多媒體技術概論 AS2

105060016 謝承儒

1. Create your own FIR filters to filter audio signal

實作步驟

1. 讀入 HW2_Mix.wav，得到震幅數值的 inputAudio(1-D Array)和採樣頻率 f_s。
2. 將 inputAudio、f_s、f_{cutoff}、N、windowName、filterName 放入 myFilter，做以下步驟：
   (1) 初始化：Normalize f_{cutoff} and w_{cutoff} so that π is equal to Nyquist angular frequency

$$f_{cutoff} = \frac{f_{cutoff}}{f_s}$$

$$w_{cutoff} = 2 * \pi * f_{cutoff}$$

   (2) 有 low-pass、high-pass、bandpass、bandstop 四種，根據 filterName 做出不同的 filter，得到 outputFilter
   每個 filter 的公式如下：

Type of filter	$h_{ideal}(n),\ n \neq 0$	$h_{ideal}(0)$
Low-pass	$\frac{\sin(2\pi f_c n)}{\pi n}$	$2f_c$
High-pass	$-\frac{\sin(2\pi f_c n)}{\pi n}$	$1 - 2f_c$
Bandpass	$\frac{\sin(2\pi f_2 n)}{\pi n} - \frac{\sin(2\pi f_1 n)}{\pi n}$	$2(f_2 - f_1)$
Bandstop	$\frac{\sin(2\pi f_1 n)}{\pi n} - \frac{\sin(2\pi f_2 n)}{\pi n}$	$1 - 2(f_2 - f_1)$

   (3) 將 window function 和 outputFilter 相乘

$$\text{outputFilter} = \text{outputFilter} * \text{window}$$

   window function 如下：

- Blackmann windowing function
  - $w(n) = 0.42 + 0.5\cos\left(\frac{2\pi n}{N-1}\right) + 0.08\cos\left(\frac{4\pi n}{N-1}\right)$

   (4) 將 inputAudio 和 outputFilter 做 convolution，即可得到 outputAudio

$$\text{outputAudio}[n] = \sum_{k=0}^{n} \text{inputAudio}[k] * \text{outputFilter}[n-k]$$

3. 利用 resample()對 outputAudio 做 resample，將取樣頻率降至 2000Hz，得到 outputAudio2K
4. 將 outputAudio, f_s, delay, gain 放進 OneFoldEcho，裡片步驟如下：
   (1) Delay 決定 outputAudio 要 shift 多少

$$\text{outputAudio}_{\text{delay}}[n] = \text{outputAudio}[n - f_s * delay]$$

(2) Gain 決定$\text{outputAudio}_{\text{delay}}[n]$要以多少倍加上

$$\text{outputAudioOneFoldEcho} = \text{outputAudio} + \text{gain} * \text{outputAudio}_{\text{delay}}$$

(3) 對outputAudioOneFoldEcho做 Normalize，將已經超過 1~-1 的範圍縮至 1~-1

5. 將 outputAudio, f_s, delay, gain 放進 MutiFoldEcho，裡片步驟如下 ：

(1) Delay 決定 outputAudioMutiFoldEcho 要 shift 多少

$$\text{outputAudioMutiFoldEcho}_{\text{delay}}[n] = \text{outputAudioMutiFoldEcho}[n - f_s * delay]$$

(2) Gain 決定$\text{outputAudio}_{\text{delay}}[n]$要以多少倍加上

$$\text{outputAudioOneFoldEcho} = \text{outputAudio} - \text{gain} * outputAudioMutiFoldEcho\_delay\ [n]$$

