

Signal and System Project 1

HeXiang Huang

2025 年 3 月 27 日

1 Problem 1

1.1 Making Continuous-Time Pole-Zero Diagrams

(a)

use the following code to make the pole-zero diagram of the system

```
1 %% Example
2 b = [1 -1];
3 a = [1 3 2];
4 zs = roots(b);
5 ps = roots(a);
6 figure(1);
7 subplot(2,2,1)
8 plot(real(zs),imag(zs),'o');
9 hold on;
10 plot(real(ps),imag(ps),'x');
11 title('Example')
12 grid;
13 axis([-3 3 -3 3])
14
15 %% Exercise a
16 % Exercise a1
17 a = [1 5];
18 b = [1 2 3];
```

```

19 aroot = roots(a);
20 broot = roots(b);
21 subplot(2,2,2)
22 plot(real(aroot),imag(aroot),'o');
23 hold on;
24 plot(real(broot),imag(broot),'x');
25 title('H(s)=\frac{s+5}{s^2+2s+3}$','Interpreter','
        latex')
26 grid;
27 axis([-6 2 -2 2])
28
29 % Exercise a2
30 a = [2 5 12];
31 b = [1 2 10];
32 aroot = roots(a);
33 broot = roots(b);
34 subplot(2,2,3)
35 plot(real(aroot),imag(aroot),'o');
36 hold on;
37 plot(real(broot),imag(broot),'x');
38 title('H(s)=\frac{2s^2+5s+12}{s^2+2s+10}$','
        Interpreter','latex')
39 grid;
40 axis([-2 0 -4 4])
41
42 % Exercise a3
43 a = [2 5 12];
44 b = [1 4 14 20];
45 aroot = roots(a);
46 broot = roots(b);
47 subplot(2,2,4)
48 plot(real(aroot),imag(aroot),'o');
49 hold on;

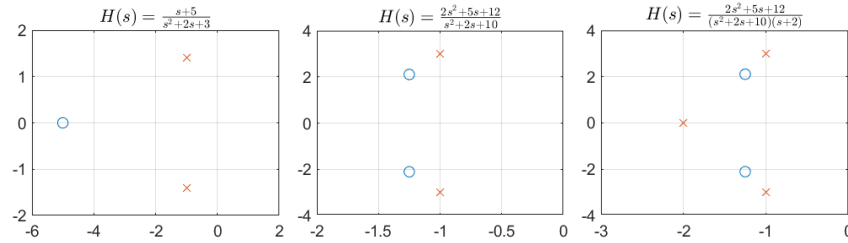
```

```

50 plot(real(broot),imag(broot),'x');
51 title('$H(s)=\frac{2s^2+5s+12}{(s^2+2s+10)(s+2)}$',
      'Interpreter','latex')
52 grid;
53 axis([-3 0 -4 4])

```

Then we can get the following pole-zero diagrams



(b)

A system is stable when the ROC includes the imaginary axis.

The poles of $H(s) = \frac{s+5}{s^2+2s+3}$ are $s = -1 \pm j\sqrt{2}$, the system is stable so that the ROC is $Re(s) > -1$

The poles of $H(s) = \frac{2s^2+5s+12}{s^2+2s+10}$ are $s = -1 \pm j\sqrt{3}$, the system is stable so that the ROC is $Re(s) > -1$

The poles of $H(s) = \frac{2s^2+5s+12}{(s^2+2s+10)(s+2)}$ are $s = -1 \pm j\sqrt{3}$ and $s = -2$, the system is stable so that the ROC is $Re(s) > -1$

(c)

