# Lab6 The Cepstrum and Homomorphic Speech Processing

张旭东 12011923

## 1. Introduction

The generation of speech is represented by source and filter models. That means the vocal cord vibration is regarded as the excitation source(denoted by $E(n)$) and the vocal tract is regarded as a filter $H(n)$. The speech signal is obtained by convolution of the two in time domain. In order to analyze each one, a method called deconvolution is generated.

Cepstrum calculation is essentially homomorphic processing, which is a method of deconvolution. It can separate each one of them  and get corresponding signal in time domain. We use windows to intercept signals in the time domain for short-time analysis, and use $FFT$, $log$ and $ifft$ to generate the cepstrum signal. Different points of $FFT$ have different effect on real and complex cepstrum result.

## 2. Lab result and analysis

### 2.1 problem1

In problem1, we are asked to write a MATLAB program to compute the complex and real cesptra of the finite duration signal. The steps to compute complex and real cesptra are shown in `Fig 1`.

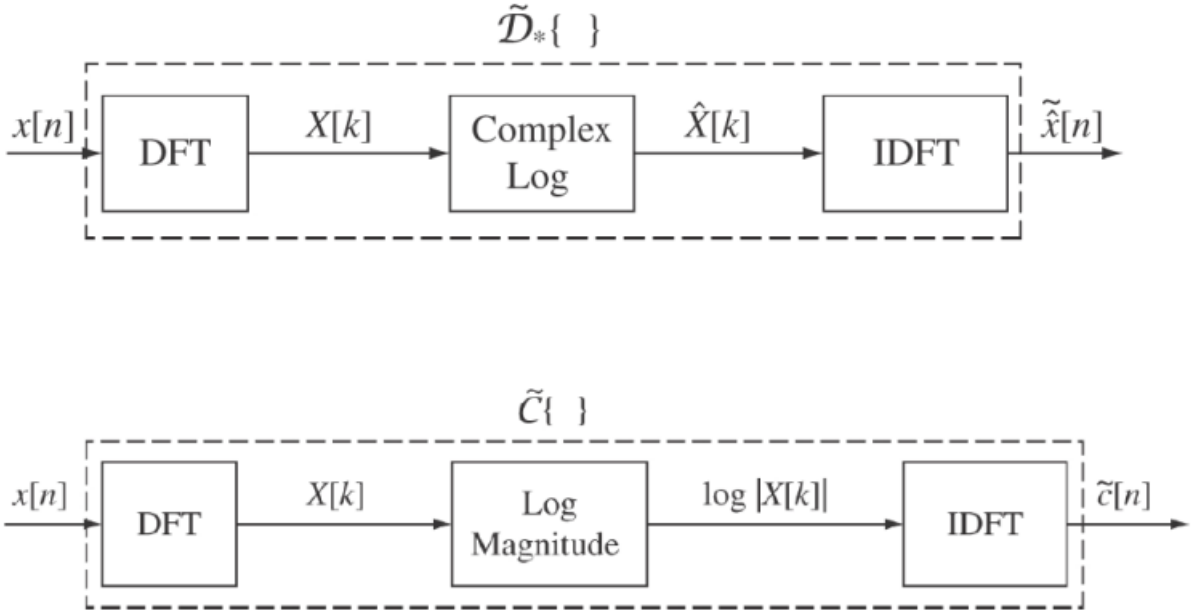



**Fig.1 steps to compute complex and real cesptra**

The first steps of both of them are to calculate $DFT$ of $x[n]$. For complex cesptra, what is needed to do is to calculate the complex $log$ of $X[k]$. The complex log, denoted by $\hat{X}[k]$,should satisfy the phase continuity condition. For real cesptra, what is needed to do is to calculate the $log$ magnitude, $log|X[k]|$. The last steps of both of them are to do inverse $DFT$.

The code of `Cepstrum` is as below:

```
function [ccepstrum,rcepstrum]=Cepstrum(x,nfft)
    X=fftshift(fft(x,nfft));
    theta=unwrap(angle(X));
    X_log=complex(log(abs(X)),theta);
    ccepstrum=ifftshift(ifft(ifftshift(X_log),nfft));
    rcepstrum=ifftshift(ifft(ifftshift(log(abs(X))),nfft));
end
```

The function `unwrap` expands the radian phase in the vector. Whenever the jump between successive phase angles is greater than or equal to $\pi$, the unwrap shifts the phase Angle by increasing it by an integer multiple of $\pm 2\pi$ until the jump is less than $\pi$, which makes the complex $log$ satisfy the phase continuity condition. The function `complex(a,b)` create a complex number, denoted by $a + bj$, which generate the result of complex $log$.

Then the function is used to compute the complex and real cepstra of $x_1[n] = \delta[n] + 0.85\delta[n-100], 0 \leq n \leq 100$. According to our lecture, the cepstrum for $x_1[n] = \delta[n] + 0.85\delta[n-N_p]$ is:

$$\hat{x}[n] = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}\alpha^k}{k}\delta[n-kN_p] \tag{1}$$

The code for plotting the signal, the complex cepstrum, and the real cepstrum om a series of three plots is as below:

```
x=[1 zeros(1,99) 0.85];
nfft=4096;
[ccepstrum,rcepstrum]=Cepstrum(x,nfft);
subplot(3,1,1);
stem(0:100,x);
title('x1[n]');
xlabel('n');
subplot(3,1,2);
stem(-nfft/2:1:nfft/2-1,ccepstrum);
title('complex cepstrum nfft=4096 using our method');
xlabel('qrefuency')
subplot(3,1,3);
stem(-nfft/2:1:nfft/2-1,rcepstrum);
title('real cepstrum nfft=4096 using our method');
xlabel('qrefuency')

xhat=ifftshift(cceps(x,nfft));
x_new=[x zeros(1,nfft-length(x))];
[y,ym]=rceps(x_new);
figure
subplot(3,1,1);
stem(0:100,x);
title('x1[n]');
xlabel('n');
subplot(3,1,2);
stem(-nfft/2:1:nfft/2-1,xhat);
title('complex cepstrum nfft=4096 using MATLAB function');
xlabel('qrefuency')
```

```
subplot(3,1,3);
stem(-nfft/2:1:nfft/2-1,ifftshift(y));
title('real cepstrum nfft=4096 using MATLAB function');
xlabel('qrefuency')
```

The result using our method is shown in `Fig 2`.

