Introduction to Multimedia Homework 2

104062361

陳永恒

Q1. Create FIR filters to filter audio signal

Determine the filters

- There are 3 songs combined into 1 music file. This is clear that they are in 3 separate frequency boundary.
- As the Ideal Filters are provided, I choose Low-pass, High-pass and bandpass filter.

Implement the Filter and convolutions

- Ideal Filter is rectangle function in frequency domain. In theory, we have to apply DFT(frequency domain) and inverse DFT(time domain) in the filter. Lecture notes provides the algorithm implementation for time domain. So I can follow the formula provided and apply convolution.

```
if strcmp(filterName, 'low-pass') == 1
    for n = -floor(N/2)+1: floor(N/2)
    if (n==0) fltr(middle) = 2*fcutoff1;
        else fltr(n+middle) = sin(2*pi*fcutoff1*n)/(pi*n);
        end
    end
elseif strcmp(filterName, 'high-pass') == 1
    …
elseif strcmp(filterName, 'bandpass') == 1
    …
end
```

Implement the Filter and convolutions

- Before apply convolution, window function are necessary to make the impulse response finite (a sin function goes on infinitely in +ve and -ve directions.)

```
if strcmp(windowName,'Blackman') == 1
    for n = 1: N
        fltr(n)=fltr(n)*( (0.42)-0.5*cos((2*pi*n)/(N-1))+0.08*cos((4*pi*n)/(N-1)));
    end
end
```

- Call the convolution function `conv_imple.m`, apply the filter on input audio.

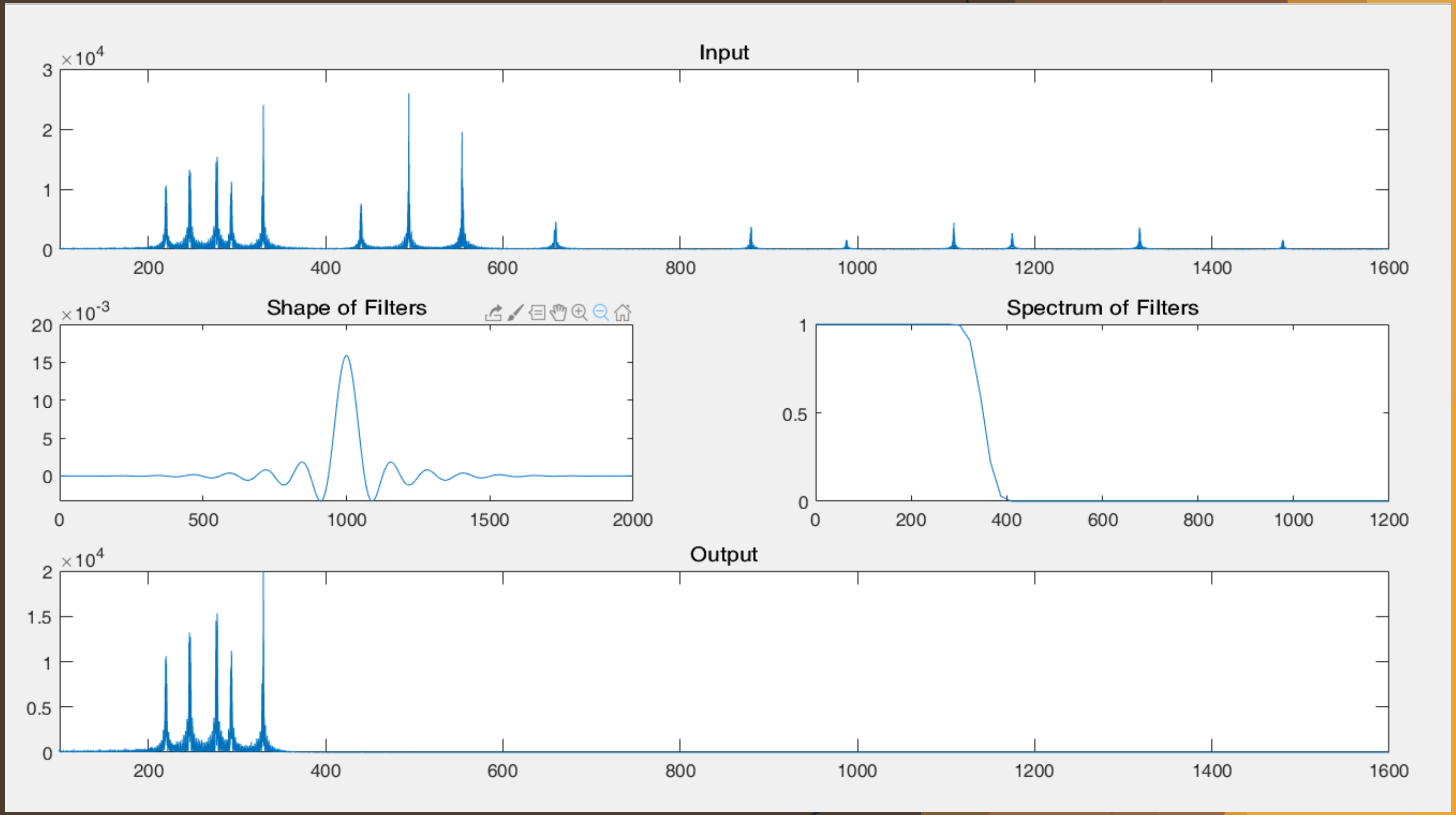
Apply the filter to separate the mixed song