Do the Laplace transform of the following equations

$$\frac{dy(t)}{dt} - 3y(t) = \frac{d^2x(t)}{dt^2} + 2\frac{dx(t)}{dt} + x(t)$$

we can get

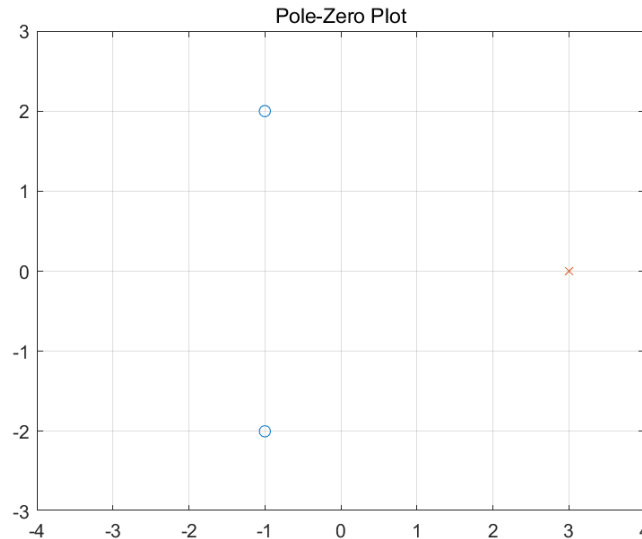
$$sY(s) - 3Y(s) = s^2X(s) + 2sX(s) + X(s)$$

so that

$$H(s) = \frac{Y(s)}{X(s)} = \frac{s^2+2s+1}{s-3}$$

The poles of $H(s) = \frac{s^2+2s+1}{s-3}$ are $s = -1 \pm j$, and the zeros are $s = 3$, and

we can draw the following pole-zero diagrams



(d)

In the function `pzplot`, it uses the function `roots` to find the poles and zeros of the system, and then plots the poles and zeros on the complex plane. For every pole, if the pole is on the left side of the given point, the ROC should contain the right side of the pole, and if the pole is on the right side of the given point, the ROC should contain the left side of the pole.

1.2 Making Discrete-Time Pole-Zero Diagrams

Note

In the M-file `dpzplot.m`, it uses the outdated function `clg`, in order to inform that the function works well, we use the function `clf` instead of `clg`.

1

```
%clf;
```

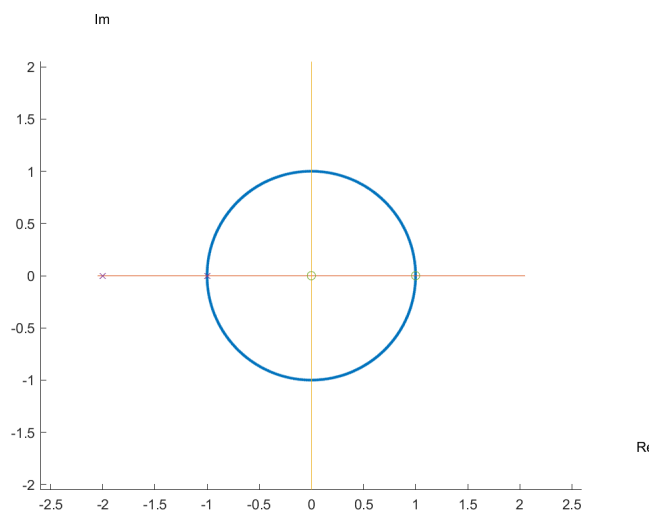
```
2      clf;
```

(a)

use the following easy code to make the pole-zero diagram of the system

```
1  b = [1 -1 0]; % 分子系数
2  a = [1 3 2]; % 分母系数
3  dpzplot(b, a); % 绘制零极点图
```

Then we can get the following pole-zero diagrams



(b)

Do the Z-transform of the following equations

$$y[n] + y[n-1] + 0.5y[n-2] = x[n]$$

we can get

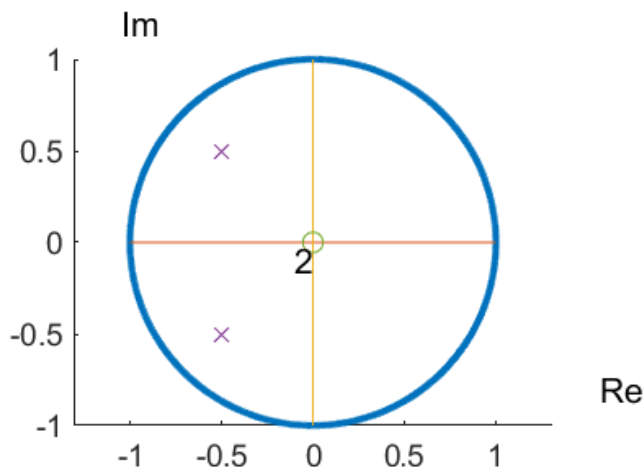
$$Y(z) + Y(z)z^{-1} + 0.5Y(z)z^{-2} = X(z)$$