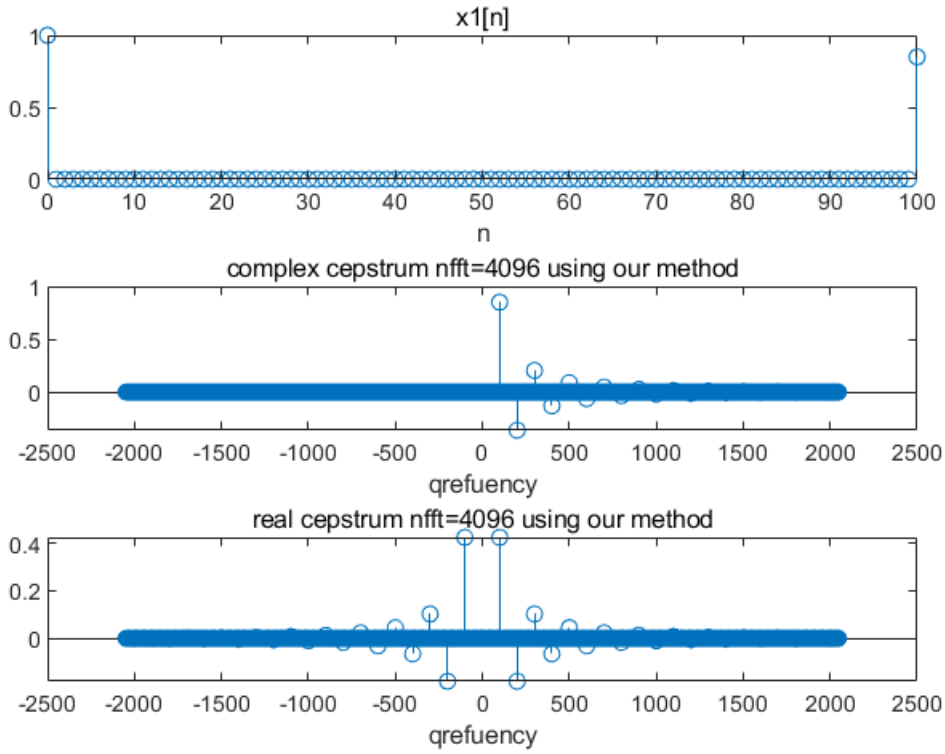



**Fig.2 result with FFTsize=4096**

The result using  MATLAB function is shown in `Fig 3`.

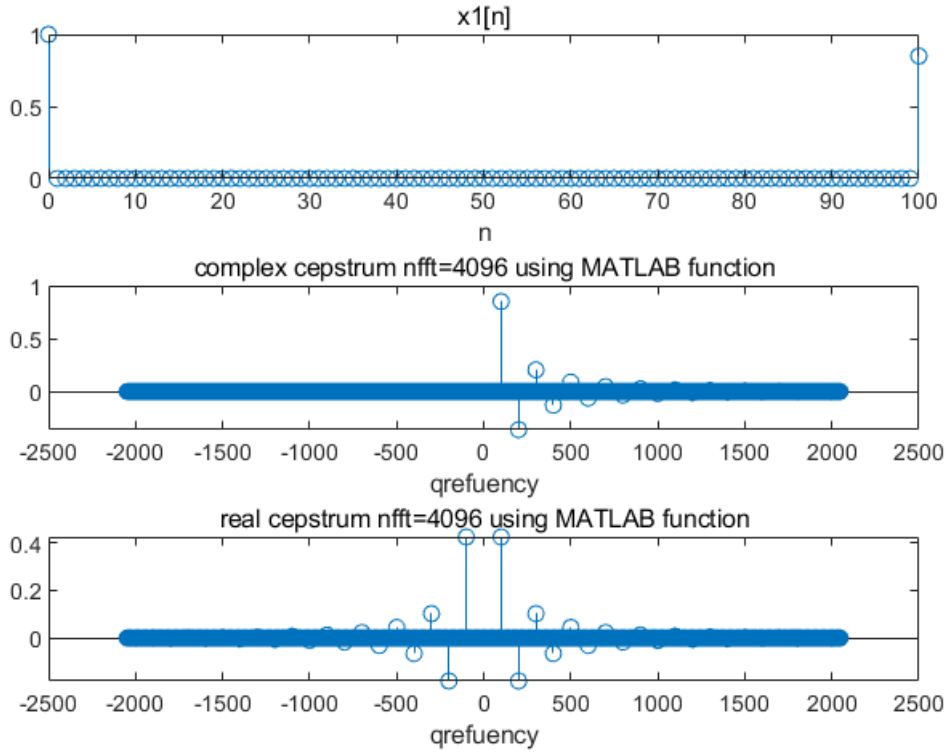
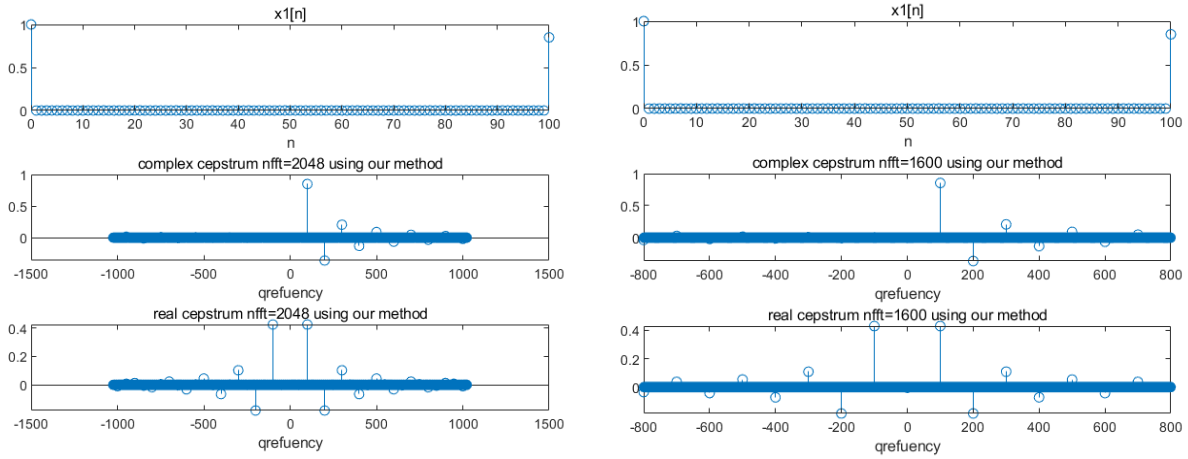