	Low-pass Filter Low frequency song	High-pass Filter High frequency song	Bandpass Medium boundary song
N(Window size)	1999	1999	1999
Cutoff frequency (Hz)	350	800	400-700

- If N is higher, the higher possibility to give a sharp cut for separating the song.
- To get the cutoff frequency, it is necessary to listen the output audio and repeat the process to get the best result.

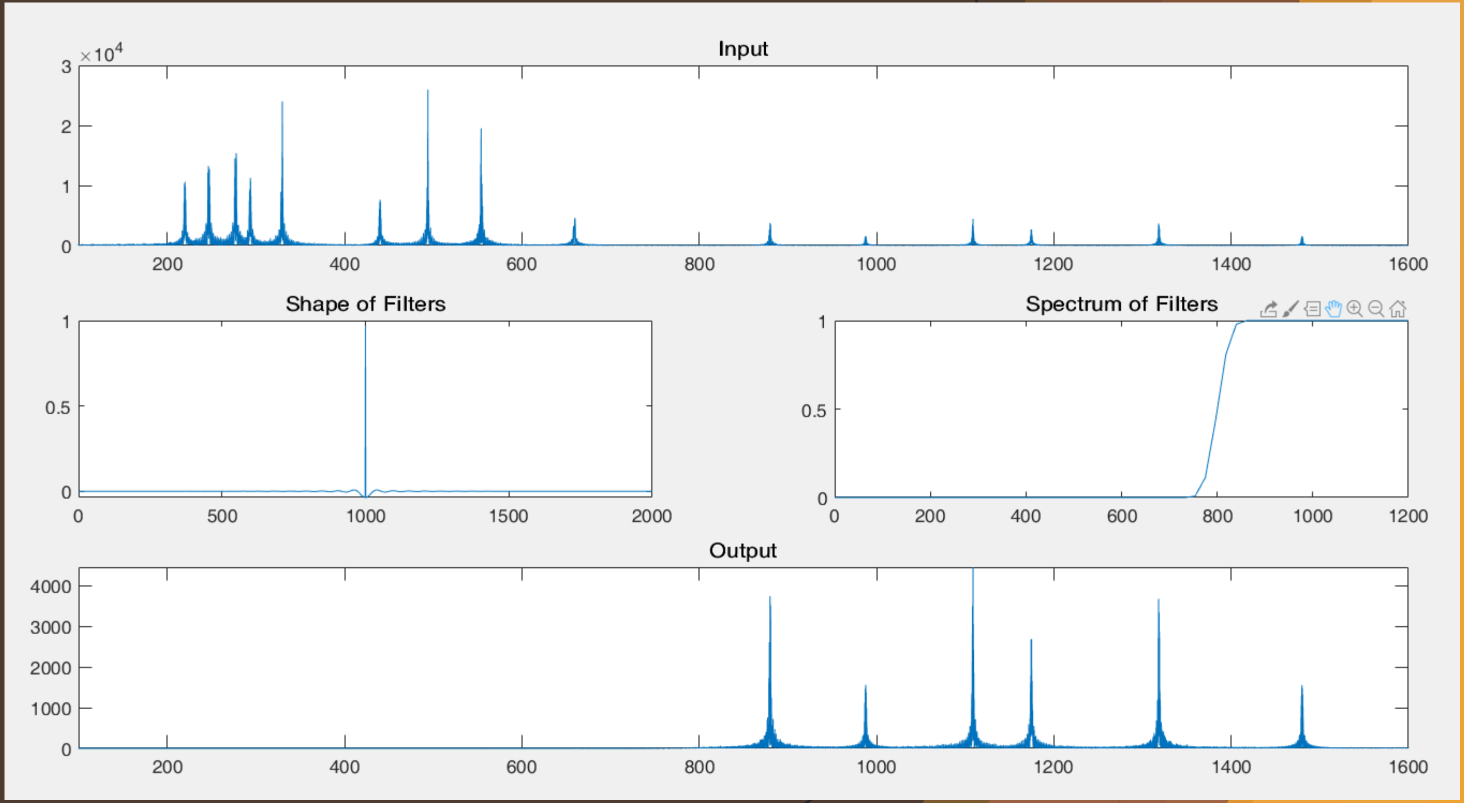
Compare spectrum and shape of filters-low pass

