Lab 5: System, Convolution and Filter

Author	Name & Student ID : 王卓扬（12112907）、冯彦捷（12010825）

Introduction

1. Using MATLAB to generate a white noise.
2. Using MATLAB to generate Butterworth filters.
3. Using MATLAB to calculate power spectral density.
4. Using MATLAB to generate a speech-shaped noise.
5. Using MATLAB to adjust the SNR.
6. Using MATLAB to normalize the energy of the signal.
7. Using MATLAB to load and play .wav files.
8. Using MATLAB to extract speech envelopes.

Problem 1. *Generate a speech-shaped noise (SSN), and plot the spectra of the speech signal and SSN (e.g., use Matlab function 'psd' or 'pwelch', or other power spectral density estimation functions).*

Solution

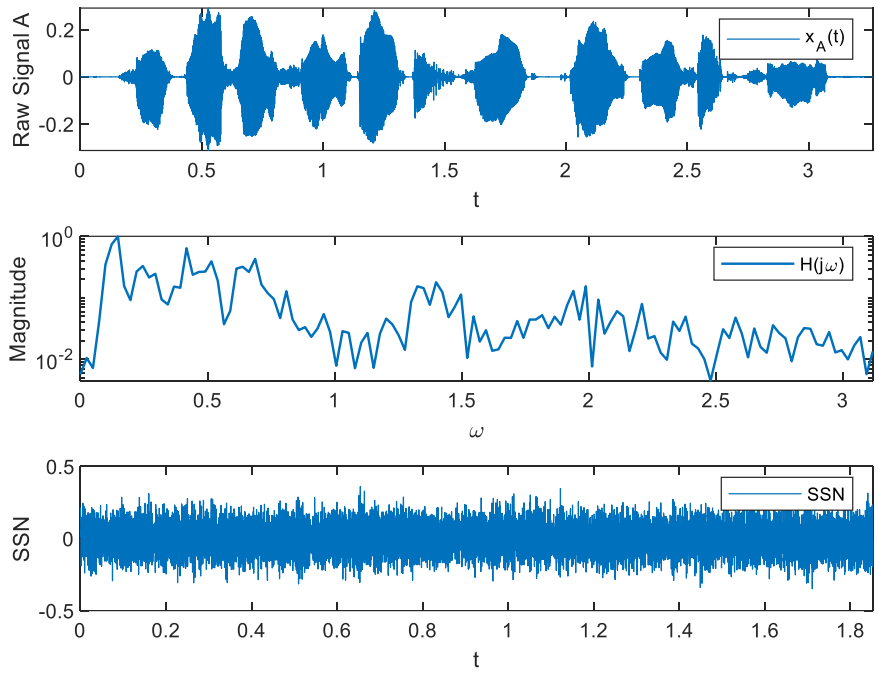


Fig (a)

Analysis

The result is shown in Fig (a). The first subplot is the raw signal from *C_01_01.wav*. The second one is the frequency response of the filter generated by *fir2*. And the last one is pattern of a piece of the target signal SSN.

Problem 2. *Read a speech signal x(t), adjust the SNR (x(t) to the above SSN) to -5 dB, let $y = x + SSN$, and normalize the energy of y in relative to x(t), i.e., modify the energy of y so that it equals to the energy of x.*

Solution

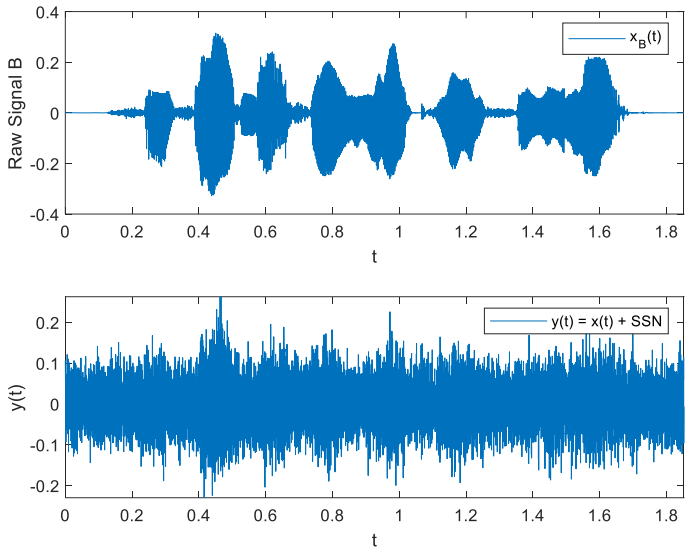


Fig (b)

Analysis

In this problem we take *SSN* in Problem 1, and the raw signal *x(t)* from *C_01_02.wav*. Our target is to make that

$$SNR = 20\log_{10}\frac{||x(t)||}{||SSN||} = -5$$

Which means

$$\frac{||x(t)||}{||SSN||} = 10^{-\frac{1}{4}}$$

So

$$SSN = SSN_{raw} \times \frac{||x(t)||}{||SSN_{raw}||} \times 10^{\frac{1}{4}}$$

Then we add *x(t)* and *SSN* together, and normalize the result *y(t)*'s energy in the similar way.

