Lab7 Linear Predictive Analysis of Speech Signal

张旭东 12011923

1. Introduction

In this experiment, we mainly analyze the problem of Linear Predictive Analysis of speech signal. LPC analysis is used for feature extraction to exact the key parameters reflecting the signal features to reduce the dimension and facilitate subsequent processing. LPC plays an important role in speech recognition, speech coding, formant spectrum envelope estimation, speech synthesis and so on.

The main idea of LPC is that the sample of a speech signal can be approximated by the linear combination of several samples in the past. However, linear prediction has an error with the actual sample value. By approximating the actual sampling in the sense of minimum mean square error, the coefficients α_k can be obtained.

2. Lab result and analysis

2.1 Problem1

The requirement of problem1 is to write a MATLAB program to perform LPC analysis on a frame of speech and plot the original speech signal for the specified frame, the LPC error signal for the specified frame, the signal short-time Fourier transform log magnitude spectrum(dB) along with the LPC spectrum for the specified frame and the error signal log magnitude spectrum.

The process of LPC analysis is as follow:

1. do segmentation and extract the frame using window function(always Hamming window). The original signal is denoted by s_n and the frame is denoted by $s_{\hat{n}}$.
2. According to the basic principle of LPC analysis, the signal can represented by sum of coefficient a_k and delay.

$$s(n) = \sum_{k=1}^{p} a_k s(n-k) \tag{1}$$

3. The main difficulty is to solve the coefficient a_k. Basically, a_k is chosen to minimize the mean-squared prediction error:

$$E_{\hat{n}} = \sum_m s_{\hat{n}}^2(m) - \sum_{k=1}^{p} a_k \sum_m s_{\hat{n}}(m) s_{\hat{n}}(m-k) \tag{2}$$

4. Usually, we use the **Autocorrelation Method** after we do segments, we define:

$$R_{\hat{n}}[k] = \sum_{m=0}^{L-1-k} s_{\hat{n}}[m] s_{\hat{n}}[m+k], k = 1, 2, \dots, p \tag{3}$$

5. The relationship between coefficient a_k and R is :

$$\sum_{k=1}^{p} a_k R_{\hat{n}}(|i-k|) = R_{\hat{n}}(i) \tag{4}$$

$$\begin{bmatrix} R_{\hat{n}}(0) & R_{\hat{n}}(1) & . & . & R_{\hat{n}}(p-1) \\ R_{\hat{n}}(1) & R_{\hat{n}}(0) & . & . & R_{\hat{n}}(p-2) \\ . & . & . & . & . \\ R_{\hat{n}}(p-1) & R_{\hat{n}}(p-2) & . & . & R_{\hat{n}}(0) \end{bmatrix} \times \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ . \\ . \\ \alpha_p \end{bmatrix} = \begin{bmatrix} R_{\hat{n}}(1) \\ R_{\hat{n}}(2) \\ . \\ . \\ R_{\hat{n}}(p) \end{bmatrix} \tag{5}$$

6. According to the `formula(5)` a toplitz matrix is constructed to solve the coefficients a_k and the resynthesized speech signal is:

$$\hat{s}_n = \sum_{k=1}^{p} a_k s(n-k) \tag{6}$$

the error is :

$$e(n) = s(n) - \hat{s}(n) \tag{7}$$

The final LPC model consists of the LPC parameters, $\{\alpha_k\}, k = 1, 2, \dots, p$ and the gain, G. The gain is determined by the matching the energy of the model to the short-time energy of the speech signal, which is:

$$G = E_{\hat{n}} = \sum_{m} (e_{\hat{n}}(m))^2 = R_{\hat{n}}(0) - \sum_{k=1}^{p} \alpha_k R_{\hat{n}}(k) \tag{8}$$

The spectrum of LPC is :

$$H(e^{j\omega}) = \frac{G}{1 - \sum_{k=1}^{p} a_k e^{-j\omega k}} \tag{9}$$

The function `LPC` I write is as below:

```
function [a,err]=LPC(x,order)
        %x is the frame
        R0=sum(x.^2);
        R_1_p=zeros(1,order);
        for k=1:order
            for m=0:length(x)-1-k
                R_1_p(k)=R_1_p(k)+x(m+1).*x(m+1+k);
            end
        end
        R_0_p=[R0 R_1_p(1:order-1)];
        A=toeplitz(R_0_p,R_0_p);
        a=(A^(-1))*R_1_p';


        %compute singal hat
        sig_hat=zeros(1,length(x));
        framepad=[zeros(1,order),x];
        for n=1:length(x)
            for k=1:order
                sig_hat(n)=sig_hat(n)+a(k).*framepad(order+n-k);
            end
        end
        err=x-sig_hat;

end
```