**Fig.3 result with FFTsize=4096 using MATLAB function**

The result using our method is inline with the theory and the result using MATLAB function.

Then we change the value of $nfft$ to analyze the result.



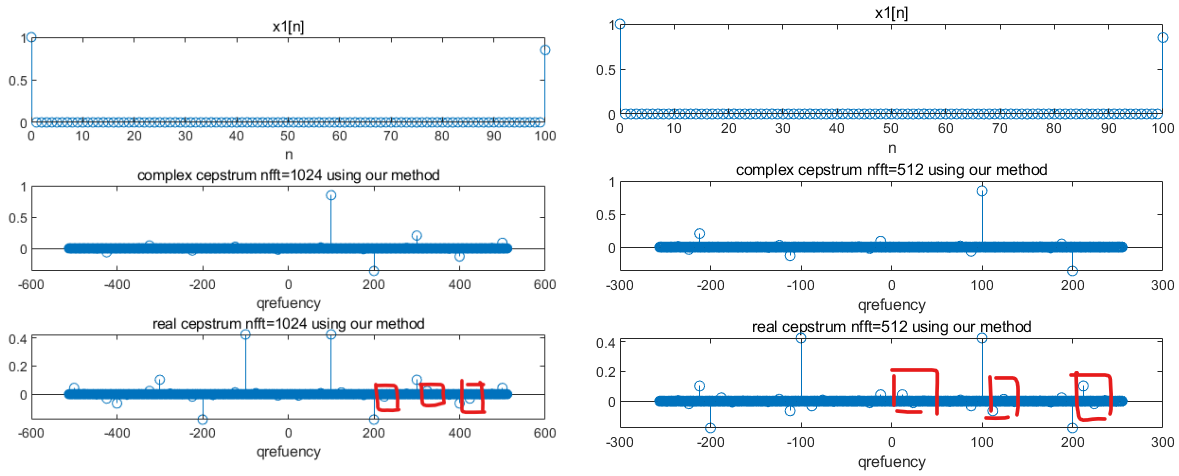



**Fig.4 result of different FFTsize**

As we can see, with the point of $FFT$ is 4096 or 2048, there is no aliasing phenomenon. However, while the point of $FFT$ is 1024 or 512, aliasing appears(marked in red). Some small pulses should appear later, instead of having appeared between several correct pulses in multiples of 100.

According to `formula(1)`, $\hat{x}[n]$ is non-zero for $n = kN_p$ for $1 \leq m \leq \infty$, so aliasing will produce complex cepstrum values that are out of sequence. The non-zero values of $\hat{x}[n]$ are at the position whose index is the positive multiple of $N_p$.

## 2.2 problem2

In problem2, we are asked to compute the cepstrum of the signal $x_2[n] = a^n u[n], |a| < 1$. The value of $a$ should be selected and the point of $FFT$ is chosen to minimize the aliasing effects. The steps to calculate cepstrum have been shown in `Fig 1`.

According to the steps, we can derivate the formula for the cepstrum as follow:

$$\begin{aligned} x_2[n] &= a^n u[n] \\ X_2(z) &= \frac{1}{1 - az^{-1}} \\ \hat{X}_2(z) = log(X_2(z)) = -log(1 - az^{-1}) &= -\sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n} (-a)^n z^{-n} \end{aligned} \tag{2}$$

For $x_3[n] = \frac{a^n}{n} u[n-1]$, the cepstrum for is is:

$$\hat{X}_3(z) = \sum_{n=1}^{\infty} \frac{a^n}{n} z^{-n} \tag{3}$$

From `formula (2) and (3)`, it is known that $x_2[n]$ and $x_3[n]$ have similar form.

The function is as below:

```
function cepstrum=problem1_function(a,N,nfft)
        n=0:N-1;
        x=a.^n;
        [cepstrum,rcepstrum]=Cepstrum(x,nfft);
end
```

Then we use $a = 0.5$ or $0.9$, $N = 10, 200$ and $nfft = 10, 16, 200, 256$ to test the function. The code for testing is as below:

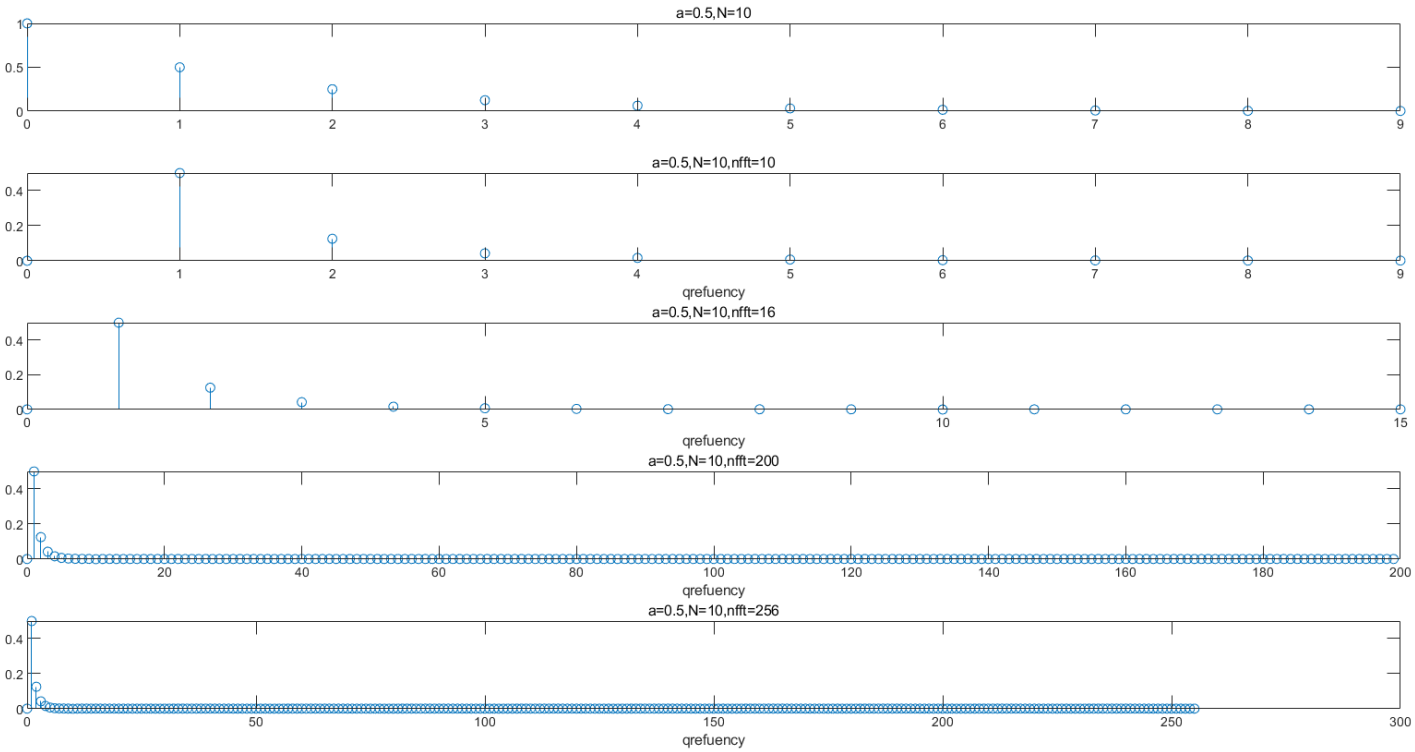
```
cepstrum11=problem1_function(0.5,10,10);
cepstrum12=problem1_function(0.5,10,16);
cepstrum13=problem1_function(0.5,10,200);
cepstrum14=problem1_function(0.5,10,256);
figure
subplot(5,1,1);
stem(0:9,(0.5).^(0:9));
title('a=0.5,N=10');
subplot(5,1,2);
stem(0:9,ifftshift(cepstrum11));
title('a=0.5,N=10,nfft=10');
xlabel('qrefuency');
subplot(5,1,3);
stem(0:15,ifftshift(cepstrum12));
title('a=0.5,N=10,nfft=16');
xlabel('qrefuency');
subplot(5,1,4);
stem(0:199,ifftshift(cepstrum13));
title('a=0.5,N=10,nfft=200');
xlabel('qrefuency');
subplot(5,1,5);
stem(0:255,ifftshift(cepstrum14));
title('a=0.5,N=10,nfft=256');
xlabel('qrefuency');
```

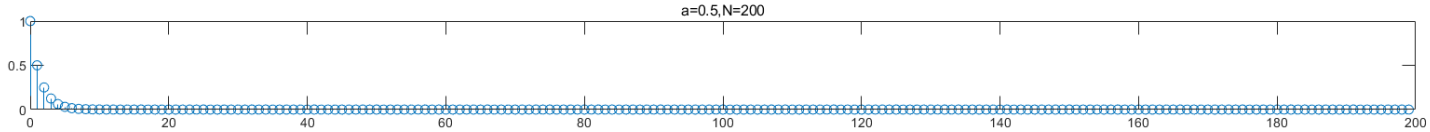
```
cepstrum21=problem1_function(0.5,200,10);
cepstrum22=problem1_function(0.5,200,16);
cepstrum23=problem1_function(0.5,200,200);
cepstrum24=problem1_function(0.5,200,256);
figure
subplot(5,1,1);
stem(0:199,(0.5).^(0:199));
title('a=0.5,N=200');
subplot(5,1,2);
stem(0:9,ifftshift(cepstrum21));
title('a=0.5,N=200,nfft=10');
xlabel('qrefuency');
subplot(5,1,3);
stem(0:15,ifftshift(cepstrum22));
title('a=0.5,N=200,nfft=16');
xlabel('qrefuency');
subplot(5,1,4);
stem(0:199,ifftshift(cepstrum23));
title('a=0.5,N=200,nfft=200');
xlabel('qrefuency');
subplot(5,1,5);
stem(0:255,ifftshift(cepstrum24));
title('a=0.5,N=200,nfft=256');
xlabel('qrefuency');


cepstrum31=problem1_function(0.9,10,10);
cepstrum32=problem1_function(0.9,10,16);
cepstrum33=problem1_function(0.9,10,200);
cepstrum34=problem1_function(0.9,10,256);
figure
subplot(5,1,1);
stem(0:9,(0.9).^(0:9));
title('a=0.9,N=10');
subplot(5,1,2);
stem(0:9,ifftshift(cepstrum31));
```