so that

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{1+z^{-1}+0.5z^{-2}} = \frac{z^2}{z^2+z+0.5}$$

use the following code we can get the following pole-zero diagrams

```
1 b = [1 0 0];      % 分子系数
2 a = [1 1 0.5];    % 分母系数
3 dpzplot(b, a);    % 绘制零极点图
```



(c)

Do the Z-transform of the following equations

$$y[n] - 1.25y[n-1] + 0.75y[n-2] - 0.125y[n-3] = x[n] + 0.5x[n-1]$$

we can get

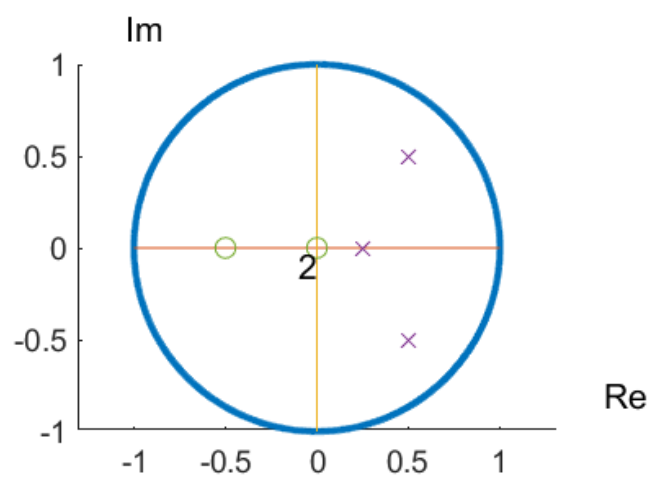
$$Y(z) - 1.25Y(z)z^{-1} + 0.75Y(z)z^{-2} - 0.125Y(z)z^{-3} = X(z) + 0.5X(z)z^{-1}$$

so that

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1+0.5z^{-1}}{1-1.25z^{-1}+0.75z^{-2}-0.125z^{-3}} = \frac{z^3+0.5z^2}{z^3-1.25z^2+0.75z-0.125}$$

use the following code we can get the following pole-zero diagrams

```
1 b = [1 0.5 0 0];  % 分子系数
2 a = [1 -1.25 0.75 -0.125]; % 分母系数
3 dpzplot(b, a);    % 绘制零极点图
```



2 Problem 2

2.1 Smiley

(a)

$$y[n] = (p * x)[n] = \sum_{k=-\infty}^{\infty} x[k]p[n-k]$$

In order to make $y[n]$ maximized when $n=2$, we need to make $x[k]p[2-k] = 1$ for $k = 1, 2, 3$ and $x[k]p[2-k] = 0$ for other k .

So we can get the following $p[n]$

$$\begin{aligned} p[-1] &= 1 \\ p[0] &= -1 \\ p[1] &= -1 \\ p[n] &= 0, n \neq 0, \pm 1 \end{aligned}$$

(b)

Now let us turn to finding nose.

