Signals and systems

Lab 02

3190102060　黄嘉欣

Problem 1

Solutions:

(a) 1. For System 1:

By using MATLAB, the figure of the four functions is as follows:

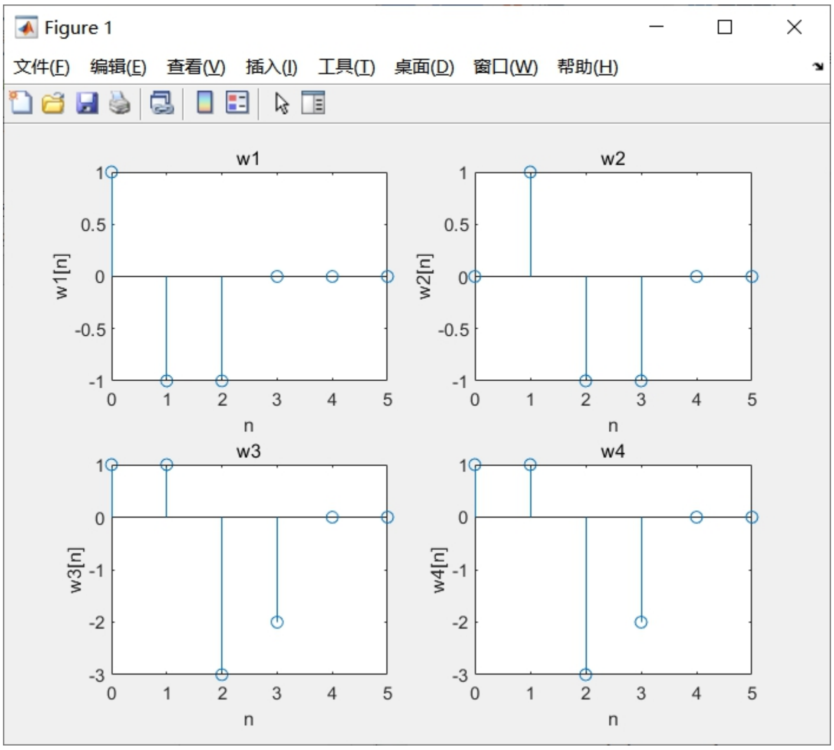

Figure 1.1　Outputs of System 1

And the MATLAB code is:

```
% prob1a_system1.m
clear;
clc;

% the input signals
x1=[1 0 0 0 0 0];
x2=[0 1 0 0 0 0];
x3=[1 2 0 0 0 0];
```

```
% the coefficients of the difference equation
a=[1 0 0];
b=[1 -1 -1];

% compute the output signals by using filter command
w1=filter(b,a,x1);
w2=filter(b,a,x2);
w3=filter(b,a,x3);
w4=w1+2*w2;

% figure of w1[n]
subplot(2,2,1);
stem(0:5,w1);
title('w1');
xlabel('n');
ylabel('w1[n]');
axis([0 5 -inf inf]);

% figure of w2[n]
subplot(2,2,2);
stem(0:5,w2);
title('w2');
xlabel('n');
ylabel('w2[n]');
axis([0 5 -inf inf]);

% figure of w3[n]
subplot(2,2,3);
stem(0:5,w3);
title('w3');
xlabel('n');
ylabel('w3[n]');
axis([0 5 -inf inf]);

% figure of w4[n]
subplot(2,2,4);
stem(0:5,w4);
title('w4');
xlabel('n');
ylabel('w4[n]');
axis([0 5 -inf inf]); % Limit the range of abscissa
```

2. For System 2:

The figure of the four functions is:

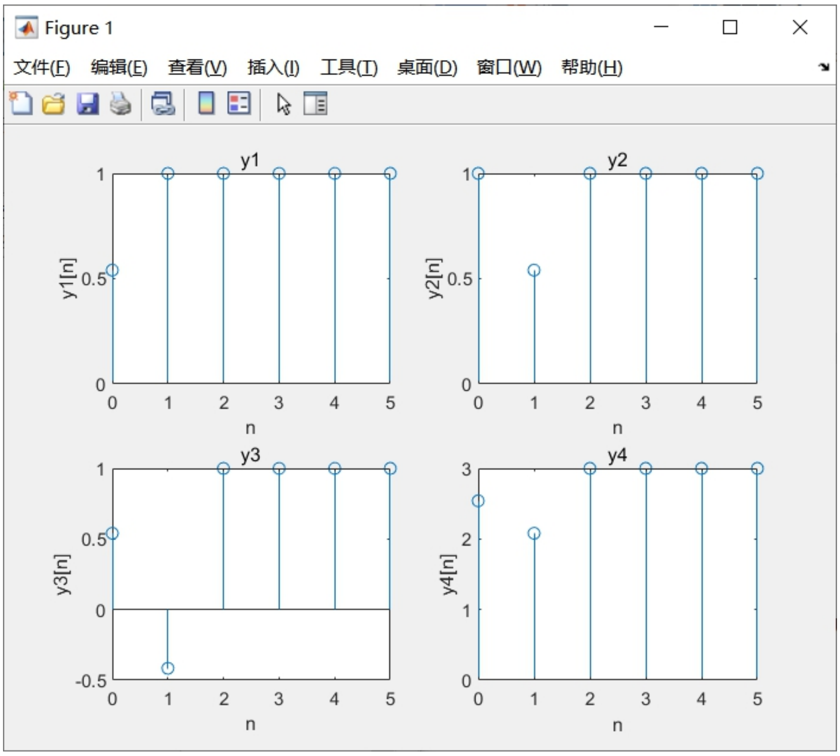

Figure 1.2 Outputs of System 2

And the MATLAB code is as follows:

```
% prob1a_system2.m
clear;
clc;

% the input signals
x1=[1 0 0 0 0 0];
x2=[0 1 0 0 0 0];
x3=[1 2 0 0 0 0];

% the output signals
y1=cos(x1);
y2=cos(x2);
y3=cos(x3);
y4=y1+2*y2;

% figure of y1[n]
subplot(2,2,1);
stem(0:5,y1);
```

```
title('y1');
xlabel('n');
ylabel('y1[n]');
axis([0 5 0 1]);

% figure of y2[n]
subplot(2,2,2);
stem(0:5,y2);
title('y2');
xlabel('n');
ylabel('y2[n]');
axis([0 5 0 1]);

% figure of y3[n]
subplot(2,2,3);
stem(0:5,y3);
title('y3');
xlabel('n');
ylabel('y3[n]');
axis([0 5 -0.5 1]);

% figure of y4[n]
subplot(2,2,4);
stem(0:5,y4);
title('y4');
xlabel('n');
ylabel('y4[n]');
axis([0 5 0 3]); % Limit the range of abscissa
```

3. For System 3:

   The outputs corresponding to the input signals are as follows:

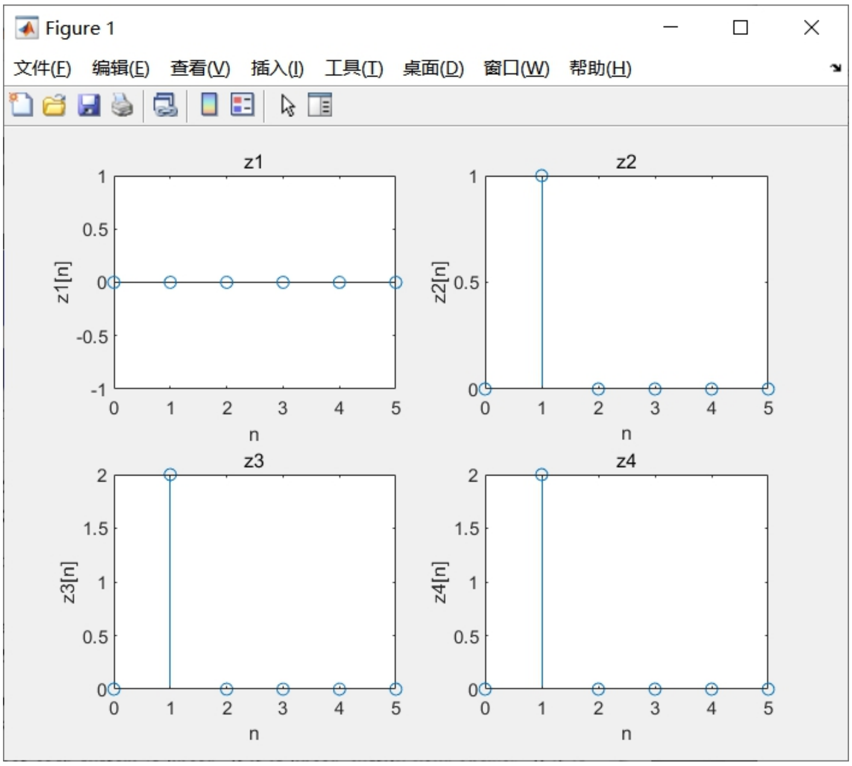

Figure 1.3 Outputs of System 3

And the MATLAB code is:

```
% prob1a_system3.m
clear;
clc;

n=0:1:5;

% the input signals
x1=[1 0 0 0 0 0];
x2=[0 1 0 0 0 0];
x3=[1 2 0 0 0 0];

% the output signals
z1=n.*x1;
z2=n.*x2;
z3=n.*x3;
z4=z1+2*z2;

% figure of z1[n]
subplot(2,2,1);
stem(n,z1);
title('z1');
```

```
xlabel('n');
ylabel('z1[n]');
axis([0 5 -1 1]);

% figure of z2[n]
subplot(2,2,2);
stem(n,z2);
title('z2');
xlabel('n');
ylabel('z2[n]');
axis([0 5 -inf inf]);

% figure of z3[n]
subplot(2,2,3);
stem(n,z3);
title('z3');
xlabel('n');
ylabel('z3[n]');
axis([0 5 -inf inf]);

% figure of z4[n]
subplot(2,2,4);
stem(n,z4);
title('z4');
xlabel('n');
ylabel('z4[n]');
axis([0 5 -inf inf]); % Limit the range of abscissa
```

(b) 1. System 1 is linear.

Proof:

Assume that $x_1[n]$ and $x_2[n]$ are arbitrary input signals for System 1 and $x_3[n]$ is an arbitrary linear combination of $x_1[n]$ and $x_2[n]$, that is,

$$x_3[n] = ax_1[n] + bx_2[n]$$

where a and b are constants.

So the corresponding output signals for the three signals are respectively,

$$w_1[n] = x_1[n] - x_1[n-1] - x_1[n-2],$$

$$w_2[n] = x_2[n] - x_2[n-1] - x_2[n-2],$$

and

$$w_3[n] = x_3[n] - x_3[n-1] - x_3[n-2]$$

Then we have,

$$\begin{aligned} w_3[n] &= ax_1[n] + bx_2[n] - ax_1[n-1] \\ &\quad - bx_2[n-1] - ax_1[n-2] - bx_2[n-2] \\ &= a(x_1[n] - x_1[n-1] - x_1[n-2]) \\ &\quad + b(x_2[n] - x_2[n-1] - x_2[n-2]) \\ &= aw_1[n] + bw_2[n] \end{aligned}$$

which shows the linearity of this system.

2. System 2 is not linear.

   According to the Figure 1.2, it is obvious that $y_3[n]$ is different with $y_4[n]$, so a linear combination of the inputs cannot obtain a linear combination of the corresponding outputs, and the linearity does not holds.

3. System 3 is linear.

   Proof:

   Assume that $x_1[n]$ and $x_2[n]$ are arbitrary input signals for System 1 and $x_3[n]$ is an arbitrary linear combination of $x_1[n]$ and $x_2[n]$, that is,

$$x_3[n] = ax_1[n] + bx_2[n]$$

where a and b are constants.

So the corresponding output signals for the three signals are respectively,

$$z_1[n] = nx_1[n],$$

$$z_2[n] = nx_2[n],$$

and

$$z_3[n] = nx_3[n]$$

Then we have,

$$\begin{aligned} z_3[n] &= n(ax_1[n] + bx_2[n]) \\ &= anx_1[n] + bnx_2[n] \\ &= az_1[n] + bz_2[n] \end{aligned}$$

which shows the linearity of this system.

(c) 1. System 1 is time-invariant.

Proof:

Assume that $x_1[n]$ is an arbitrary input signal and $x_2[n]$ is obtained by shifting $x_1[n]$, that is,

$$x_2[n] = x_1[n - n_0]$$

So the output signals corresponding to the input signals are, respectively,

$$y_1[n] = x_1[n] - x_1[n-1] - x_1[n-2],$$

$$y_2[n] = x_2[n] - x_2[n-1] - x_2[n-2],$$

then we have,

$$y_2[n] = x_1[n-n_0] - x_1[n-1-n_0] - x_1[n-2-n_0]$$

Furthermore,

$$y_1[n-n_0] = x_1[n-n_0] - x_1[n-1-n_0] - x_1[n-2-n_0]$$

so $$y_2[n] = y_1[n-n_0]$$

which means this system is time-invariant.

2. System 2 is time-invariant.

   Proof:

   Assume that $x_1[n]$ is an arbitrary input signal and $x_2[n]$ is obtained by shifting $x_1[n]$, that is,

$$x_2[n] = x_1[n-n_0]$$

   So the output signals corresponding to the input signals are, respectively,

$$y_1[n] = cos(x_1[n]),$$

$$y_2[n] = cos(x_2[n]),$$

   then we have,

$$y_2[n] = cos(x_1[n-n_0])$$

   On the other hand,

$$y_1[n-n_0] = cos(x_1[n-n_0]) = y_2[n]$$

   so the system is time-invariant.

3. System is not time-invariant.

According to the Figure 1.3, $z_2[n]$ cannot be obtained by shifting $z_1[n]$ one unit to the right. So the output signal corresponding to the shifted input signal is not equal to the output that obtained by shifting the original output signal the same distance, which means the system is time-varying.

Problem 2

Solutions:

(a) When the input is an unit step signal, that is, $x(t) = u(t)$, we have

$$\frac{ds(t)}{dt} + 3s(t) = 1$$

for $t > 0$.

It is obvious that an unique solution for this equation is $s(t) = \frac{1}{3}$.