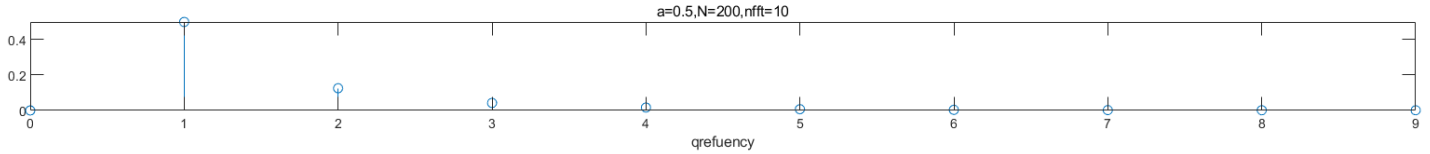
```
title('a=0.9,N=10,nfft=10');
xlabel('qrefuency');
subplot(5,1,3);
stem(0:15,ifftshift(cepstrum32));
title('a=0.9,N=10,nfft=16');
xlabel('qrefuency');
subplot(5,1,4);
stem(0:199,ifftshift(cepstrum33));
title('a=0.9,N=10,nfft=200');
xlabel('qrefuency');
subplot(5,1,5);
stem(0:255,ifftshift(cepstrum34));
title('a=0.9,N=10,nfft=256');
xlabel('qrefuency');

cepstrum41=problem1_function(0.9,200,10);
cepstrum42=problem1_function(0.9,200,16);
cepstrum43=problem1_function(0.9,200,200);
cepstrum44=problem1_function(0.9,200,256);
figure
subplot(5,1,1);
stem(0:199,(0.9).^(0:199));
title('a=0.9,N=200');
subplot(5,1,2);
stem(0:9,ifftshift(cepstrum41));
title('a=0.9,N=200,nfft=10');
xlabel('qrefuency');
subplot(5,1,3);
stem(0:15,ifftshift(cepstrum42));
title('a=0.9,N=200,nfft=16');
xlabel('qrefuency');
subplot(5,1,4);
stem(0:199,ifftshift(cepstrum43));
title('a=0.9,N=200,nfft=200');
xlabel('qrefuency');
subplot(5,1,5);
stem(0:255,ifftshift(cepstrum44));
```

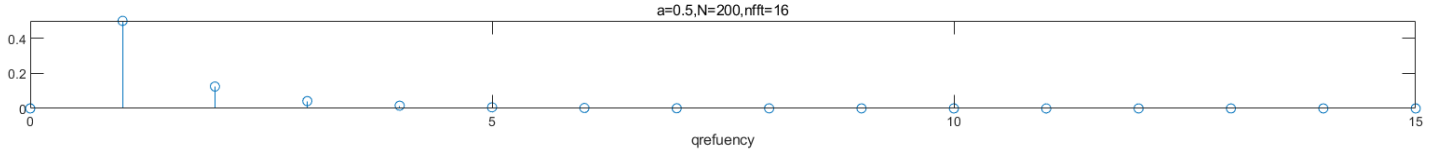
```
title('a=0.9,N=200,nfft=256');
xlabel('qrefuency');
```