Using the initial value of white and black pixels, we can notice that the white pixels contribute positive to the answer if they match but the black pixels contribute zero to the answer whether or not match. So we firstly subtract 127.5 from the pixel value so that black pixels and white pixels both contribute positively to the answer if they match and contribute negative when they don't match. One step further, we can normalize ± 127.5 to ± 1

Consider above process, the feature of nose is the following matrix

$$\begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & 1 & -1 & 1 & -1 \\ -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & 1 & 1 & 1 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

Consider the two-dimensional convolution and make $y[n,m]$ maximized when $[n,m]$ matches the row and column of the nose, we can get the following $p[n,m]$

$$\begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & 1 & 1 & 1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 1 & -1 & -1 \\ -1 & 1 & -1 & 1 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

We can use the following code to get the position of nose

```

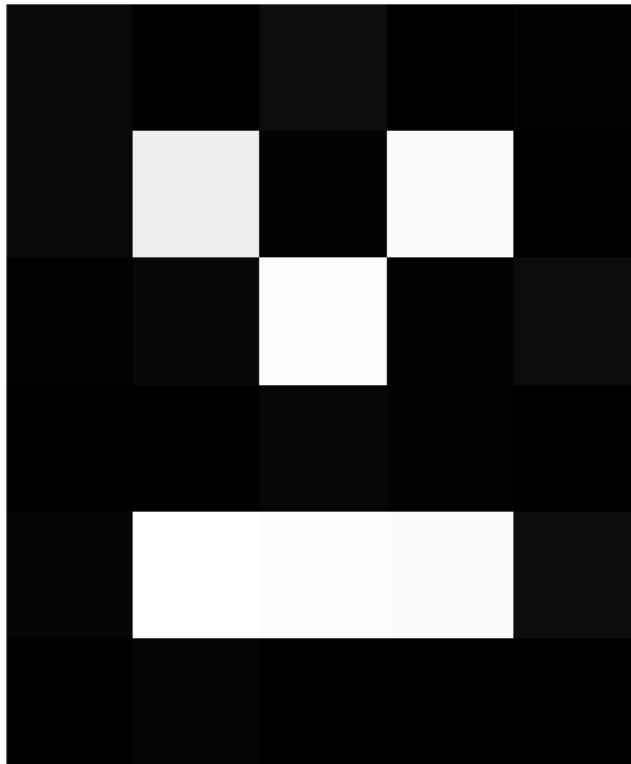
1  clc;
2  close all;
3  clear;
4  %read the image
5  img = imread("F:\School\大二下\信号与系统\project\
    project 1\introduction\Files for Problem2\
    findsmiley.jpg");
6  [img_row,img_column] = size(img);
7  %turn to double type in order to normalize the matrix
8  img_copy = double(img);
9  for i = 1:img_row
10     for j =1:img_column
11         if(img_copy(i,j)>200)
12             img_copy(i,j) = 1;
13         else
14             img_copy(i,j) = -1;
15         end
16     end
17 end
18 %create the matching matrix
19 p = [-1 -1 -1 -1 -1;
20      -1 1 1 1 -1;
21      -1 -1 -1 -1 -1;
```

```

22     -1 -1 1 -1 -1;
23     -1 1 -1 1 -1;
24     -1 -1 -1 -1 -1];
25 sum = 0;
26 sum_max = 0;
27 nose_row = 0;
28 nose_column = 0;
29 %do the convolution
30 for i =3:img_row-3
31     for j=3:img_column-2
32         img_matrix = img_copy(i-2:i+3,j-2:j+2);
33         for k = -2:3
34             for l =-2:2
35                 sum = sum+img_matrix(3+k,3+l)*p(4-k,3-
36                     l);
37             end
38         end
39         if(sum>sum_max)
40             nose_row = i;
41             nose_column =j;
42             sum_max = sum;
43         end
44         sum = 0;
45     end
46 end
47 %show the image
48 smiley_img = img(nose_row-2:nose_row+3,nose_column-2:
49     nose_column+2);
50 imshow(smiley_img, 'InitialMagnification', 'fit')

```

Run the code and we can know that the row of nose is 124 and the column of nose is 900 and we can get the following image



(c)

The problem requires us to add noise to the image and use the same method to find the nose. But I found that the noise already exists in the image, so I firstly clean the noise and then add noise to the image in order to use the function `normrnd`.