- The spectrum of filter is correctly. It filters out the signal above 350Hz.
- The shape of filters meets my expectation as it is the same the figure given in lecture notes.



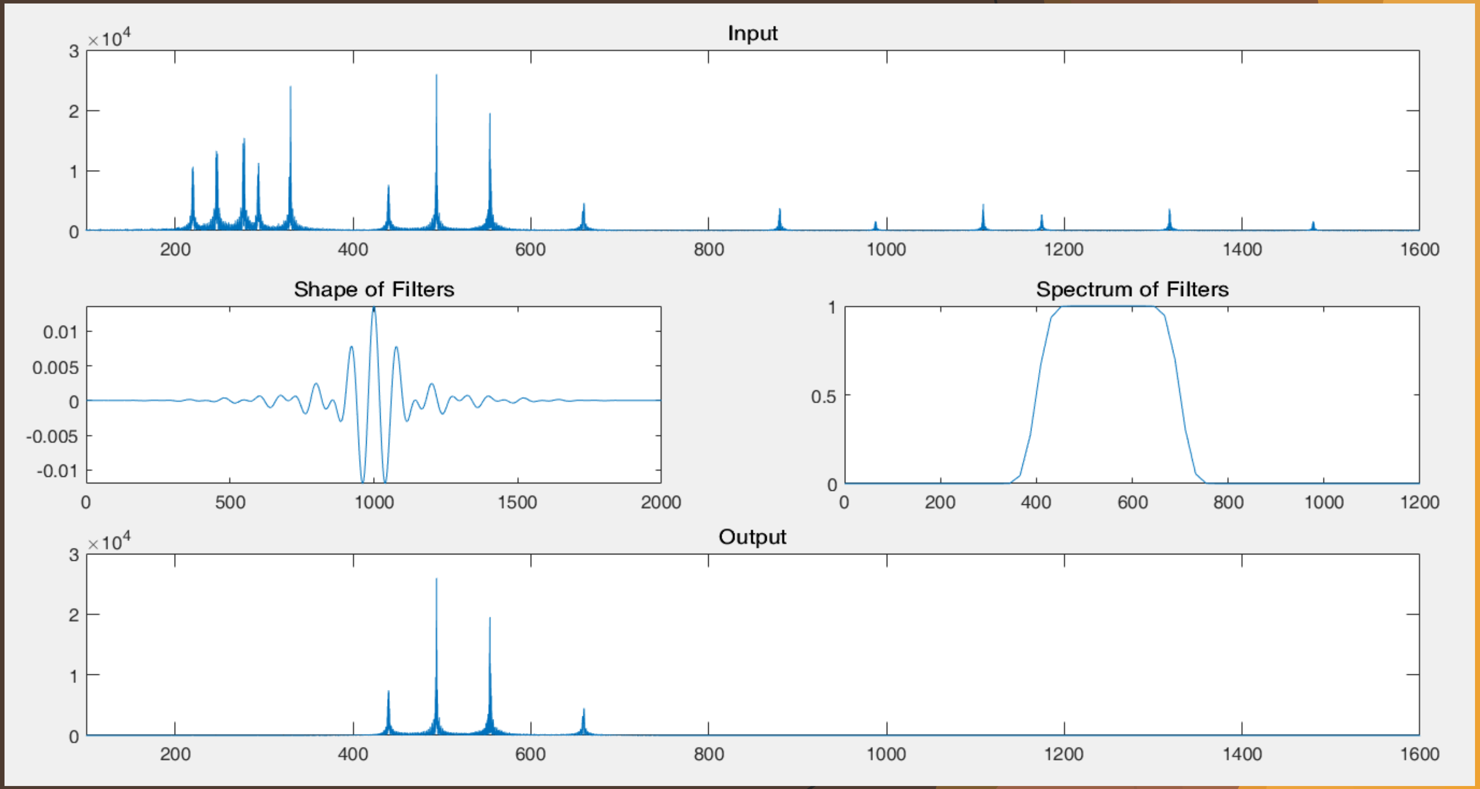
Compare spectrum and shape of filters-high pass