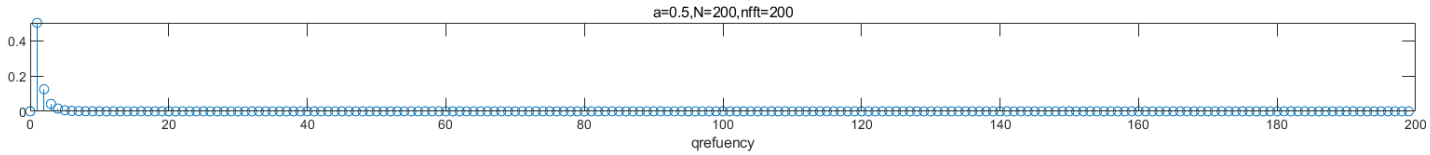
The result is shown in `Fig 5`

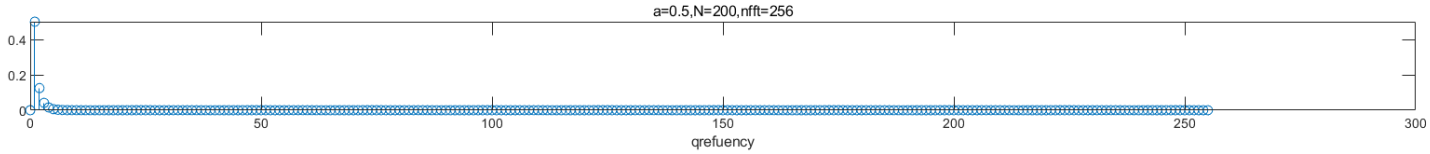


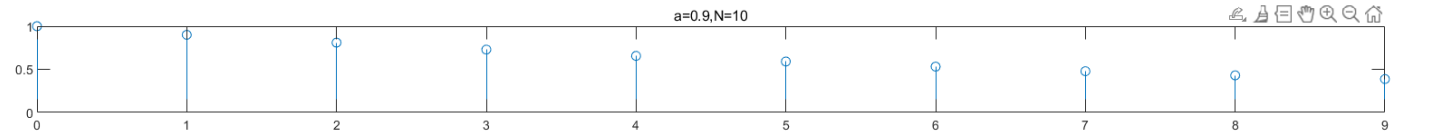

a=0.5,N=200

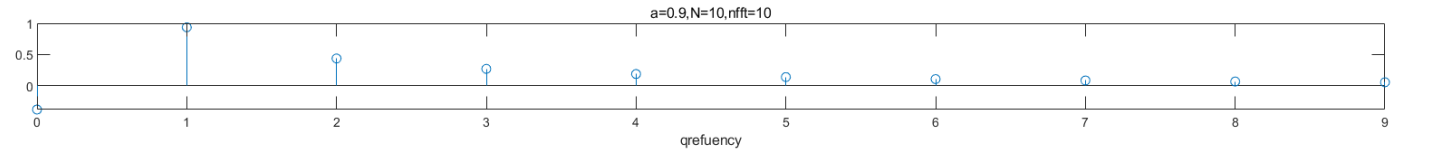

a=0.5,N=200,nfft=10
qrefuency

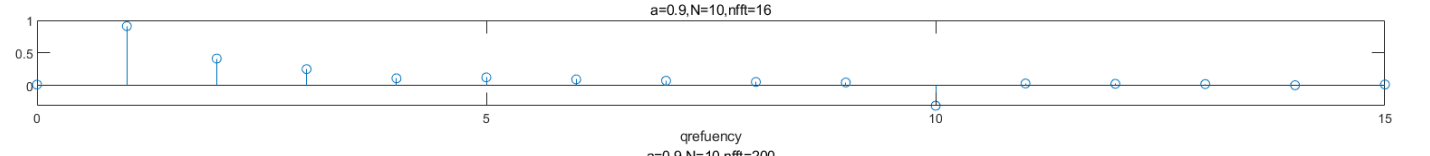

a=0.5,N=200,nfft=16
qrefuency

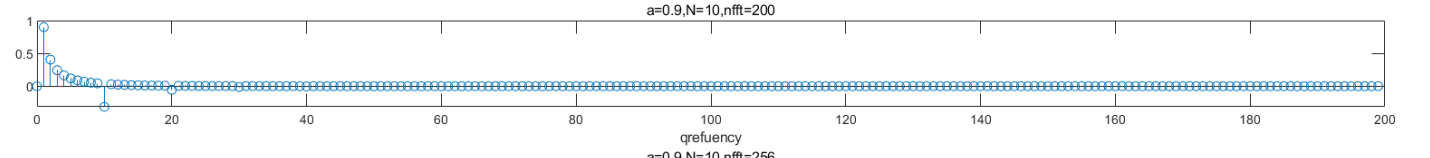

a=0.5,N=200,nfft=200
qrefuency

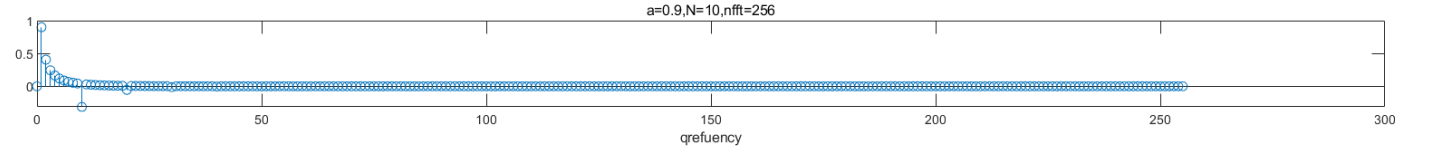

a=0.5,N=200,nfft=256
qrefuency


a=0.9,N=10


a=0.9,N=10,nfft=10
qrefuency


a=0.9,N=10,nfft=16
qrefuency


a=0.9,N=10,nfft=200
qrefuency


a=0.9,N=10,nfft=256
qrefuency

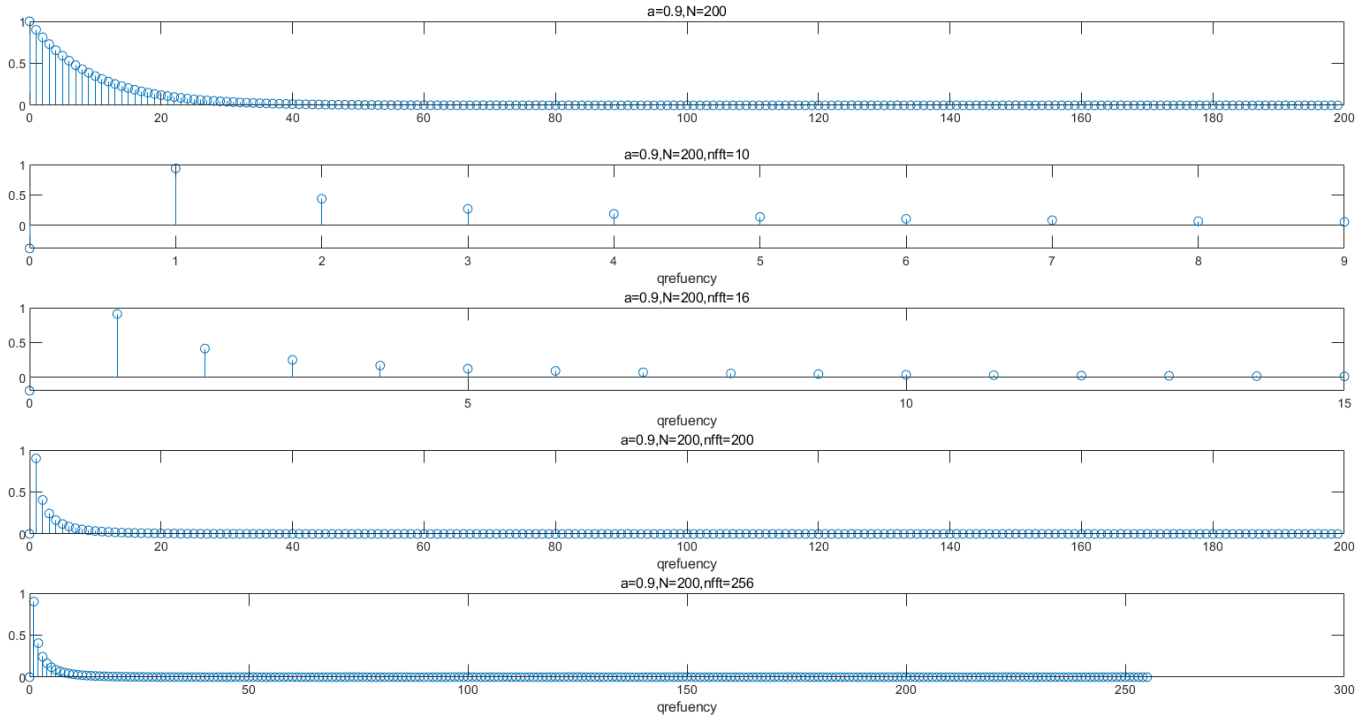



**Fig.5 result of different a,N,nfft**

The results are in line with theory. However, for $a = 0.9$ and $N = 10$ and $200$, there are different phenomena. When $nfft = 10$ and $16$, the cepstrum has value at $n = 0$. While $nfft = 200$ and $256$, the cepstrum has zero-value at $n = 0$. That is because there is aliasing phenomena when $nfft = 10$ and $16$. We can prove that if we use $nfft$ point of $FFT$, in cepatrum for samples $n > \frac{nfft}{2}$, which will cause aliasing. Increasing $nfft$ will tend to mitigate the effect by allowing more of the impulses to be at correct position and ensuring that the aliased samples will have smaller amplitudes due to the fall-off.

## 2.3 problem3

In problem3, we are asked to analyze a speech signal, using the voiced and unvoiced part to analyze its cepstrum. What's more, we need to use the low quefrency liftered log magnitude magnitude to analyze.

The overall process is shown in `Fig 6`.

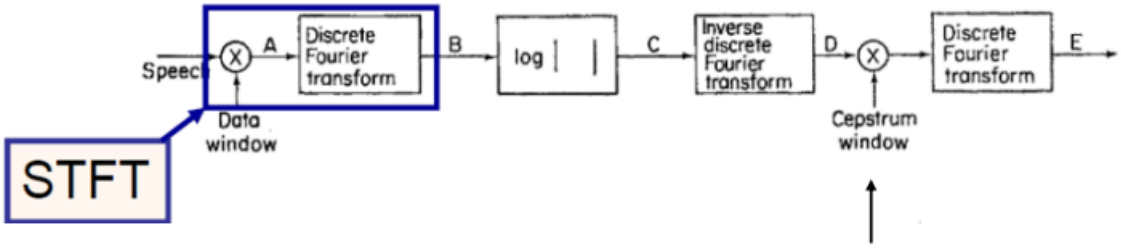



**Fig.6 overall process**

For data window, Hamming Window is used. For cepstrum window, the `formula (4)` is used.

$$\tilde{l}_{lp}[n] = \begin{cases} 1 & 0 \leqslant n < n_{co} \\ 0.5 & n = n_{co} \\ 0 & n_{co} < n < N - n_{co} \\ 0.5 & n = N - n_{co} \\ 1 & N - n_{co} < n \leq N - 1 \end{cases} \tag{4}$$

In testing, the voiced section as the starting at sample 13000 in the file and of duration 400 samples. The unvoiced section as starting at sample 3400 in the file and of duration 400 samples. And $n_{co}$ is set to 50.

The code of testing is as below:

```
[s,fs]=audioread('test_16k.wav');
duration=400;
ham=hamming(duration);
voiced_speech=s(13000:1:13400-1);