Problem 3. *Extract speech envelope.*

1) *with 2nd-order low-pass filter and cutoff frequency f_{cut} = 100, 200, and 300 Hz. Plot these three envelope waveforms in one plot, and describe the difference among them.*

Solution

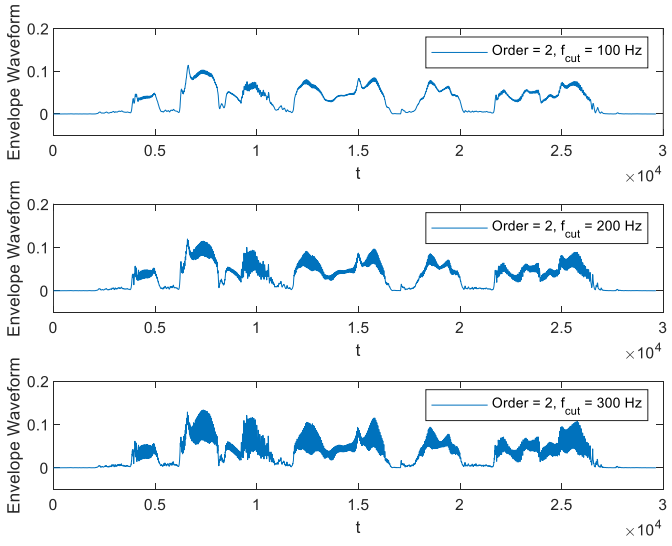


Fig (c)

Analysis

The results are shown in Fig (c). They are the response signals (Envelope waveforms) filtered by different low-pass filters. From the figure we can see that, when the order of the filter is fixed, the higher the cut-off frequency is, the more the high frequency components are left, which eventually lead to the "thick" border.

---

2) *with 2nd and 6th-order low-pass filter and cutoff frequency 200 Hz. Plot these two envelope waveforms in one plot, and describe the difference between them.*

Solution

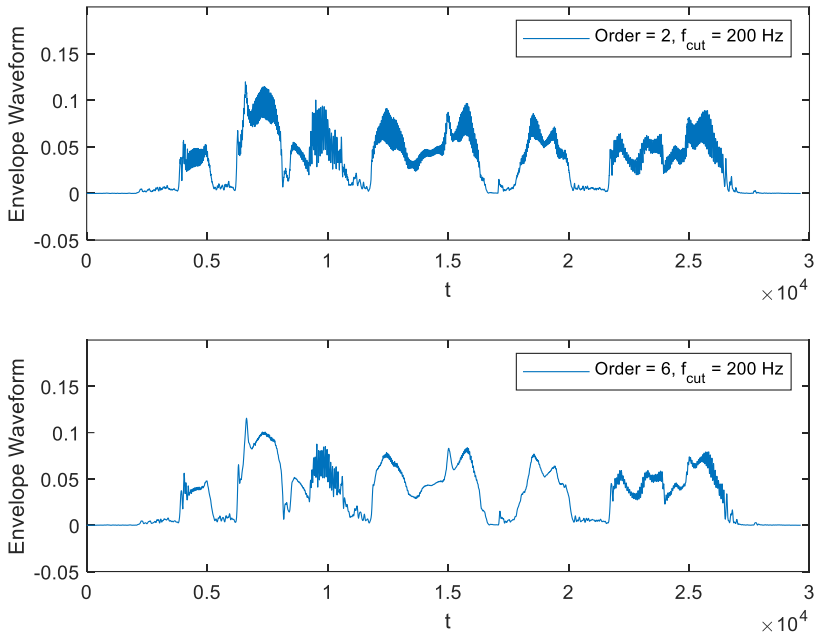


Fig (d)

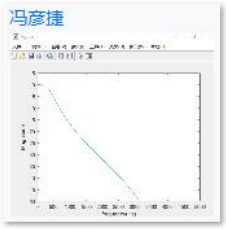
Analysis

The results are shown in Fig (d). They are the response signals filtered by low-pass filters with equivalent cut-off frequency, but different order. Obviously the high-order filter performs better, because it can distinguish the components of different frequency more effectively.

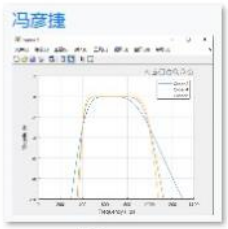
Experience

1. We learnt less about the details of the procedure and we searched for information online.
2. It's excellent to generate various filters we need.
3. We made mistakes when calculating the SNR at beginning. Be careful when multiplying 10 or 20.