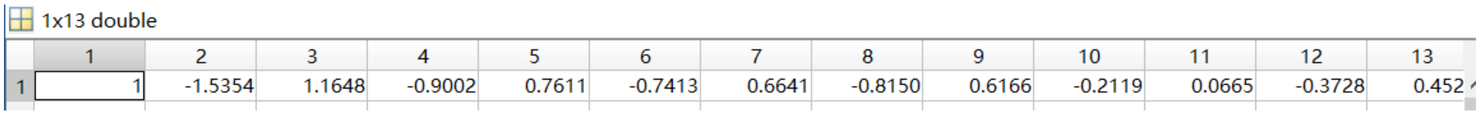
Then we test our code using the file `test_16k.wav` with the following frame parameters, including starting sample 6000, frame size of 640 samples and the LPC order of 12. And plot the original speech signal for the specified frame, the LPC error signal for the specified frame, the signal short-time Fourier transform log magnitude spectrum(dB) along with the LPC spectrum for the specified frame and the error signal log magnitude spectrum. Note that we should do STFT to the frame and error but for this frame extracted by the Hamming window, STFT is equal to do the DTFT.

The code is as follow:

```
[s,fs]=audioread('test_16k.wav');
framesize=640;
starting_sample=6000;
LPC_order=12;

%extract frame hamming window
window=hamming(640);
frame=s(starting_sample:starting_sample+framesize-1)'.*window';
[a,error]=LPC(frame,LPC_order);

%using MATLAB function
A=lpc(frame,LPC_order);

%spectrum
Frame=fft(frame,1024);
Error=fft(error,1024);
Frame=10*log10(abs(Frame));
Error=10*log10(abs(Error));

%LPC spectrum
G=sqrt(sum(error.^2));
H=(G./(1-fft([0 a'],1024)));
H=10*log10(abs(H));

figure;
subplot(4,1,1);
t=s(starting_sample:starting_sample+framesize-1);
plot((starting_sample/fs):1/fs:((starting_sample+framesize-1)/fs),t');
title('original speech frame');
```

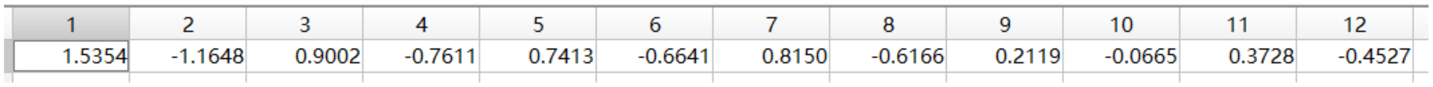
```
xlabel('time(s)');

subplot(4,1,2);
plot((starting_sample/fs):1/fs:((starting_sample+framesize-1)/fs),error);
title('LPC error signal');
xlabel('time(s)');

subplot(4,1,3)
plot(-fs/2:1/length(H)*fs:fs/2-1/length(H)*fs,H);
title('the magnitude spectrum of STFT and LPC for the specified frame');
hold on;
plot(-fs/2:1/length(Frame)*fs:fs/2-1/length(Frame)*fs,Frame);
xlabel('frequency(Hz)');
ylabel('magnitude(dB)');
legend('LPC spectrum','STFT spectrum');


subplot(4,1,4);
plot(-fs/2:1/length(Error)*fs:fs/2-1/length(Error)*fs,Error);
title('log magnitude spectrum of error signal');
xlabel('frequency(Hz)');
ylabel('magnitude(dB)');
```

Then the MATLAB function `lpc` is used to calculate a_k and compare the result using the MATLAB function `lpc` and that using our function `LPC`.