unvoiced_speech=s(3400:1:3800-1);
windowed_voiced_speech=(voiced_speech.*ham)';
windowed_unvoiced_speech=(unvoiced_speech.*ham)';

%log magnitude spectrum
magnitude_spectrum_voice=fftshift(fft(windowed_voiced_speech,duration));
magnitude_spectrum_unvoice=fftshift(fft(windowed_unvoiced_speech,duration));

%real cepstrum
[ccepstrum_voice,rcepstrum_voice]=Cepstrum(windowed_voiced_speech,duration);
```

```
[ccepstrum_unvoice,rcepstrum_unvoice]=Cepstrum(windowed_unvoiced_speech,durati
on);

%low quefrency liftered log magnitude spectrum
voiced_low=ifftshift(ccepstrum_voice);
voiced_low(51:349)=0;
voiced_low(50)=0.5*voiced_low(50);
voiced_low(350)=0.5*voiced_low(350);
voiced_low_liftered=fftshift(fft(voiced_low,duration));
unvoiced_low=ifftshift(ccepstrum_unvoice);
unvoiced_low(51:349)=0;
unvoiced_low(50)=0.5*unvoiced_low(50);
unvoiced_low(350)=0.5*unvoiced_low(350);
unvoiced_low_liftered=fftshift(fft(unvoiced_low,duration));



figure
subplot(4,1,1);
plot(13000:13400-1,windowed_voiced_speech);
title('voiced section using hamming window');
subplot(4,1,2);
plot((-duration/2:(duration-
1)/2)/duration*fs,log(abs(magnitude_spectrum_voice)));
title('voiced log magnitude spectrum');
subplot(4,1,3);
plot(-duration/2:(duration-1)/2,rcepstrum_voice);
title('voiced real cepstrum');
subplot(4,1,4);
plot((-duration/2:(duration-1)/2)/duration*fs,voiced_low_liftered);
title('voiced low quefrency liftered log magnitude spectrum');
figure
subplot(4,1,1);
plot(3400:3800-1,windowed_unvoiced_speech);
title('unvoiced section using hamming window');
subplot(4,1,2);
```

```
plot((-duration/2:(duration-
1)/2)/duration*fs,log(abs(magnitude_spectrum_unvoice)));
title('unvoiced log magnitude spectrum');
subplot(4,1,3);
plot(-duration/2:(duration-1)/2,rcepstrum_unvoice);
title('unvoiced real cepstrum');
subplot(4,1,4);
plot((-duration/2:(duration-1)/2)/duration*fs,unvoiced_low_liftered);
title('unvoiced low quefrency liftered log magnitude spectrum');
```

It is considered that the `ccepstrum` generated by function `Cepstrum` has done the operation of $fftshift$. So, for the low quefrency, we need to do operation of $fftshiftagain$ to recover it to the original result firstly. Then the low quefrency components is located in the center of samples and the liftered will be achieved.