We can get the smiley without noise by the following code

```

1 clc;
2 close all;
3 clear;
4
5 %read imag
6 img = imread("F:\School\大二下\信号与系统\project\
   project 1\introduction\Files for Problem2\
   findsmiley.jpg");

```

```

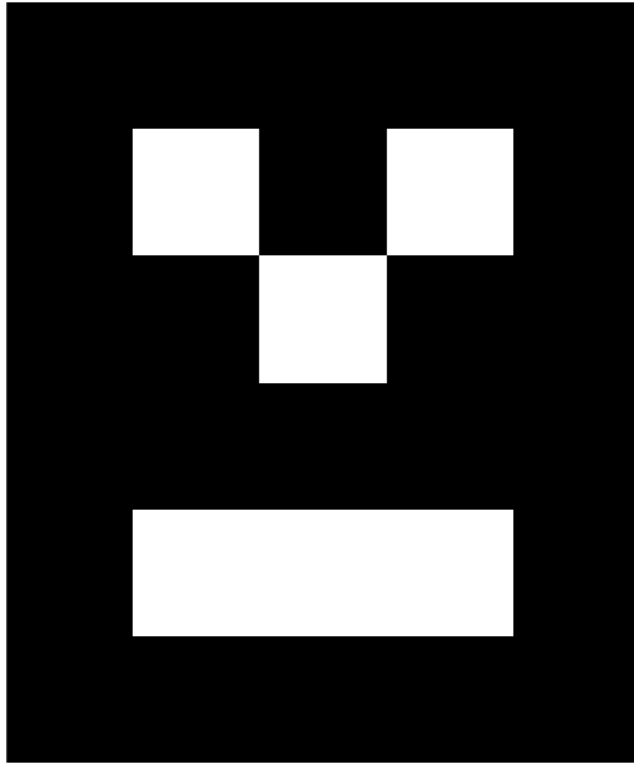
7 [img_row,img_column] = size(img);
8 img_copy = img;
9 % clean the noise
10 for i = 1:img_row
11     for j =1:img_column
12         if(img_copy(i,j)>200)
13             img_copy(i,j) = 255;
14         else
15             img_copy(i,j) = 0;
16         end
17     end
18 end
19 %save imag
20 imwrite(img_copy, 'F:\School\大二下\信号与系统\project\
    project 1\asset\smiley_without_noise.png')

```

Now let us process the image without noise.

Because of the noise, we can't normalize ± 127.5 to ± 1 , so I just subtract 127.5 from the pixel value.

Use the similar code (more detail in 2c code segment in problem 2.m), we can get that the row of nose is 124 and the column of nose is 900 and we can get the following image



We can see that the method works well even if the image has some noise.

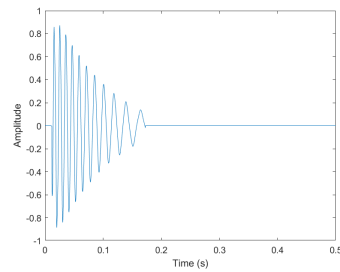
3 Problem 3

3.1 Use the Sounds in Matlab

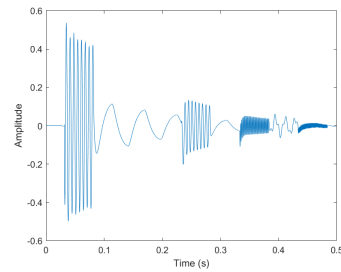
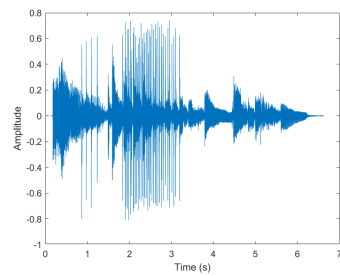
Using the following code, you can get the wave of the sound and play it.

```
1 clc;  
2 close all;  
3 clear;  
4  
5 % read .wav  
6 [y, Fs] = audioread('snare.wav');  
7  
8 % draw the wave  
9 t = (0:length(y)-1)/Fs;  
10 plot(t, y);  
11 xlabel('Time (s)');  
12 ylabel('Amplitude');  
13  
14 % play the .wav  
15 sound(y, Fs);
```

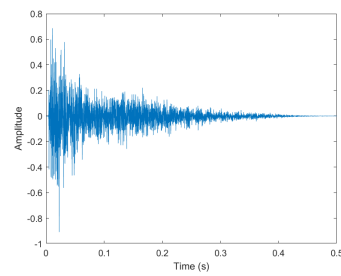
Some sound is too short to be listened clearly, but the sounds that can be listened clearly roughly meet my expectation. Now I put the wave of the sound.