Furthermore, the characteristic equation of the ODE is

$$\lambda + 3 = 0$$

and $\lambda =- 3$,

so the general solution is $s(t) = ce^{-3t}$

and the solution for the ODE is

$$s(t) = ce^{-3t} + \frac{1}{3}$$

Since the system is causal and continuous through 0, the output must satisfy the boundary condition:

$$s(0) = c + \frac{1}{3} = 0$$

therefore,

$$c = -\frac{1}{3}$$

Then according to the causality of the system, we obtain the unit step response as:

$$s(t) = (-\frac{1}{3}e^{-3t} + \frac{1}{3})u(t)$$

And by deriving s with respect to t, we can get the unit impulse response of the system as:

$$h(t) = e^{-3t}u(t)$$

By using MATLAB, we can get the figures of $s(t)$ and $h(t)$ versus t in the interval $[-1, 4]$ as follows:

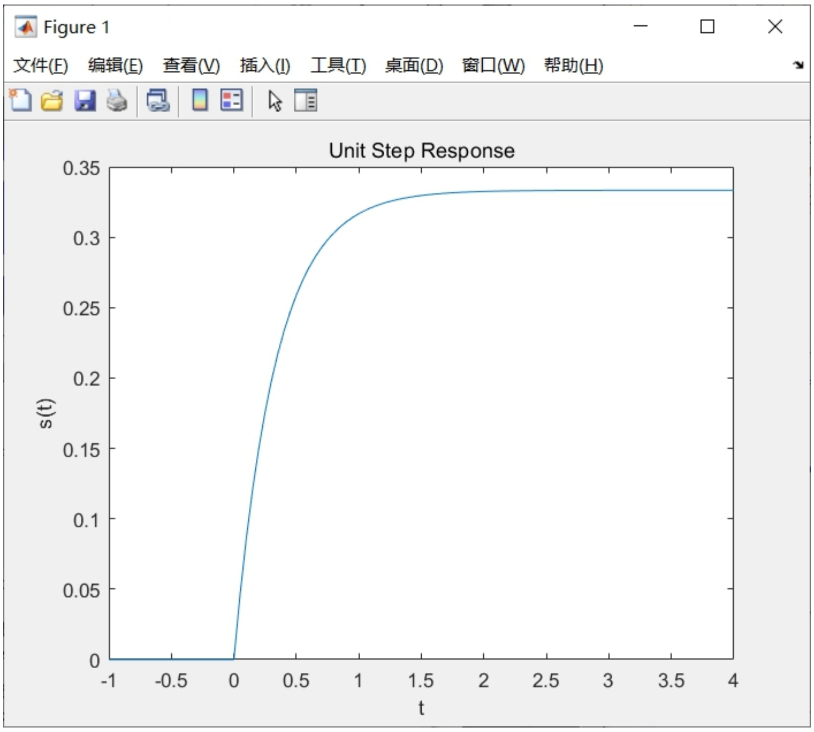

Figure 2.1.1 Image of s(t)

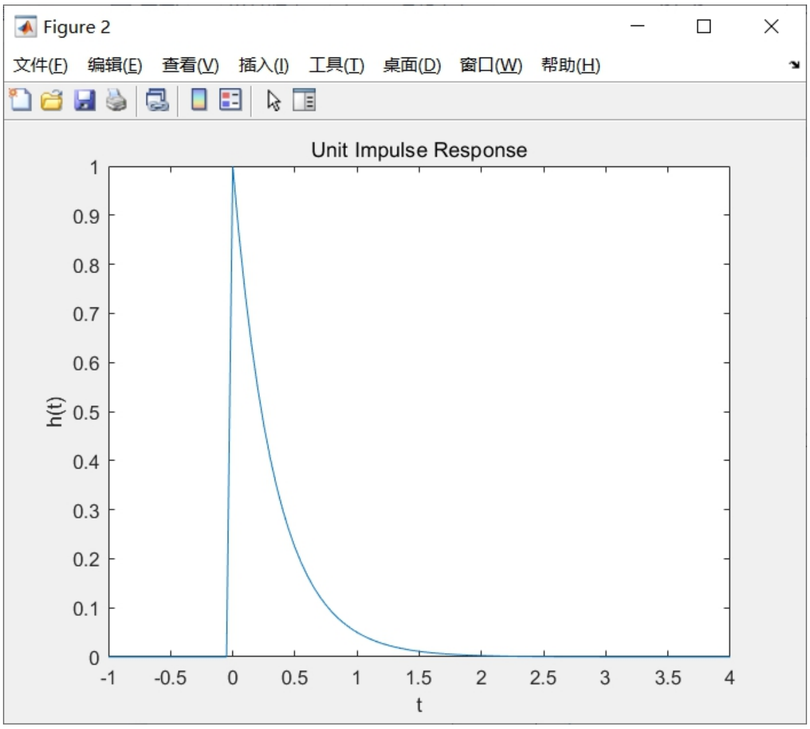

Figure 2.1.2 Image of h(t)

It is known that there is a mutation of $h(t)$ at the point of $t = 0$ from 0 to 1. However, in the program, we take a time point every 0.05 in the interval, so in Figure 2.1.2 there is a sloping line instead of a vertical one at $t = 0$. This can be reduced by narrowing the time interval of the points.

MATLAB code:

```
% prob2a.m
clear;
clc;

% time vector
t=-1:0.05:4;

% the unit step response
s=(-1/3*exp(-3*t)+1/3).*(t>=0);
```

```
% the unit impulse response
h=exp(-3*t).*(t>=0);

% figure of s(t)
figure;
plot(t,s);
title('Unit Step Response');
xlabel('t');
ylabel('s(t)');

% figure of h(t)
figure;
plot(t,h);
title('Unit Impulse Response');
xlabel('t');
ylabel('h(t)');
```

(b) By using MATLAB, we can obtain the step response and impulse response of the system in the interval $[0, 4]$ as follows:

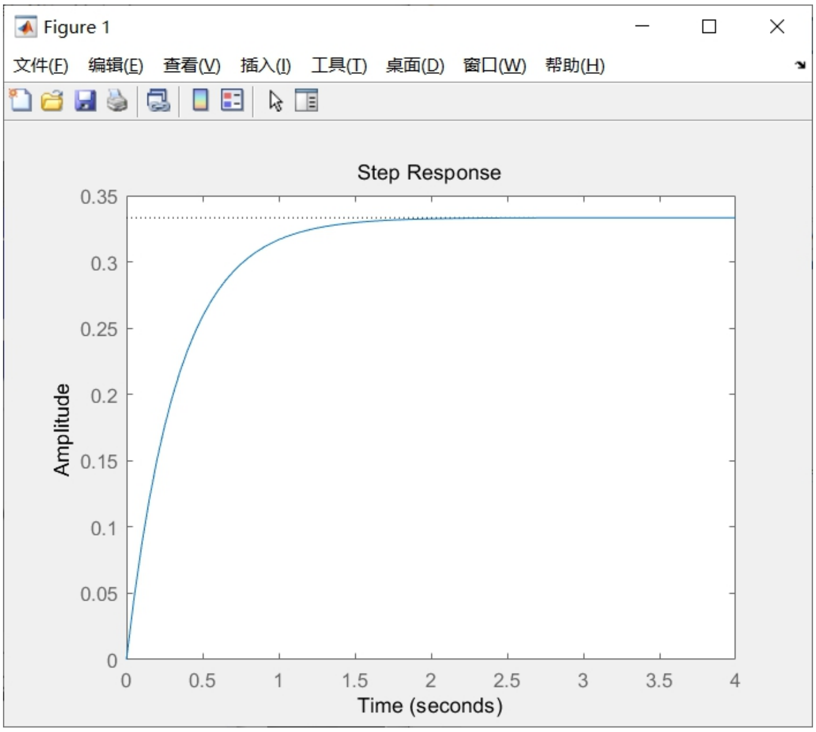

Figure 2.2.1 Image of the Step Response

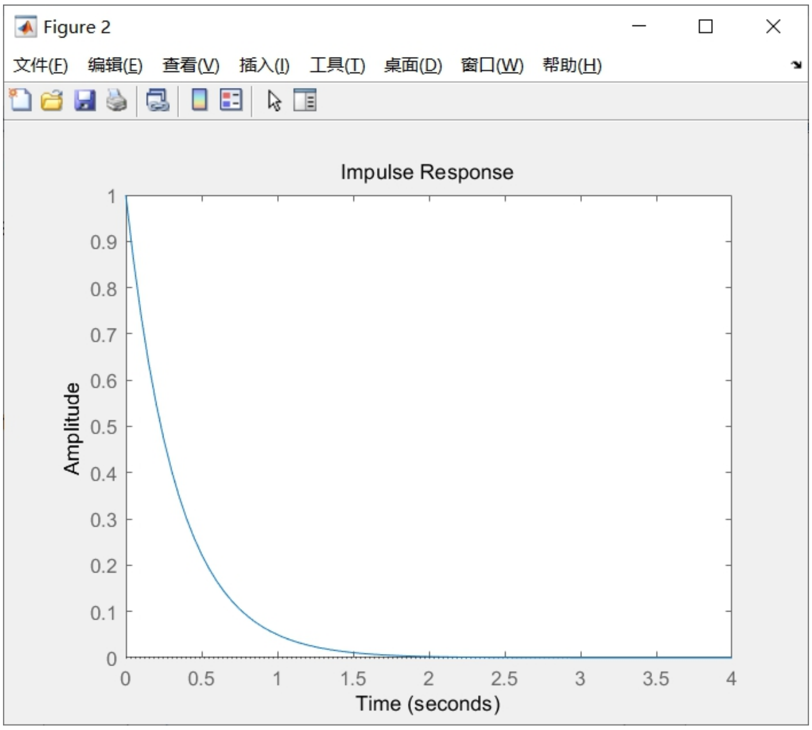

Figure 2.2.2 Image of the Impulse Response

It is obvious that the functions we computed in (a) is the same with the step and impulse responses computed by MATLAB, which means our solution is right.

MATLAB code:

```
% prob2b.m
clear;
clc;

% time vector
t=0:0.05:4;

% the coefficients of the differential equation
a=[1,3];
b=1;

% get the step response by using step command
figure;
step(b,a,t);
```

```
% get the impulse response by using impulse command
figure;
impulse(b,a,t);
```

(c) By using MATLAB, we can get the figures of the three outputs with a plot of $h(t)$ as follows:

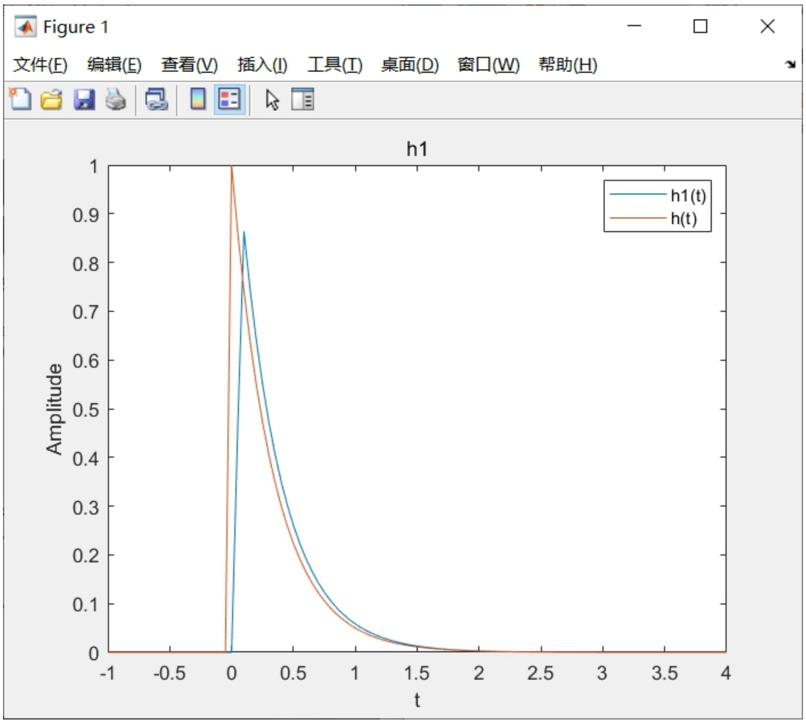

Figure 2.3.1 Image of $h_1(t)$

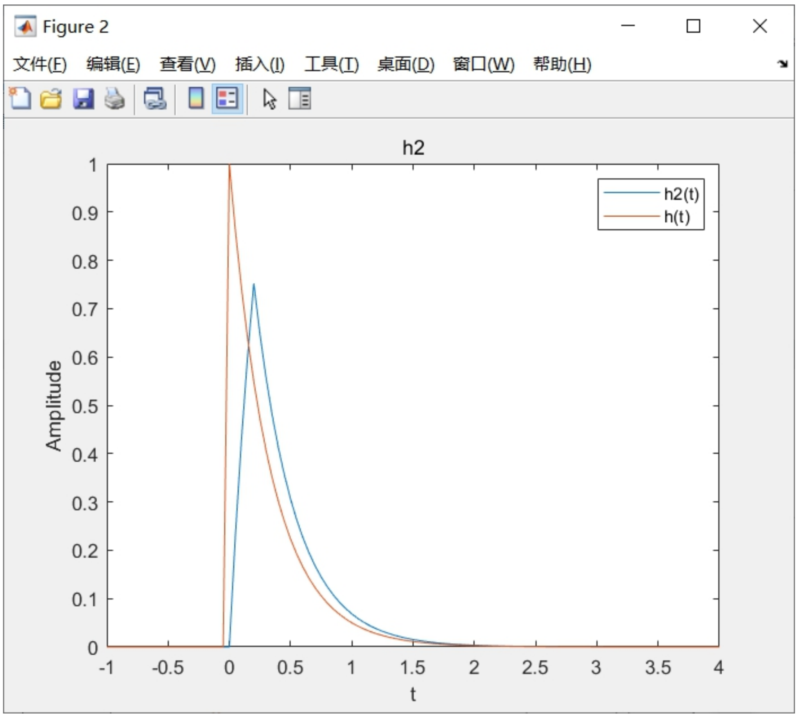

Figure 2.3.2 Image of $h_2(t)$

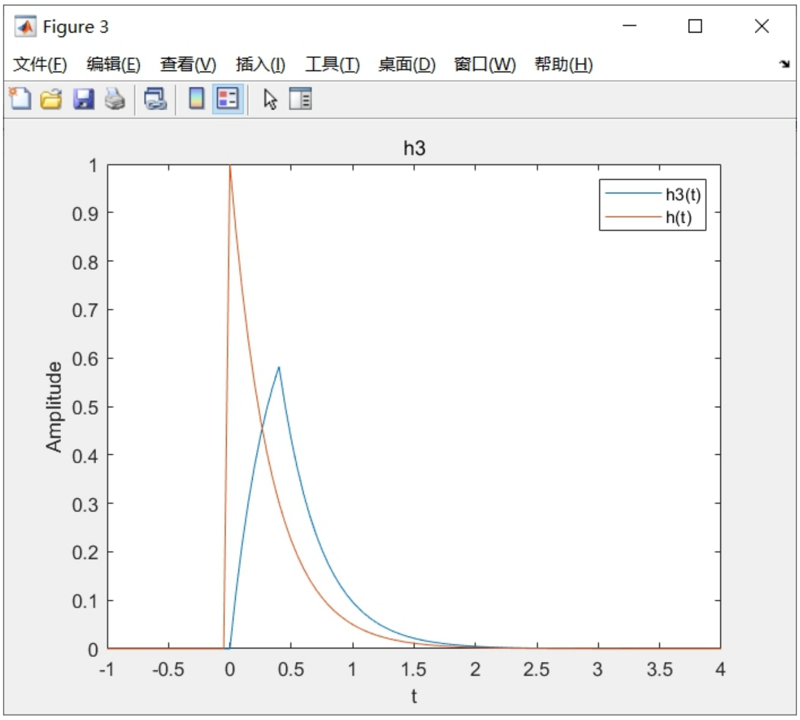

Figure 2.3.3 Image of $h_3(t)$

It is clear that $h_1^{\Delta}(t)$ is more like $h(t)$ than $h_2^{\Delta}(t)$ and $h_3^{\Delta}(t)$, and the bigger Δ is, the smaller the peak value of the output signal becomes, which means as Δ decreases, the output signal is closer to the unit impulse response $h(t)$, so it is better for us to use a narrower impulse signal to approximate the unit impulse signal.

MATLAB code:

```
% prob2c.m

clear;
clc;

% the time vector
t=-1:0.05:4;

% coefficients of the equation
a=[1 3];
b=1;

dt1=0.1;
dt2=0.2;
dt3=0.4;

% the input signals
d1=zeros(1,length(t));
% in the interval [0,dt), the inputs are 1/dt
d1(1,1/0.05+1:1/0.05+dt1/0.05)=1/dt1;
d2=zeros(1,length(t));
d2(1,1/0.05+1:1/0.05+dt2/0.05)=1/dt2;
d3=zeros(1,length(t));
d3(1,1/0.05+1:1/0.05+dt3/0.05)=1/dt3;

% simulate the corresponding outputs by using lsim
% command
h1=lsim(b,a,d1,t);
h2=lsim(b,a,d2,t);
h3=lsim(b,a,d3,t);

% the output we computed in (a)
h=exp(-3*t).*(t>=0);
```

```
% figure of h1(t)
figure;
plot(t,h1);
title('h1');
xlabel('t');
ylabel('Amplitude');
hold on;
plot(t,h); % plot h(t)
legend('h1(t)','h(t)');

% figure of h2(t)
figure;
plot(t,h2);
title('h2');
xlabel('t');
ylabel('Amplitude');
hold on;
plot(t,h);
legend('h2(t)','h(t)');

% figure of h3(t)
figure;
plot(t,h3);
title('h3');
xlabel('t');
ylabel('Amplitude');
hold on;
plot(t,h);
legend('h3(t)','h(t)');
```

(d) According to the definition of the unit impulse signal, its Integral over real numbers is 1, so we need to make sure that the pulse $\delta^{\Delta}(t)$ has unit area to fit the bill.

Furthermore, if the input signal is $D^{\Delta}(t)$, we can then get the corresponding outputs for $\Delta = 0.1, 0.2$ and 0.4 as follows:

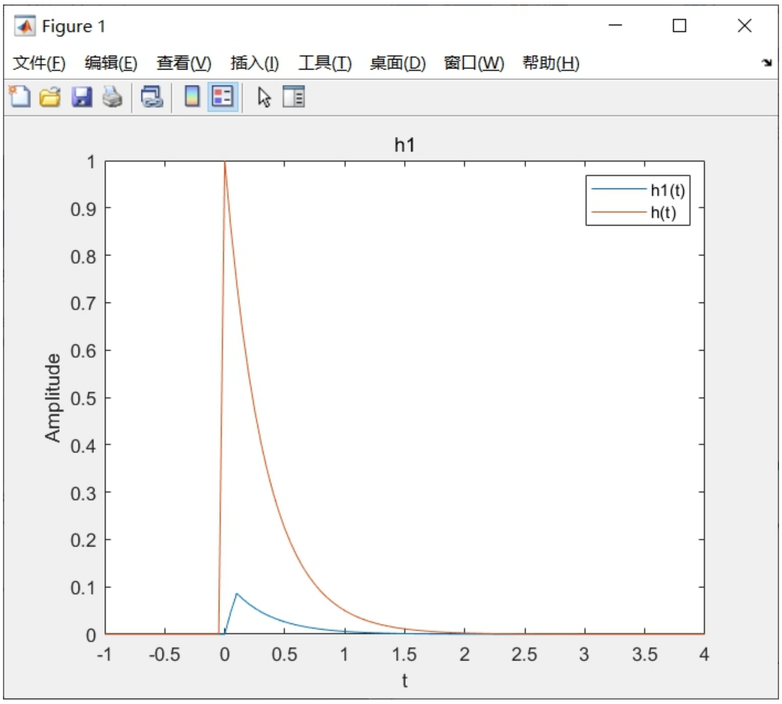

Figure 2.4.1 Image of $h_1(t)$

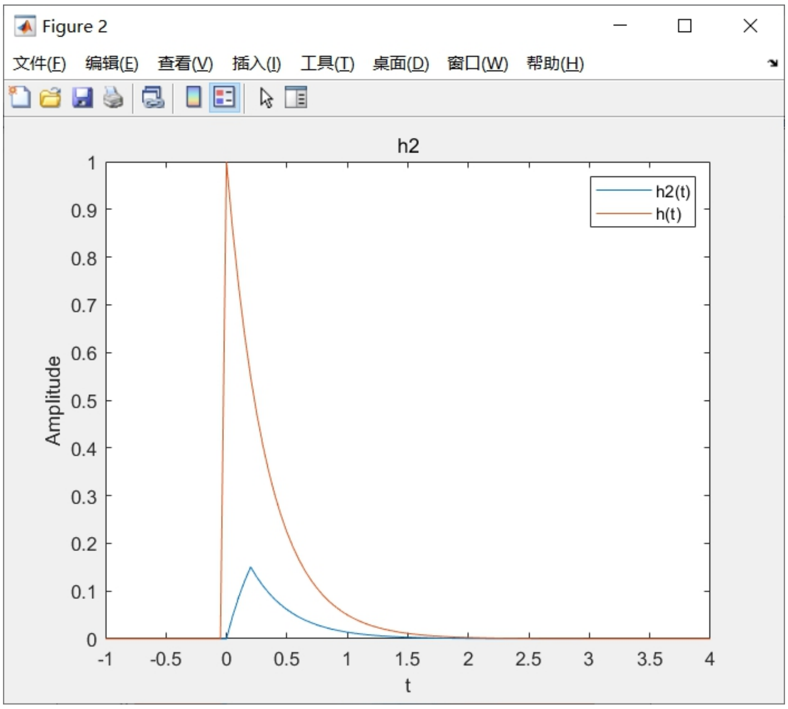

Figure 2.4.2 Image of $h_2(t)$

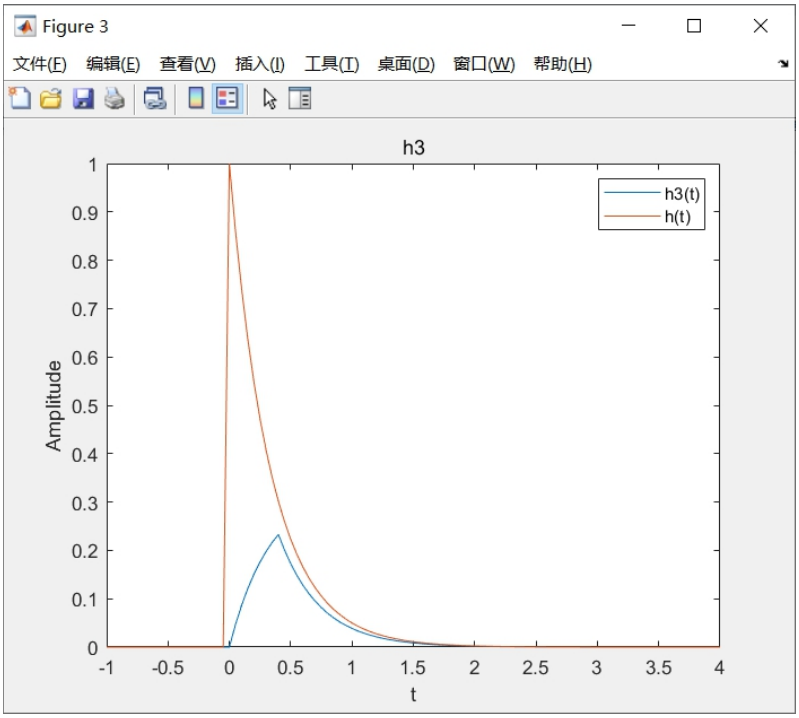

Figure 2.4.3 Image of $h_3(t)$

So as Δ increases to 1, the output signal becomes more like the unit impulse response. This is because the area of the input signal is rising to 1, which is closer to the requirement of the unit impulse signal. But since the input signal is not narrow enough, the output will still differ greatly from the impulse response even if $\Delta = 1$. Furthermore, if Δ is bigger than 1, it is predictable that the output will deviate from $h(t)$.

MATLAB code:

```
% prob2d.m

clear;
clc;

% the time vector
```

```
t=-1:0.05:4;

% coefficients of the equation
a=[1 3];
b=1;

dt1=0.1;
dt2=0.2;
dt3=0.4;

% the input signals
d1=zeros(1,length(t));
% in the interval [0,dt), the inputs are 1
d1(1,1/0.05+1:1/0.05+dt1/0.05)=1;
d2=zeros(1,length(t));
d2(1,1/0.05+1:1/0.05+dt2/0.05)=1;
d3=zeros(1,length(t));
d3(1,1/0.05+1:1/0.05+dt3/0.05)=1;

% simulate the corresponding outputs by using lsim
% command
h1=lsim(b,a,d1,t);
h2=lsim(b,a,d2,t);
h3=lsim(b,a,d3,t);

% the output we computed in (a)
h=exp(-3*t).*(t>=0);

% figure of h1(t)
figure;
plot(t,h1);
title('h1');
xlabel('t');
ylabel('Amplitude');
hold on;
plot(t,h);
legend('h1(t)','h(t)');

% figure of h2(t)
figure;
plot(t,h2);
title('h2');
xlabel('t');
ylabel('Amplitude');
```

```
hold on;
plot(t,h);
legend('h2(t)','h(t)');

% figure of h3(t)
figure;
plot(t,h3);
title('h3');
xlabel('t');
ylabel('Amplitude');
hold on;
plot(t,h);
legend('h3(t)','h(t)');
```

(e) According to the title, the area of $d_a(t)$ is

$$S = \int_{-\infty}^{+\infty} d_a(t)\, dt = a \int_{0}^{+\infty} e^{-at}\, dt =- e^{-at}|_0^{+\infty} = 1$$

which is the same with the area of unit impulse signal.

Since most of the signal energy is contained in the interval $[0, \frac{4}{a}]$, when a is sufficiently large, the interval will be narrow enough, then the response to this signal is identical to $h(t)$.

By using MATLAB, we can obtain the following output signals' figures:

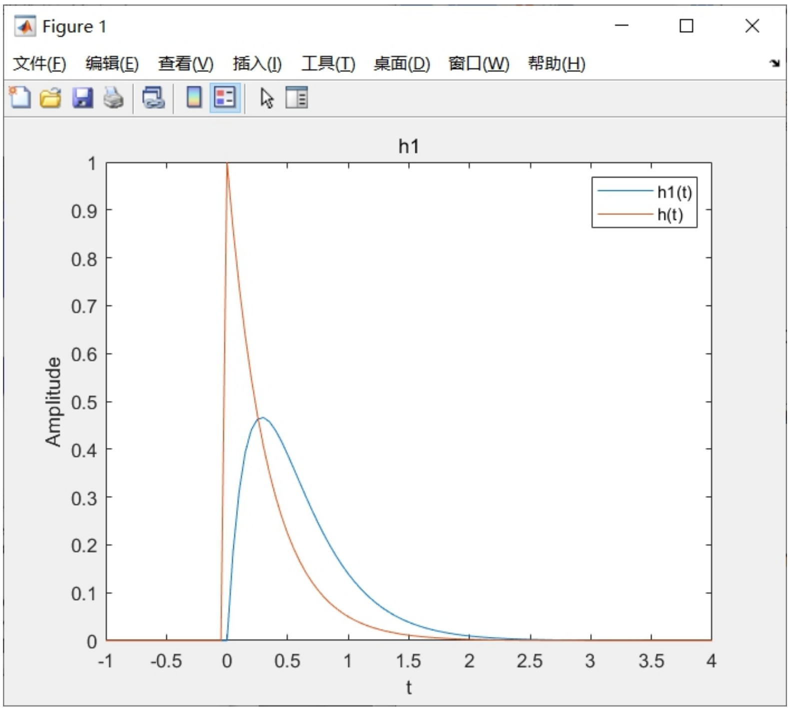

Figure 2.5.1 Image of $h_1(t)$

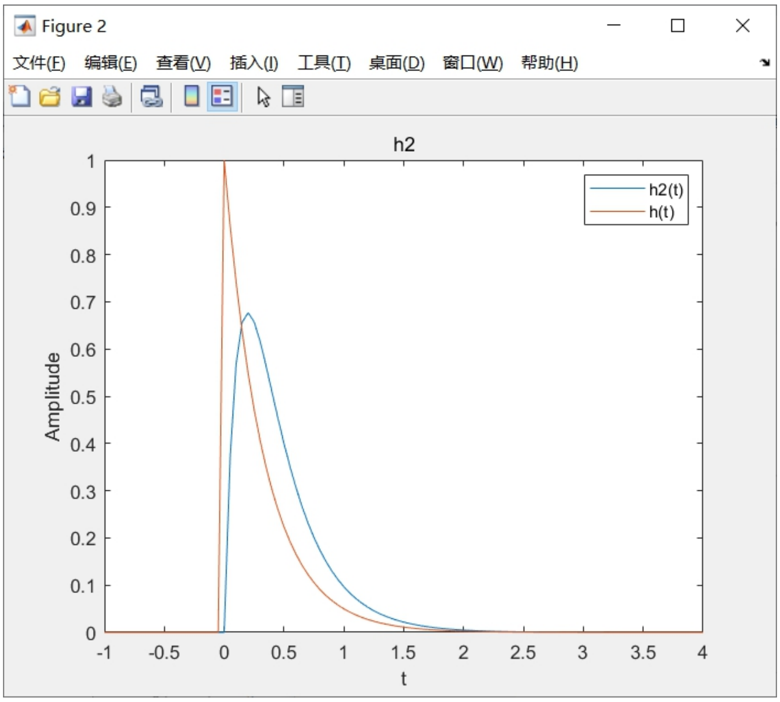

Figure 2.5.2 Image of $h_2(t)$

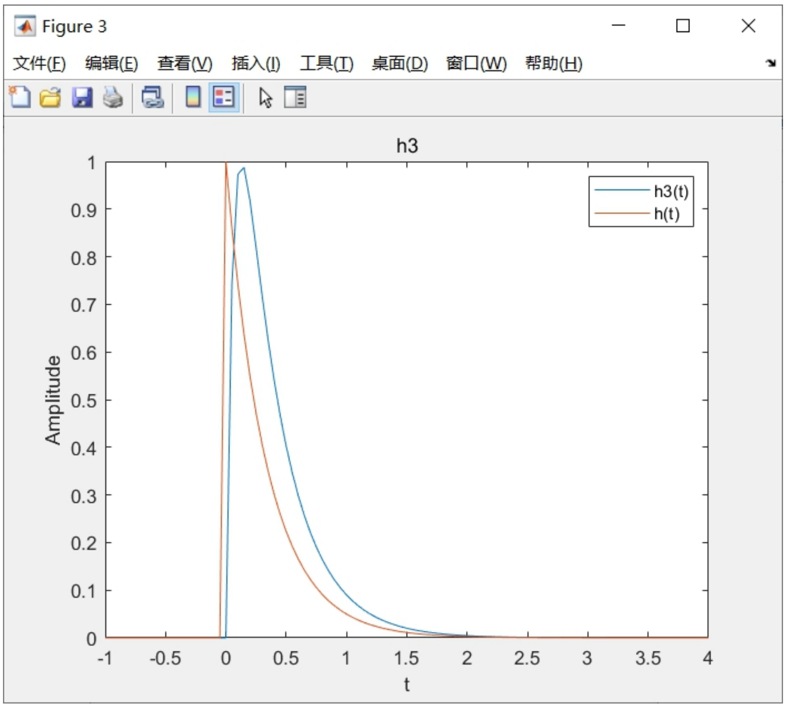

Figure 2.5.3 Image of $h_3(t)$

So as a increases, peak value of the output gets bigger and the response to $d_a(t)$ of the system becomes more like the unit impulse response, which meets our theory mentioned above.

MATLAB code:

```
% prob2e.m

clear;
clc;

% the time vector
t=-1:0.05:4;

% coefficients of the equation
a=[1 3];
b=1;

a1=4;
a2=8;
a3=16;
```

```
% values of da(t)
da1=a1*exp(-a1*t).*(t>=0);
da2=a2*exp(-a2*t).*(t>=0);
da3=a3*exp(-a3*t).*(t>=0);

% simulate the output signals by using lsim command
h1=lsim(b,a,da1,t);
h2=lsim(b,a,da2,t);
h3=lsim(b,a,da3,t);

% the output we computed in (a)
h=exp(-3*t).*(t>=0);

% figure of h1
figure;
plot(t,h1);
title('h1');
xlabel('t');
ylabel('Amplitude');
hold on;
plot(t,h); % plot h(t)
legend('h1(t)','h(t)');

% figure of h2
figure;
plot(t,h2);
title('h2');
xlabel('t');
ylabel('Amplitude');
hold on;
plot(t,h);
legend('h2(t)','h(t)');

% figure of h3
figure;
plot(t,h3);
title('h3');
xlabel('t');
ylabel('Amplitude');
hold on;
plot(t,h);
legend('h3(t)','h(t)');
```

Problem 3

Solutions:

(a) Since the input of the system is unit impulse signal, that is,

$$x[n] = \begin{cases} 1, & n = 0 \\ 0, & Otherwise \end{cases}$$

then the impulse response is

$$h[n] = x[n] + 0.5x[n-1000]$$
$$= \begin{cases} 1, & n = 0 \\ 0.5, & n = 1000 \\ 0, & Otherwise \end{cases}$$

By using MATLAB, we plotted the output signal as follows:

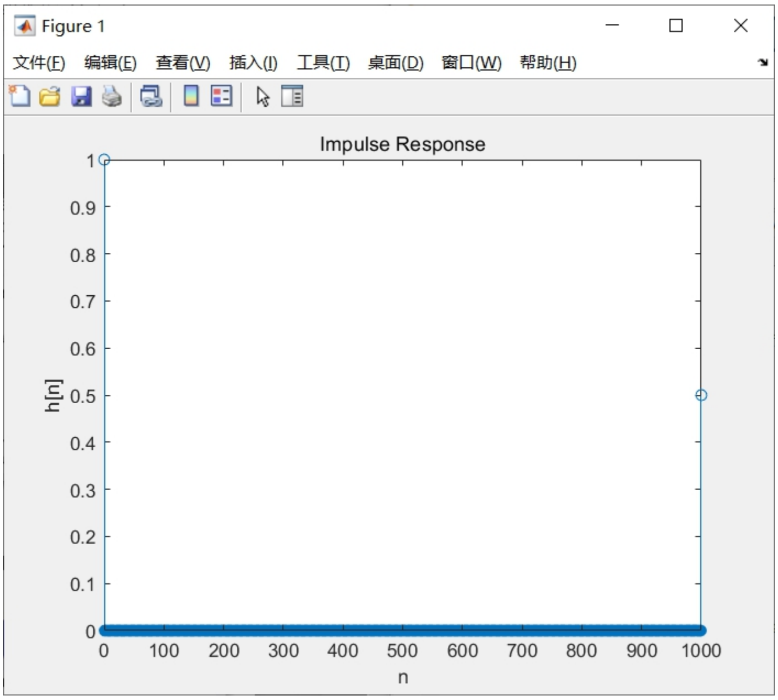

Figure 3.1 Impulse Response of the System

MATLAB code:

```
% prob3a.m
clear;
clc;

N=1000;
alpha=0.5;
```

```
% the coefficients of the difference equation (2.21)
a=1;
b=zeros(1,N+1);
b(1)=1;
b(N+1)=alpha;

% the input signal
delta=zeros(1,N+1);
delta(1)=1;

% simulate and plot the impulse response using the
% filter command
he=filter(b,a,delta);
stem(0:1000,he);
title('Impulse Response');
xlabel('n');
ylabel('h[n]');
```

(b) According to the title, we have,

$$y[n] = x[n] + \alpha x[n-N] \qquad ①$$

$$z[n] + \alpha z[n-N] = y[n] \qquad ②$$

Substitute ① into ②, we can obtain that

$$z[n] + \alpha z[n-N] = x[n] + \alpha x[n-N]$$

and it is obvious that $z[n] = x[n]$ is a valid solution to the difference equation derived above, which means if we take the output $y[n]$ of system ① as the input signal of system ②, it will produce an output equal to the input to system ①. So ② is indeed an inverse of ①.