- The spectrum of filter is correctly. It filters out the signal below 800Hz.
- I don’t know if the shape of filters(amplitude vs time) is reasonable because applying filter convolution to audio, audio in the section in 1000 will be remained, and minimize in others.



Compare spectrum and shape of filters-bandpass

- The spectrum of filter is correctly. It remains the signal between 400 and 750Hz.
- The shape of filter: Mentioned in preious page, I don't know if the shape of filters(amplitude vs time) is reasonable.



Implement the one/multiple fold echo

```
% one-fold echo
[row, col] = size(outputSignal);
oneEcho = zeros(row, 1);
for n = 1: row
    if (n-3200 > 1)
        oneEcho(n, 1) = outputSignal(n, 1) + 0.8*outputSignal(n-3200, 1);
    end
end
```

- fs=44100Hz, time difference of echo and original signal = 3200/fs = 0.07s
- Adding signal before 0.07s is so weak, nearly no effect.
- Reference: http://mirlab.org/jang/books/audiosignalProcessing/filterApplication_chinese.asp?title=11-1%20Filter%20Applications%20(%C2o%AAi%BE%B9%C0%B3%A5%CE)

Implement the one/multiple fold echo

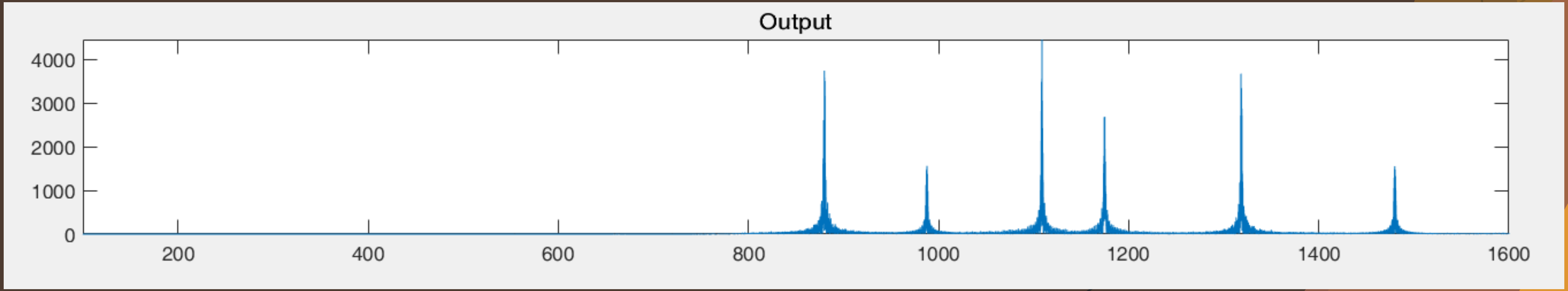
```
% multiple-fold echo
multiEcho = zeros(row, 1);
for n = 1: row
    if (n-3200 > 1)
        multiEcho(n, 1) = outputSignal(n, 1) + 0.8*multiEcho(n-3200, 1);
    end
end
```