1x13 double

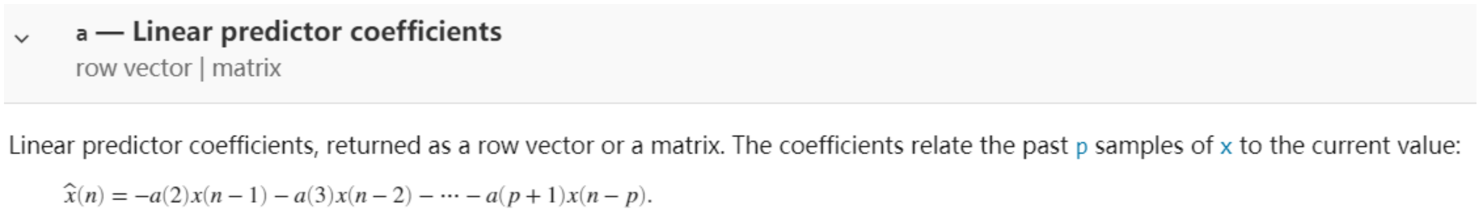
	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	-1.5354	1.1648	-0.9002	0.7611	-0.7413	0.6641	-0.8150	0.6166	-0.2119	0.0665	-0.3728	0.452



1	2	3	4	5	6	7	8	9	10	11	12
1.5354	-1.1648	0.9002	-0.7611	0.7413	-0.6641	0.8150	-0.6166	0.2119	-0.0665	0.3728	-0.4527

Fig.1 ak calculated using MATLAB function(above) and using written function(below)

Then we consult the help document of `lpc`.



a — Linear predictor coefficients

row vector | matrix

Linear predictor coefficients, returned as a row vector or a matrix. The coefficients relate the past p samples of x to the current value:

$$\hat{x}(n) = -a(2)x(n-1) - a(3)x(n-2) - \cdots - a(p+1)x(n-p).$$

Fig.2 output argument of ak about MATLAB function lpc

From the help document of `lpc`, what can be known is that $-A(2:p+1)$ is the coefficient a_k we want, which is the same as the coefficient a_k using written function. That means the function `lpc` we written is correct.

The plot of the original speech signal for the specified frame, the LPC error signal for the specified frame, the signal short-time Fourier transform log magnitude spectrum(dB) along with the LPC spectrum for the specified frame and the error signal log magnitude spectrum is as below:

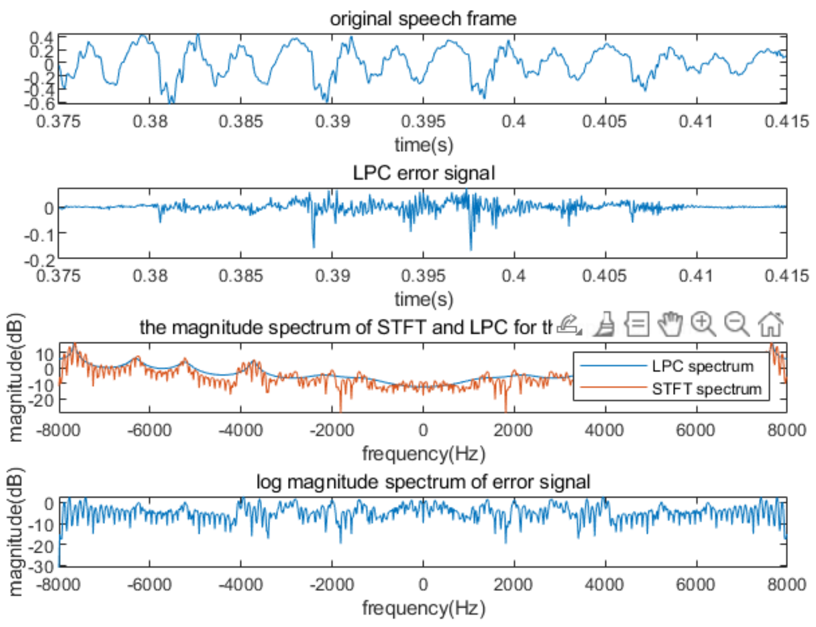


Fig.3 result with order p=12

From the above picture, it is obvious that the LPC spectrum is similar to that of STFT. The LPC spectrum is an envelope of speech frame spectrum. The reason behind this is that a_k is calculated by autocorrelation function, which can eliminate the influence of excitation, making the spectrum have a more clear spectral structure. What's more, the error at the beginning and end of the frame is small due to the effect of Hamming window. The Hamming window can weaken the influence of the frame edge because the value of Hamming window near frame edge is about zero. Besides, the frequency spectrum of the error have a similar change trend to that of the frame, but the energy of it is lower than that of frequency spectrum of frame.