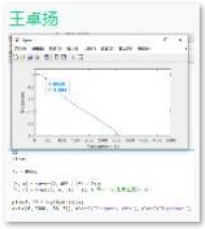
*** The screenshots in lab class:**



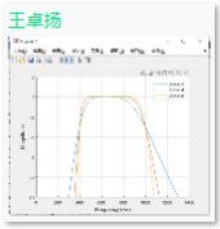
fyj01.png



fyj02.png



wzy01.png



wzy02.png

Score	99

Appendix

Programs
(All-in-one)

```
clear;

%% Load Raw Signals
[y_wav_a, fs_wav_a] = audioread("C_01_01.wav");
[y_wav_b, fs_wav_b] = audioread("C_01_02.wav");
y_wav_a = y_wav_a';
y_wav_b = y_wav_b';

%% Generate Filter
[Pxx, w] = pwelch(repmat(y_wav_a, 1, 10), [], [], 512, fs_wav_a); % Power
Spectral Density
b = fir2(30000, w / (fs_wav_a / 2), sqrt(Pxx / max(Pxx))); % Generate
coefficients
[h, hw] = freqz(b, 1, 128);

%% Obtain SSN for Signal B
N = length(y_wav_b);
noise = 1 - 2 * rand(1, N + length(b) - 1); % White Noise
ssn = filter(b, 1, noise);
ssn = ssn(length(b) : end);

%% Adjust Signal Density
x = y_wav_b;
ssn = ssn / norm(ssn) * norm(x) * 10 ^ (1 / 4);
disp(20 * log10(norm(x) / norm(ssn))); % Test the value
y = x + ssn;
y = y / norm(y) * norm(x); % normalization

%% Generate Low-pass Filters
fs = fs_wav_b;
[b1, a1] = butter(2, 100 / (fs / 2));
[b2, a2] = butter(2, 200 / (fs / 2));
[b3, a3] = butter(2, 300 / (fs / 2));
[b4, a4] = butter(6, 200 / (fs / 2));

x = abs(x);
y1 = filter(b1, a1, x);
y2 = filter(b2, a2, x);
y3 = filter(b3, a3, x);
y4 = filter(b4, a4, x);
```

```
%% Plotting
figure(1);
subplot(3, 1, 1);
plot(linspace(0, length(y_wav_a) / fs_wav_a, length(y_wav_a)), y_wav_a);
xlabel("t"), ylabel("Raw Signal A"), xlim([0, length(y_wav_a) / fs_wav_a]);
legend("x_A(t)");
subplot(3, 1, 2);
semilogy(hw, abs(h), "LineWidth", 1);
xlabel("\omega"), ylabel("Magnitude"), xlim([hw(1), hw(end)]);
legend("H(j\omega)");
subplot(3, 1, 3);
plot(linspace(0, N / fs_wav_b, N), ssn);
xlabel("t"), ylabel("SSN"), xlim([0, N / fs_wav_b]);
legend("SSN");

figure(2);
subplot(2, 1, 1);
plot(linspace(0, length(y_wav_b) / fs_wav_b, length(y_wav_b)), y_wav_b);
xlabel("t"), ylabel("Raw Signal B"), xlim([0, length(y_wav_b) / fs_wav_b]);
legend("x_B(t)");
subplot(2, 1, 2);
plot(linspace(0, length(y) / fs_wav_b, length(y)), y);
xlabel("t"), ylabel("y(t)"), xlim([0, length(y) / fs_wav_b]);
legend("y(t) = x(t) + SSN");

figure(3);
subplot(3, 1, 1);
plot(y1), xlabel("t"), ylabel("Envelope Waveform"), ylim([-0.05, 0.2]);
legend("Order = 2, f_{cut} = 100 Hz");
subplot(3, 1, 2);
plot(y2), xlabel("t"), ylabel("Envelope Waveform"), ylim([-0.05, 0.2]);
legend("Order = 2, f_{cut} = 200 Hz");
subplot(3, 1, 3);
plot(y3), xlabel("t"), ylabel("Envelope Waveform"), ylim([-0.05, 0.2]);
legend("Order = 2, f_{cut} = 300 Hz");

figure(4);
subplot(2, 1, 1);
plot(y2), xlabel("t"), ylabel("Envelope Waveform"), ylim([-0.05, 0.2]);
legend("Order = 2, f_{cut} = 200 Hz");
subplot(2, 1, 2);
plot(y4), xlabel("t"), ylabel("Envelope Waveform"), ylim([-0.05, 0.2]);
legend("Order = 6, f_{cut} = 200 Hz");
```