- `0.8*multiEcho(n-3200, 1)` brings the previous signal that can give a really strong effect.

Reducing sampling rate

- Low-pass (~350Hz): No difference
- Bandpass (400~750Hz): No difference
- High-pass (800Hz~): Some high pitch signals are filtered out.
- From the graph, we can see that there are highest frequency signals component is 1500Hz. By Nyquist's Sampling Theorem, the sampling rate must at least twice highest frequency component ~= 3000Hz > 2000Hz.
- So, the result within expectation.

Note: 2kHz music only works with Windows Media Player, MATLAB audioplayer(), player() and iTunes(Windows)



Q2. Audio dithering and noise shaping

Implement Bit Reduction

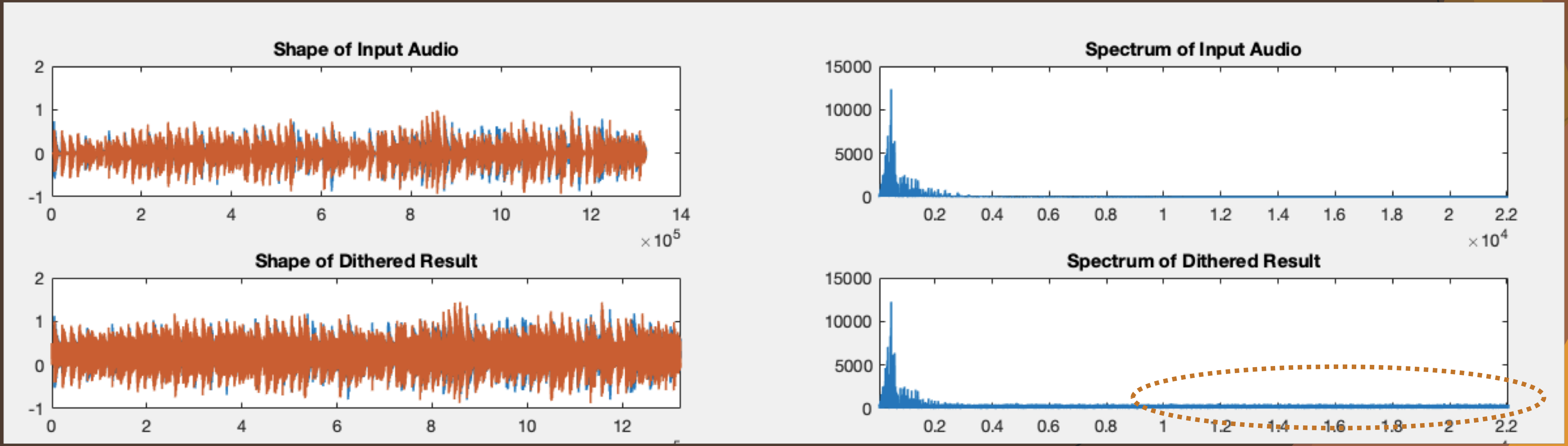
```
input8bits = uint8( (input + 1)/2 * 255 );
audiowrite('Tempest_8bit.wav', input8bits, fs, 'BitsPerSample', 8);
```

- The raw data(16-bit) is normalized between -1 to 1. Apply the function above which can transfer data into 0 to 255. uint8() means data are in 8-bit.