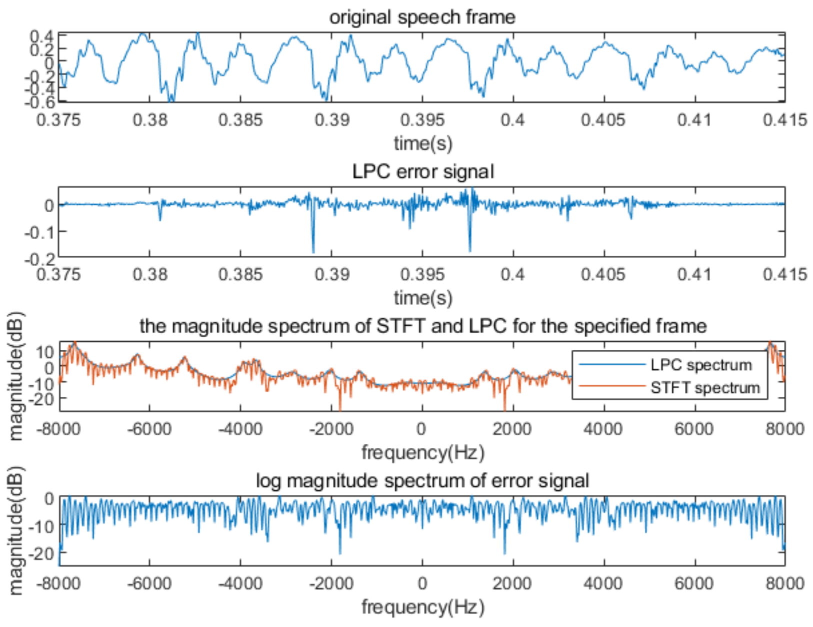


Fig.4 result with order p=30

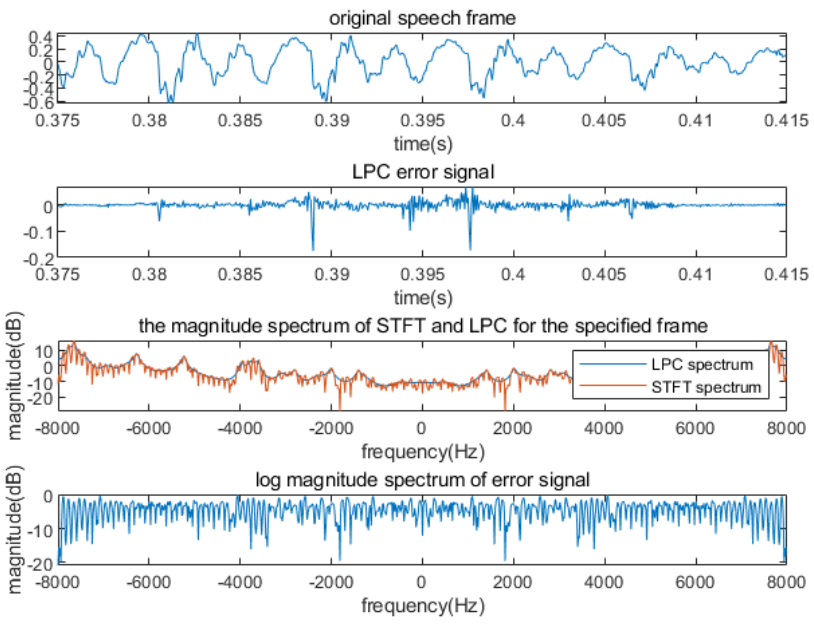


Fig.5 result with order p=50

From the above pictures, with the order increasing, the amplitude of error signal at some places decreases and the envelope of frequency spectrum of error signal is more flat. What's more, the LPC spectrum is more closer to the envelope of the STFT spectrum. However, this will increase computation and complexity.

2.2 Problem2

The requirement of problem1 is to write a MATLAB program to analyze a speech file using LPC analysis methods, extract the error signal, and then use the error signal to do an exact reconstruction of the original speech file. Then we use the speech file `s5.wav` as input with LPC frame size of 320 samples, LPC frame shift of 80 samples and LPC order $p = 12$.