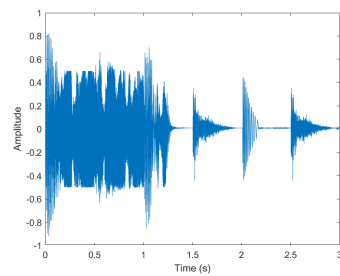
bassdrum



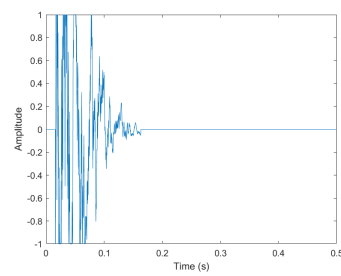
bleep



castanets44m



hatclosed



mixed

snare

A. Time-reverse & Timescale

The sound vector is a column vector, so we can use the function `flipud`.

In `TimeReverse.m`

```
1 function sound_rev = TimeReverse(y)
2     sound_rev = flipud(y);
3 end
```

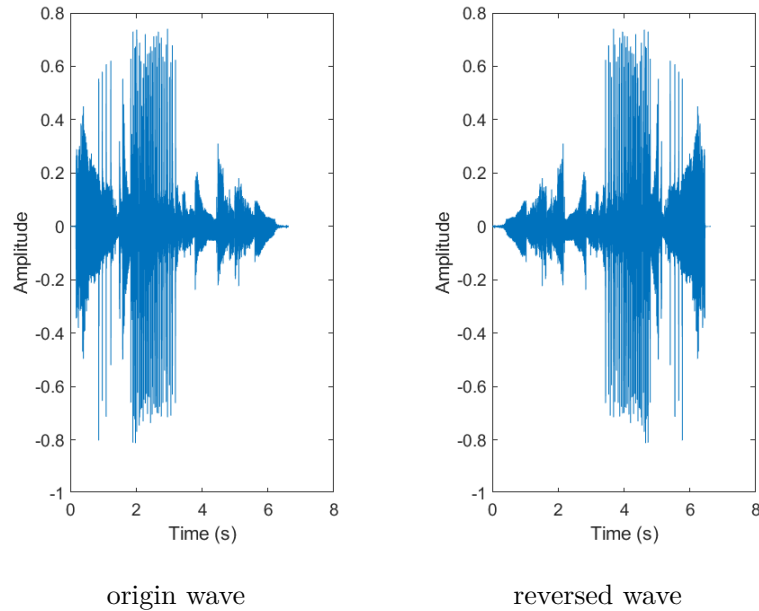
In `problem3.m`

```

1  clc;
2  close all;
3  clear;
4
5  [y, Fs] = audioread( 'castanets44m.wav' );
6  y_rev = TimeReverse(y);
7  % draw the wave
8  t = (0:length(y)-1)/Fs;
9
10 figure(1)
11 subplot(1,2,1)
12 plot(t, y);
13 xlabel( 'Time (s) ');
14 ylabel( 'Amplitude' );
15
16 subplot(1,2,2)
17 plot(t, y_rev);
18 xlabel( 'Time (s) ');
19 ylabel( 'Amplitude' );
20
21 % play the sound
22 sound(y_rev, Fs);

```


We can get two compared wave of the sound.