The following output signal we get by MATLAB proves our derivation, and the equation's input is the output we obtained in (a), while the output is the unit impulse signal:

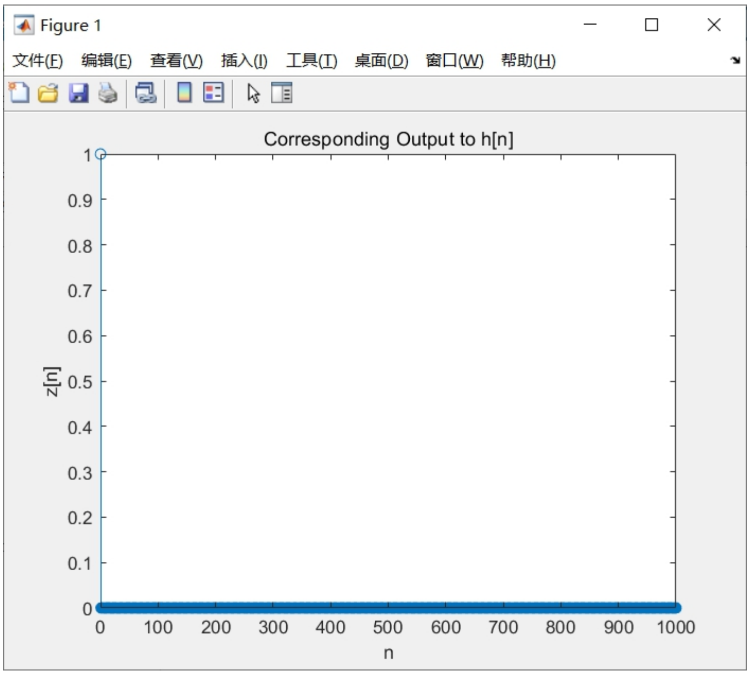

Figure 3.2 Image of z[n]

MATLAB code:

```
% prob3b.m
clear;
clc;

N=1000;
alpha=0.5;

% the coefficients of the difference equations
a=1;
b=zeros(1,N+1);
b(1)=1;
b(N+1)=alpha;

% the input signal in (a)
delta=zeros(1,N+1);
delta(1)=1;

% the impulse response in (a)
he=filter(b,a,delta);

% simulate and plot the output of the system by using
```

```
% filter
z=filter(a,b,he);
stem(0:1000,z);
title('Corresponding Output to h[n]');
xlabel('n');
ylabel('z[n]');
```

(c) By using MATLAB, we have:

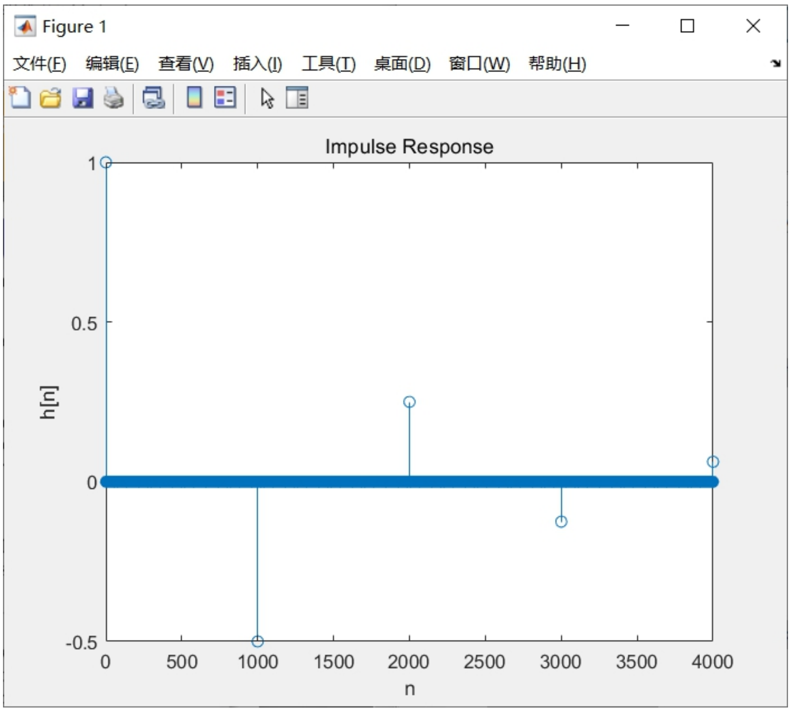

Figure 3.3 Impulse Response of System 2.22

It can be seen that for $n = 1000i$,

$$h[n] = (-\frac{1}{2})^i$$

where $i = 0,1,2,3,4$;

and at other points, $h[n] = 0$.

MATLAB code:

```
% prob3c.m
clear;
```

```
clc;

N=1000;
alpha=0.5;

% the coefficients of the difference equations
a=zeros(1,N+1);
a(1)=1;
a(N+1)=alpha;
b=1;

% the input signal
d=[1 zeros(1,4000)];

% simulate the impulse response by filter
her=filter(b,a,d);
stem(0:4000,her);
title('Impulse Response');
xlabel('n');
ylabel('h[n]');
```

(d) Implemented the echo removal system, we get the following output:

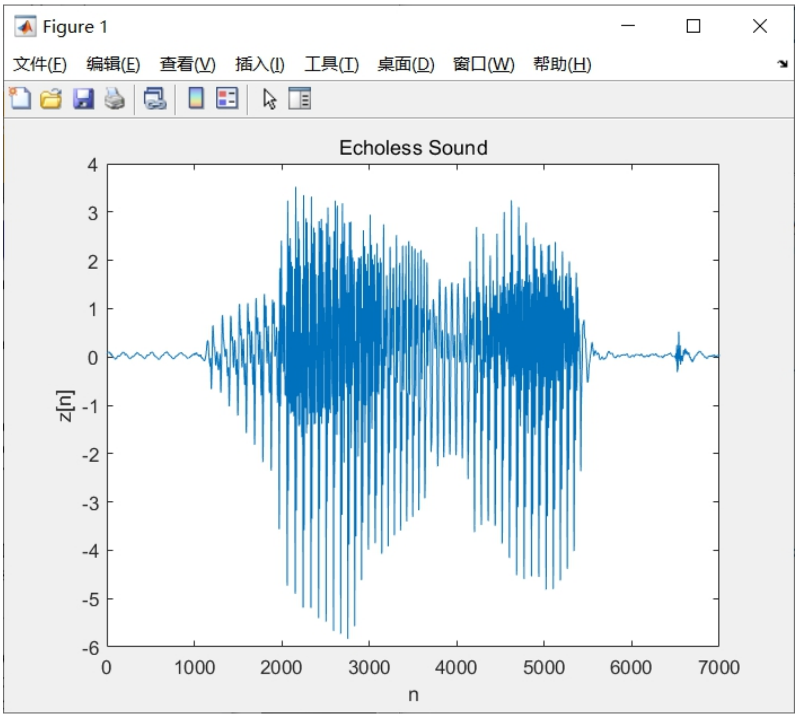

Figure 3.4 Image of Output Sound

And the output sound has no longer the echo, which is much clearer than the original one. Although the audio file only record one clip, we can still tell surely that the echo removal system did work by the non-echo sound.

MATLAB code:

```
% prob3d.m
clear;
clc;

load('lineup.mat')

N=1000;
alpha=0.5;

% the coefficients of the difference equation
a=zeros(1,N+1);
a(1)=1;
a(N+1)=alpha;
b=1;

% simulate the output by filter command
z=filter(b,a,y);
plot(z);
title('Echoless Sound');
xlabel('n');
ylabel('z[n]');

% hear the output sound
sound(z,8192);


% save the wave file
fs=8000;
audiowrite('lineup.wav',z/6,fs);
% prevent z from being cut
```