The code is as follow:

```
[s,fs]=audioread('s5.wav');
framesize=320;
frameshift=80;
order=12;
sig_hat=zeros(1,length(s));
window=hamming(framesize);

[a0,error0]=LPC(s(1:framesize)'.*window',order);
sig_hat(1:framesize)=s(1:framesize)'-error0;



for n=2:(length(s)-framesize)/frameshift

    frame=s((n-1)*frameshift+1:(n-1)*frameshift+framesize)'.*window';

    [a,error]=LPC(frame,order);

    sig_hat((n-1)*frameshift+1:(n-1)*frameshift+framesize)=s((n-
1)*frameshift+1:(n-1)*frameshift+framesize)'-error;

end
error_whole=s'-sig_hat;
```

```
figure;
subplot(3,1,1);
t=1/fs:1/fs:length(s)/fs;
plot(t,s');
title('original speech signal');
xlabel('time(s)');

subplot(3,1,2);
plot(t,error_whole);
title('LPC error signal');
xlabel('time(s)');
axis([0,length(s)/fs,-1,1]);

subplot(3,1,3)
plot(t,sig_hat);
title('Resynthesized speech signal');
xlabel('time(s)');
axis([0,length(s)/fs,-1,1]);
```

The result is as below:

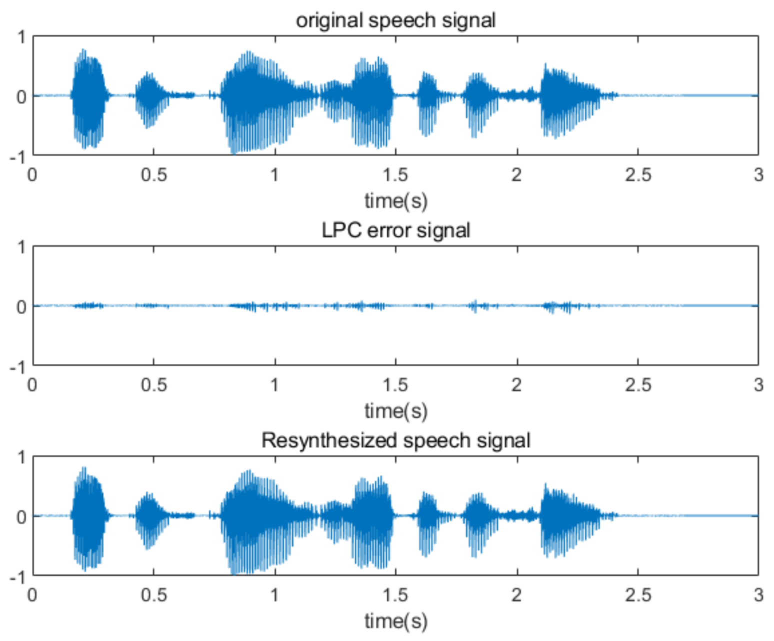


Fig.7 LPC analysis for the speech file

From the above picture, it is obvious that the amplitude of error signal is much less than that of original speech signal and resynthesized speech signal reconstructed by LPC. The resynthesized speech signal maintains the amplitude and waveform close to the original signal, which means the algorithm is correct.

Listen to all the three files. The original signal sounds the clearest and the resynthesized speech signal is also, but its volume changes slightly and is a little hoarse. As for the error signal, I can still hear the fluctuation of original voice. However, it is difficult to understand the meaning.

2.3 Problem3

The requirement of problem1 is to perform LPC analysis on the given speech file `vowel_iy_100hz.wav` and the vowel `/i/` I recorded in `lab2` and cross synthesize by using one's $e[n]$ but the other's LPC coefficients.

