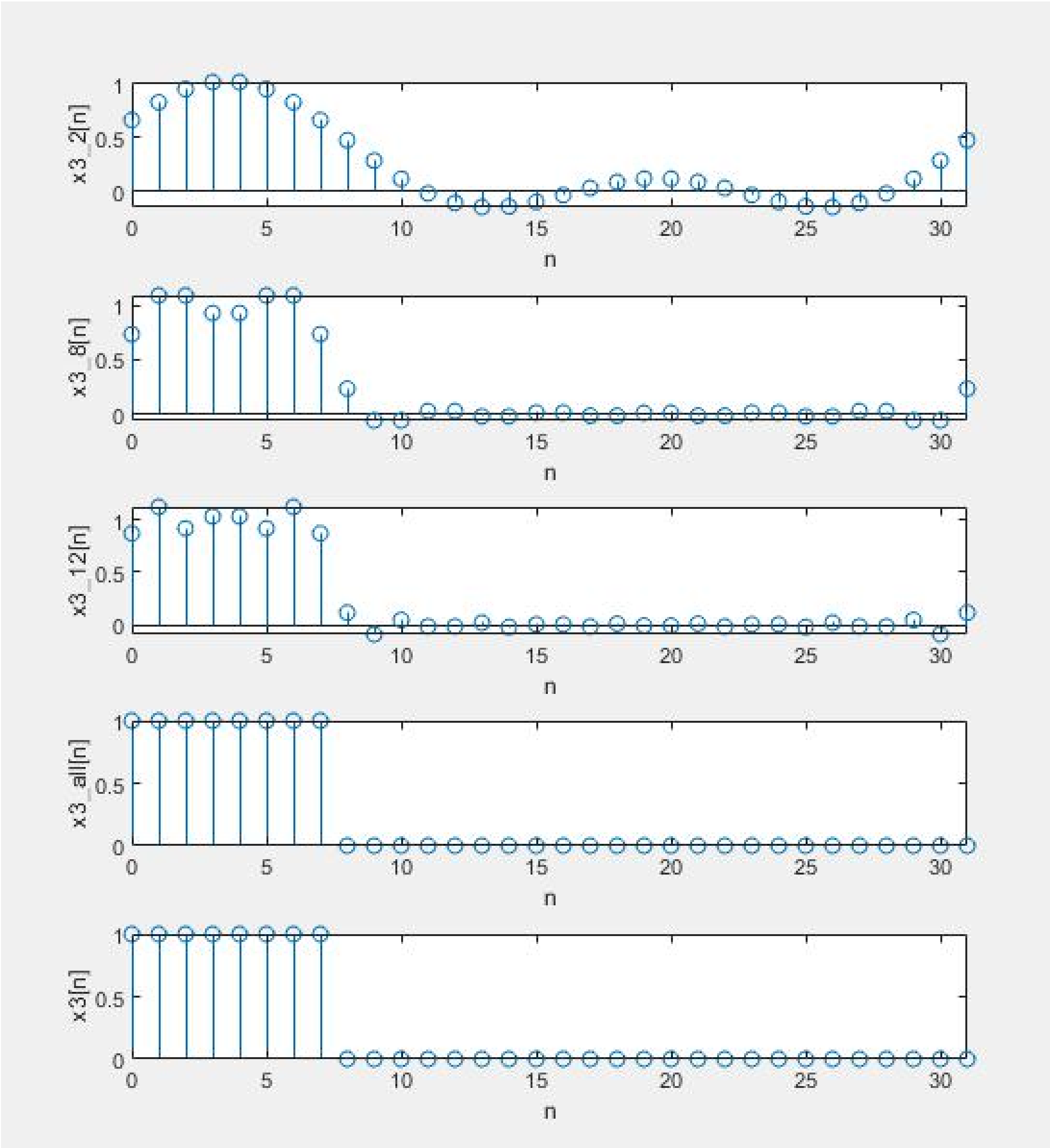

Figure 3.8 converging to $x_3[n]$

As we can see, as more of the DTFS coefficients are included in the sum, the signals converge to $x_3[n]$ (visually looks more like $x_3[n]$), and when all the coefficients are included, signal $x_{3\_all}[n]$ becomes exactly the original one $x_3[n]$. Through the progress of synthesis, there is no Gibb's phenomenon being displayed because the DTFS is expressed by finite term's sum and exists no convergence problem.

MATLAB code:

```
% prob3f&h.m

clear;
clc;

x3=[ones(1,8),zeros(1,24)];
a3=1/32*fft(x3); % the FTFS coefficients
nx=0:31;

x3_2=zeros(1,32); % preset the memory
x3_8=zeros(1,32);
x3_12=zeros(1,32);
x3_all=zeros(1,32);

for n=1:32 % x3_2
    for k=31:32 % the negative parts
       x3_2(n)=x3_2(n)+a3(k)*exp(1j*(k-33)*2*pi/32*n);
    end
    for k=1:3 % the positive parts
        x3_2(n)=x3_2(n)+a3(k)*exp(1j*(k-1)*2*pi/32*n);
    end
end
x3_2=circshift(x3_2,1);
x3_2r=real(x3_2); % ignore the very small imainary part

for n=1:32 % x3_8
    for k=25:32 % the negative parts
       x3_8(n)=x3_8(n)+a3(k)*exp(1j*(k-33)*2*pi/32*n);
    end
    for k=1:9 % the positive parts
        x3_8(n)=x3_8(n)+a3(k)*exp(1j*(k-1)*2*pi/32*n);
    end
end
x3_8=circshift(x3_8,1);
```

```
x3_8r=real(x3_8);

for n=1:32 % x3_12
    for k=21:32 % the negative parts
      x3_12(n)=x3_12(n)+a3(k)*exp(1j*(k-33)*2*pi/32*n);
    end
    for k=1:13 % the positive parts
        x3_12(n)=x3_12(n)+a3(k)*exp(1j*(k-1)*2*pi/32*n);
    end
end
x3_12=circshift(x3_12,1);
x3_12r=real(x3_12);

for n=1:32 % x3_all
    for k=1:32 % the positive and negative parts
     x3_all(n)=x3_all(n)+a3(k)*exp(1j*(k-1)*2*pi/32*n);
    end
end
x3_all=circshift(x3_all,1);
x3_allr=real(x3_all);

% plots
figure;
subplot(5,1,1); % x3_2[n]
stem(nx,x3_2r);
xlabel('n');
ylabel('x3_2[n]','Interpreter','none');
axis([0,31,-inf,inf]);

subplot(5,1,2); % x3_8[n]
stem(nx,x3_8r);
xlabel('n');
ylabel('x3_8[n]','Interpreter','none');
axis([0,31,-inf,inf]);

subplot(5,1,3); % x3_12[n]
stem(nx,x3_12r);
xlabel('n');
ylabel('x3_12[n]','Interpreter','none');
axis([0,31,-inf,inf]);

subplot(5,1,4); % x3_all[n]
stem(nx,x3_allr);
xlabel('n');
```

```
ylabel('x3_all[n]','Interpreter','none');
axis([0,31,-inf,inf]);

subplot(5,1,5); % x3[n]
stem(nx,x3);
xlabel('n');
ylabel('x3[n]');
axis([0,31,-inf,inf]);
```