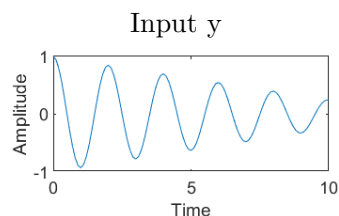
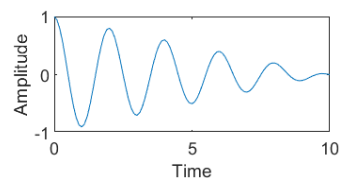
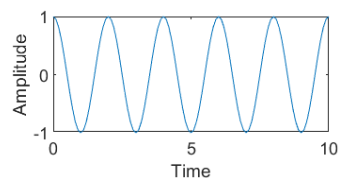
When use the function `timescale`, the pitch changes with the change of the frequency of the sound.

B.Fader

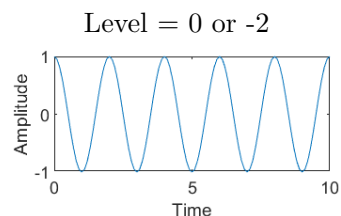
In `fade.m`

```
1 % process the level
2 if(level>1)
3     level = 1;
4 elseif(level<0)
5     level = 0;
6 end
7
8 %create the ramp vector
9 t = linspace(1,level,length(x));
10 % multiply the audio vector with the ramp vector to
   fade
11 y = t .* x;
```

Using the function(more detail in problem3.m),we can get the following plot.



Level = 0.25



Level = 100

C.Repeater

In repeat.m

```

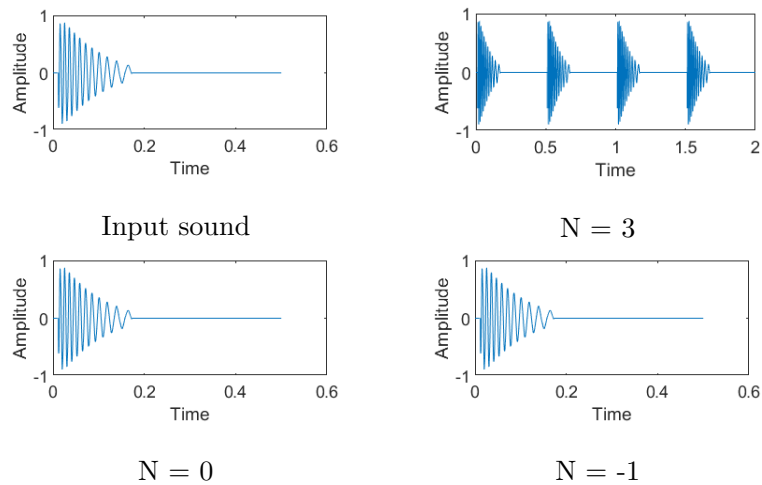
1 function out = repeat(sound,N)
2
3 %process N
4 if (N<0)
5     N = 0;
6 end
7 % turn N to integer
8 N = fix(N);
9
10 % repeat the sound
11 if (N==0)
12     out = sound;
13 else
14     out = sound;
15     for i=1:N
```

```

16         out = [out;sound];
17     end
18 end

```

Using the function(more detail in problem3.m),we can get the following plot.



D.Delay

In delay.m

```

1 function out =delay(sound,delay,Fs)
2
3 if(delay>0)
4     zero_length = delay*Fs;
5     zero_sound = zeros(zero_length,1);
6     out = [zero_sound;sound];
7 elseif(delay == 0)
8     out = sound;
9 else
10    advance_length = -delay*Fs;
11    if(advance_length<length(sound))

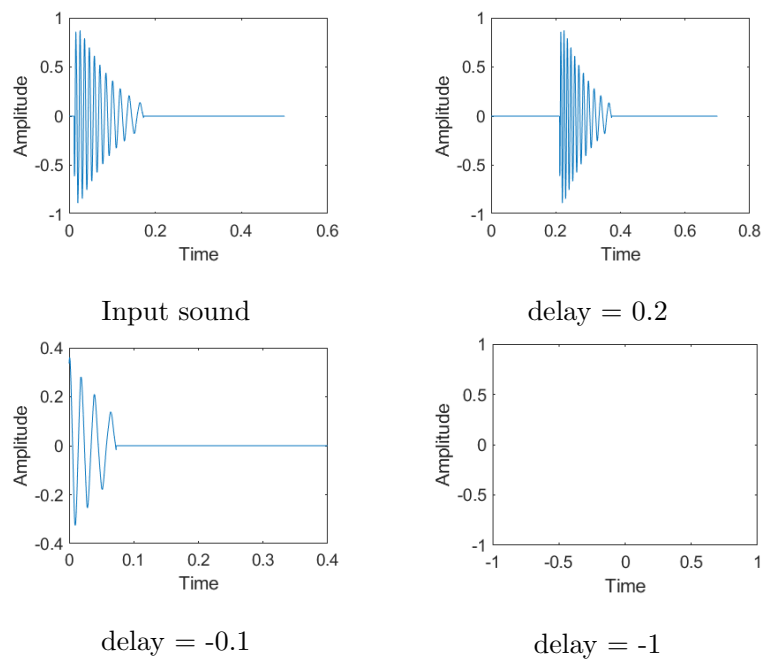
```

```

12         out = sound((advance_length+1):length(sound)
13                     ,:);
14     else
15         out = 0;
16     end
end