The code is as below:

```
[s1,fs1]=audioread('vowel_iy_100hz.wav');
[s2,fs2]=audioread('i.wav');

framesize=320;
frameshift=80;
order=12;
window=hamming(framesize);

sig_hat_vowel=zeros(1,length(s1));
sig_hat_i=zeros(1,length(s2));

[a0_vowel,error0_vowel]=LPC(s1(1:framesize)'.*window',order);
[a0_i,error0_i]=LPC(s2(1:framesize)'.*window',order);
framepad0_vowel=[zeros(1,order),s1(1:framesize)'];
framepad0_i=[zeros(1,order),s2(1:framesize)'];
for n=1:framesize
    for k=1:order
        sig_hat_vowel(n)=sig_hat_vowel(n)+a0_i(k).*framepad0_vowel(order+n-k);
        sig_hat_i(n)=sig_hat_i(n)+a0_vowel(k).*framepad0_i(order+n-k);
    end
end
sig_hat_vowel(1:framesize)=sig_hat_vowel(1:framesize)+error0_vowel;
sig_hat_i(1:framesize)=sig_hat_i(1:framesize)+error0_i;


for n=2:(length(s1)-framesize)/frameshift

    frame_vowel=s1((n-1)*frameshift+1:(n-1)*frameshift+framesize)'.*window';

    [a1,error1]=LPC(frame_vowel,order);

    frame_i=s2((n-1)*frameshift+1:(n-1)*frameshift+framesize)'.*window';

    [a2,error2]=LPC(frame_i,order);

    framepad_vowel=[zeros(1,order),frame_vowel];
    framepad_i=[zeros(1,order),frame_i];
    for i=(n-1)*frameshift+1:(n-1)*frameshift+framesize
            for k=1:order
                sig_hat_vowel(i)=sig_hat_vowel(i)+a2(k).*framepad_vowel(order+
(i-(n-1)*frameshift)-k);
                %disp(sig_hat_vowel(i))
                sig_hat_i(i)=sig_hat_i(i)+a1(k).*framepad_i(order+(i-(n-
1)*frameshift)-k);

            end
    end
    sig_hat_vowel((n-1)*frameshift+1:(n-
1)*frameshift+framesize)=sig_hat_vowel((n-1)*frameshift+1:(n-
1)*frameshift+framesize)+error1;
    sig_hat_i((n-1)*frameshift+1:(n-1)*frameshift+framesize)=sig_hat_i((n-
1)*frameshift+1:(n-1)*frameshift+framesize)+error2;
end

figure
subplot(4,1,1);
```

```
plot(1/fs1:1/fs1:length(s1)/fs1,s1);
title('original vowel iy 100hz');
xlabel('time(s)');

subplot(4,1,2);
plot(1/fs1:1/fs1:length(s1)/fs1,sig_hat_vowel);
title('cross synthesized vowel iy 100hz');
xlabel('time(s)');

subplot(4,1,3);
plot(1/fs1:1/fs1:length(s1)/fs1,s2(1:length(s1)));
title('original vowel i');
xlabel('time(s)');

subplot(4,1,4);
plot(1/fs1:1/fs1:length(s1)/fs1,sig_hat_i(1:length(s1)));
title('cross synthesized vowel i');
xlabel('time(s)');
```

The result is as below:

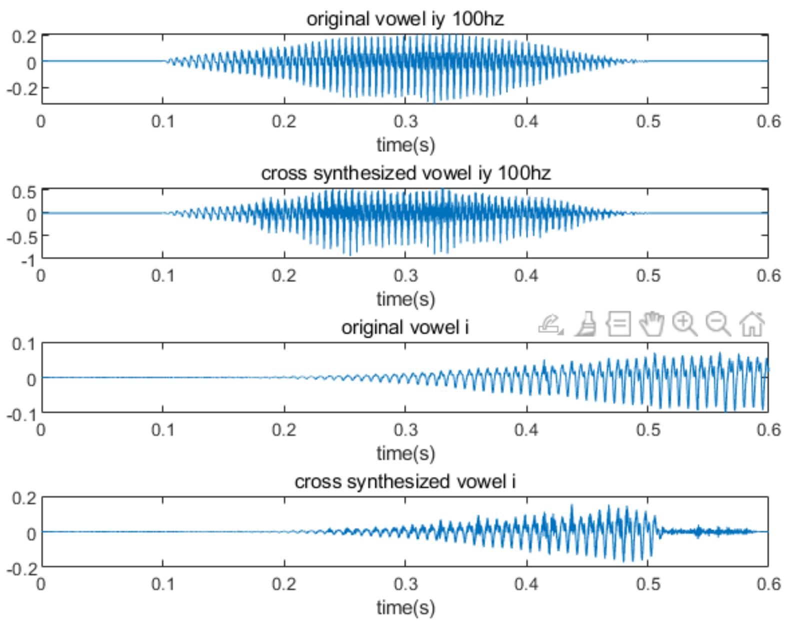


Fig.8 cross synthesized vowels

From the above pictures, what can be known is that the given speech file `vowel_iy_100hz.wav` and the vowel `/i/` I recorded in `lab2` have almost the same coefficient α_k. And the amplitude of cross synthesized vowel is larger than that of original speech signal, which means the energy of cross synthesized vowel is larger. Note the total samples of `vowel_iy_100hz.wav` is less than that of the vowel `/i/` I recorded in `lab2`, so I just do LPC analysis for the first `length(vowel_iy_100hz)` samples for vowel `/i/` I recorded in `lab2`. It is obvious that the latter part of cross synthesized vowel `/i/` only contains the error signal because the coefficient of `vowel_iy_100hz.wav` is zero in the latter part.

3. Conclusion

Our main harvest in this experiment is to master the principle of linear predictive and the method to calculate the coefficient α_k. Besides, we analyze the error signal, resynthesized speech signal and their spectrum effects.