Implement Audio Dithering

```
input8bits_dither = input8bits_nor + rand(size(input8bits_nor))/5;
```

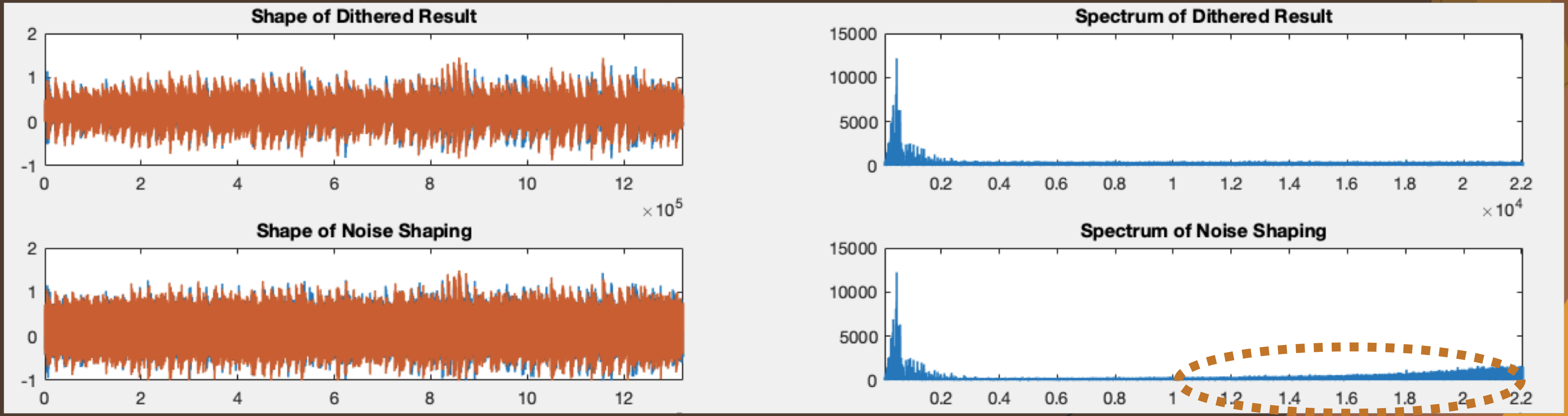
- Audio Dithering is add random noise. We can see that the amplitude of audio is larger than the input significantly. (Shape of Audio)
- Magnitude in high frequency is also larger than the input. (Spectrum of Audio)



Implement Noise Shaping

```
for C = 1: col
    for R = 1: row
        if (R-1 >= 1)
            shapingOutput(R, C) = shapingOutput(R, C)+
                    c*(input8bits_nor(R-1,C)-shapingOutput(R-1,C));
        end
    end
end
```

- Redistribution the quantization error so that the noise is concentrated in the higher frequencies.



Implement Low-pass Filter

- Same way in Q1. However, this is stereo audio(two sound play together). I have modified the code to fulfill this situation.

Implement of Audio Limitation

- This is hard clipping. As the data has been normalized during audioread() i.e. data should be -1<y[x]<1. After audio dithering, random sharping, applying low-pass filter will be excess the boundary. Every number less than -1 will assign as -1, larger than 1 will be assign as 1.