The result is shown in `Fig 7` and `Fig 8`.

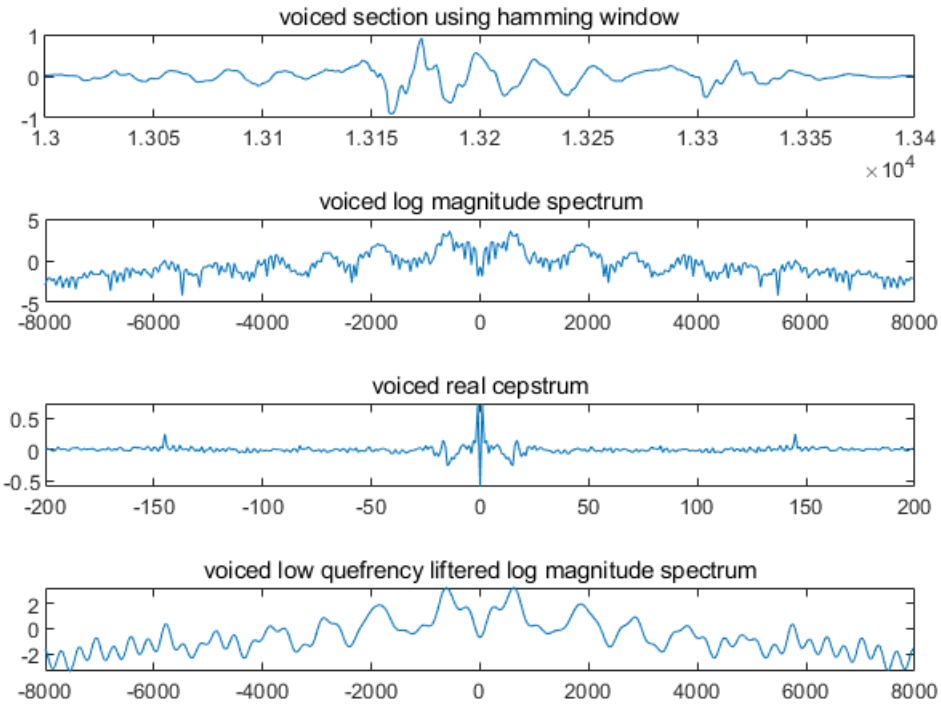



**Fig.7 voiced part result**

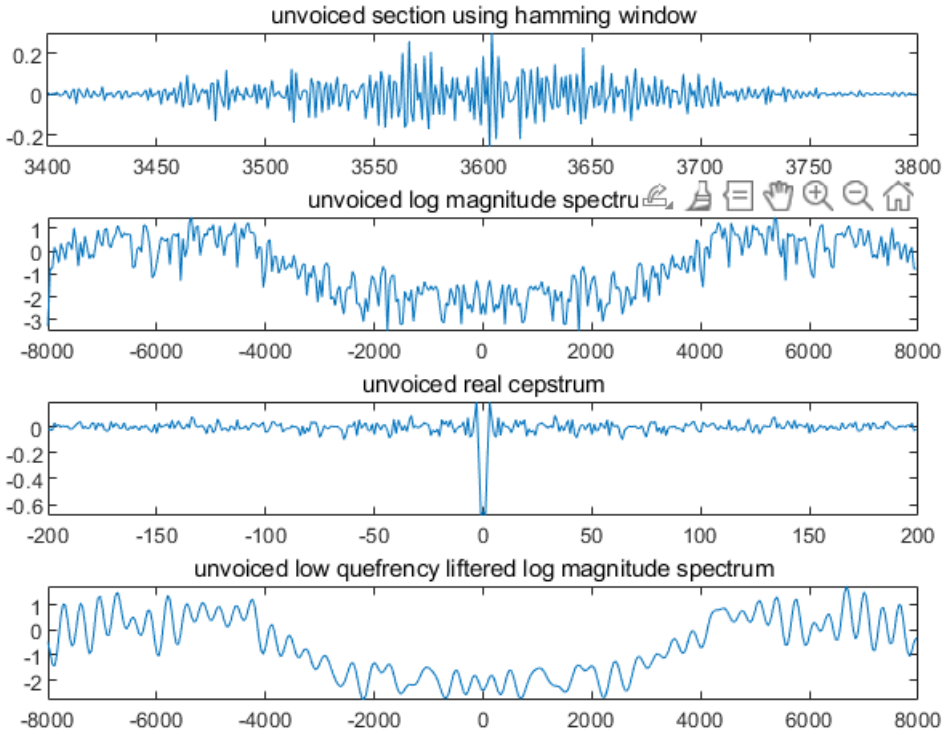



**Fig.8 unvoiced part result**

It is obvious that the log magnitude is more smooth and changes slowly after liftering. According to the `formual (4)`, with $n_{co}$ increasing, the low quefrency liftered log magnitude spectrum is more closer to original logarithmic amplitude. What's more, the envelope of the signal is more clear after liftering. Also, there is a sharp pulse in the real cepstrum of voiced section while the real cepstrum of unvoiced section doesn't have sharp pulse.

## 3. conclusion

Our main harvest in this experiment is to learn to compute complex and real cepstra and compare the cepstra of voiced and unvoiced segments. Cepstrum calculation is essentially homomorphic processing, which is a method of deconvolution. It can separate each one of vocal cord vibration and vocal tract and get corresponding signal in time domain. However, the point of $FFT$ has an effect on cepstrum. In order to minimize aliasing, use as large a value of point of $FFT$ as possible.