```

Use the function(more detail in problem3.m),we can get the following plot.



When I delay by a negative number, I put the sound advance.

E.Mixer

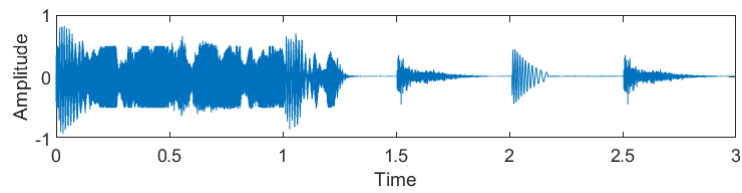
In order to provide more options for users, I offer gain1 for sound1 and gain2 for sound2. In the function I want to mix two sounds with both feature of sounds saved. So I firstly process gain1 and gain2 and in result gain1 plus gain2 equals to 1. In the end of the function, I try to amplify the sound so that the sound can

be listened clearly.

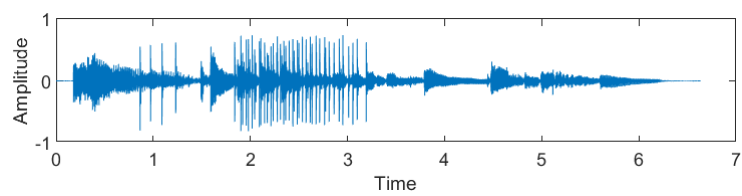
In mixer.m

```
1 function out = mix(sound1, sound2, gain1, gain2)
2
3 sound_copy1 = sound1;
4 sound_copy2 = sound2;
5
6 gainf1 = gain1/(gain1+gain2);
7 gainf2 = gain2/(gain1+gain2);
8
9 length1 = length(sound1);
10 length2 = length(sound2);
11 maxlength = max(length1, length2);
12 if(length1 < maxlength)
13     zerolength = maxlength - length1;
14     zerosound = zeros(zerolength, 1);
15     sound_copy1 = [sound_copy1; zerosound];
16 end
17
18 if(length2 < maxlength)
19     zerolength = maxlength - length2;
20     zerosound = zeros(zerolength, 1);
21     sound_copy2 = [sound_copy2; zerosound];
22 end
23
24 out = sound_copy1.*gainf1 + sound_copy2.*gainf2;
25 maxAmp = max(abs(out));
26 out = out.*(1/maxAmp);
```

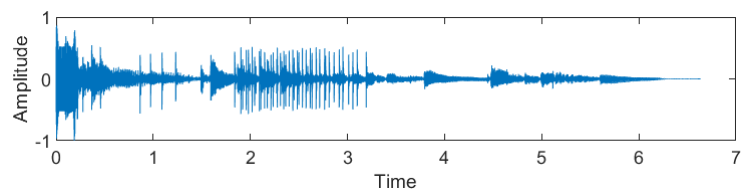
Using the function (more detail in problem3.m) with $\text{gain1} = 3$ and $\text{gain2} = 2$, we can get the following plot.



sound1



sound2



mix sound