Implement Normalization

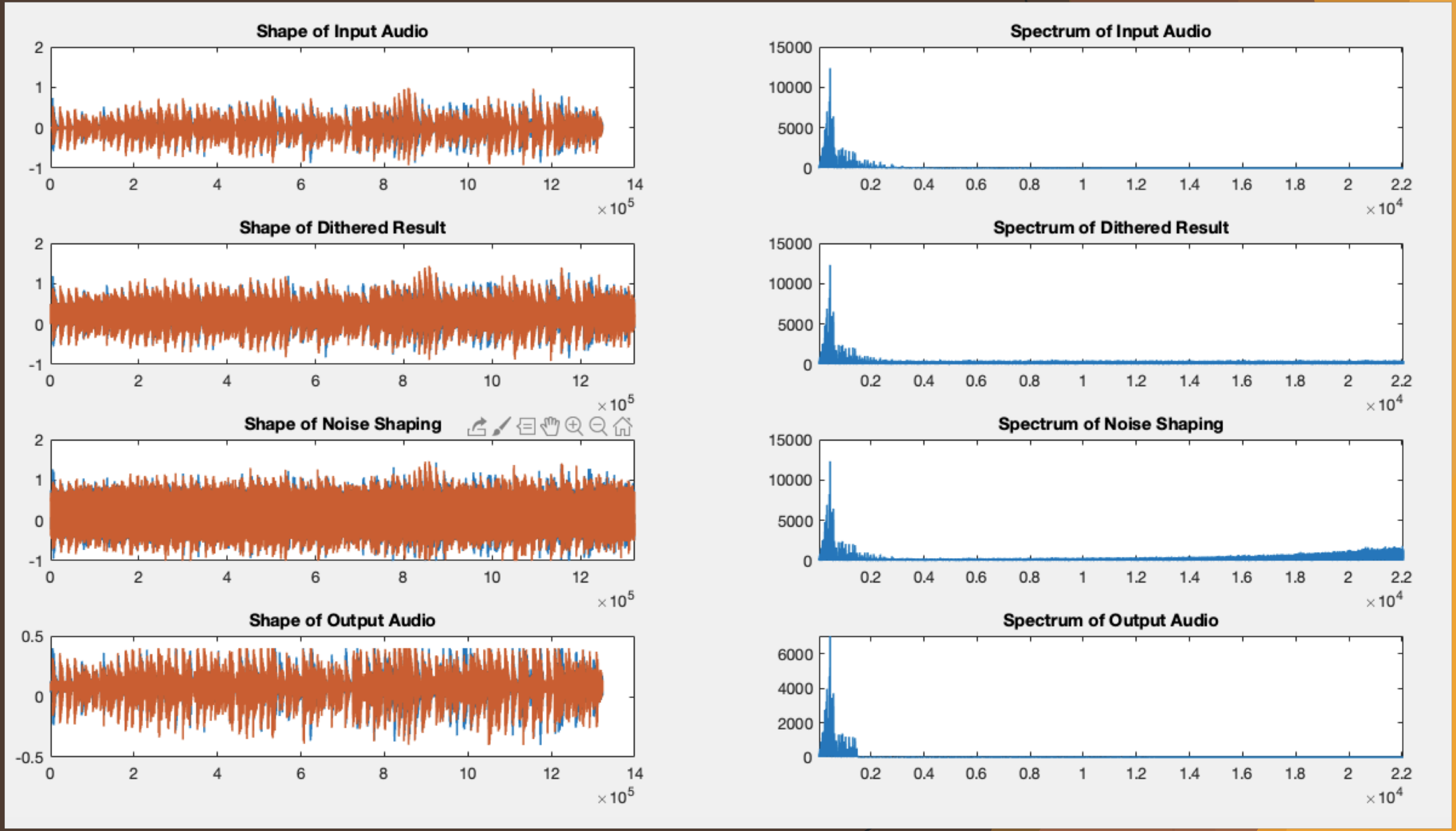
```
desireMax = .5;
for C = 1: col
   ampM = max(limitingOutput(:,C));
   normalizeOutput(:,C) = limitingOutput(:,C)*(desireMax/ampM);
end
```

- Find the highest amplitude sample, determine the rate of amplify and raise all samples with amplify rate.

Final Result

- In Spectrum of Output, thanks for Low-pass Filter, the high frequency noise is filtered out.
- Normalization used to set the Maximum amplitude(0.5 in this case).

Note: All spectrum in Q2 has been chipped because amplitude of 0Hz is extremely high, which is not helpful for analysis.


Shape of Input Audio
Spectrum of Input Audio
Shape of Dithered Result
Spectrum of Dithered Result
Shape of Noise Shaping
Spectrum of Noise Shaping
Shape of Output Audio
Spectrum of Output Audio
$\times 10^5$
$\times 10^4$