(3) 對outputAudioOneFoldEcho做 Normalize，將已經超過 1~-1 的範圍縮至 1~-1

結果圖片

為了方便觀察，跟時間有關的圖只會顯示 1000 個 time step(約 0.0227 秒)，

跟頻率有關的圖只會顯示 0Hz~1200Hz

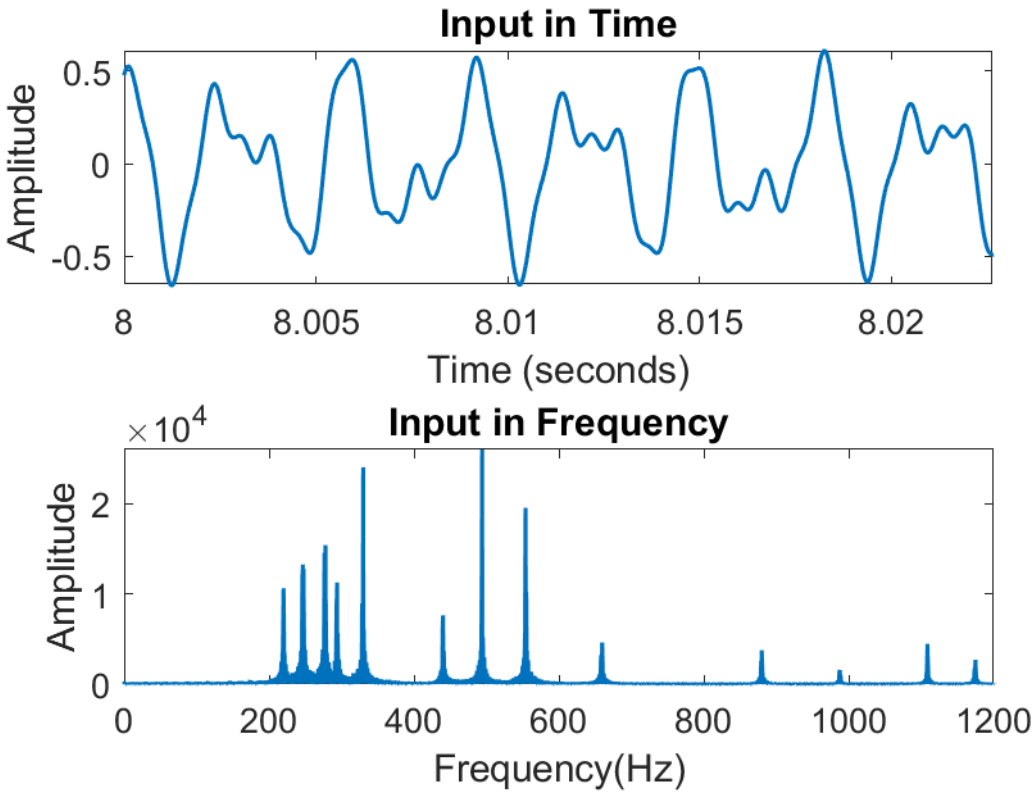


圖 1 input 的 Shape 和 Spectrum

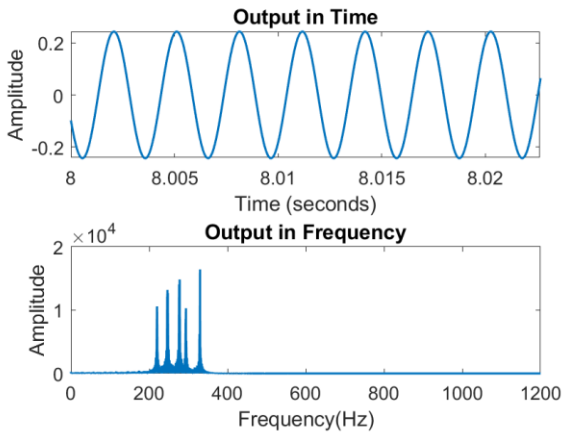


圖 2 low-pass filter 後 input 的 Shape 和 Spectrum

F_{cutoff}=350Hz

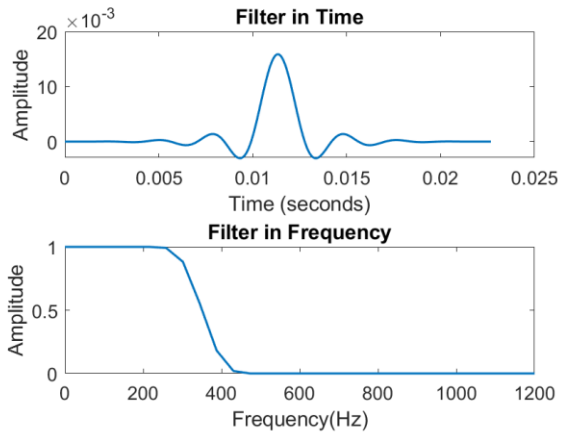


圖 3 low-pass filter 的 Shape 和 Spectrum

F_{cutoff}=350Hz

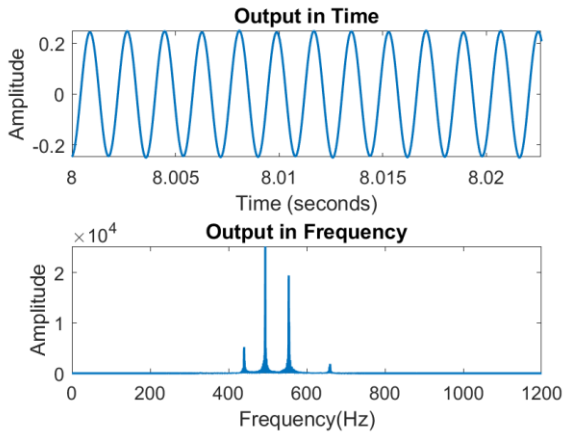


圖 4 bandpass filter 後 input 的 Shape 和 Spectrum

F_{cutoff}=420Hz、650Hz

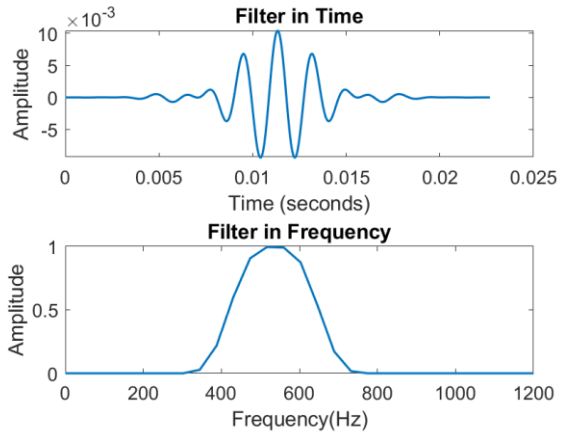


圖 5 bandpass filter 的 Shape 和 Spectrum

F_{cutoff}=420Hz、650Hz

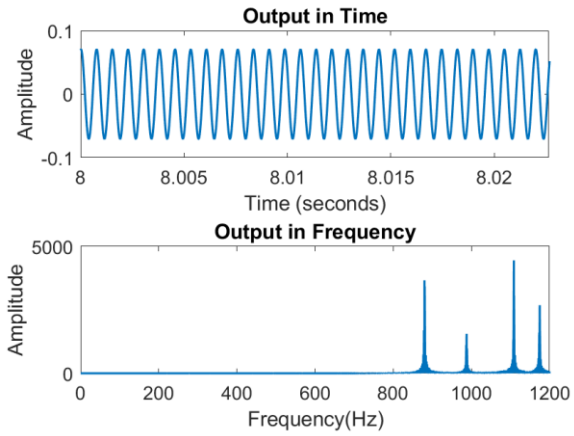


圖 6 high-pass filter 後 input 的 Shape 和 Spectrum

F_{cutoff}=800Hz

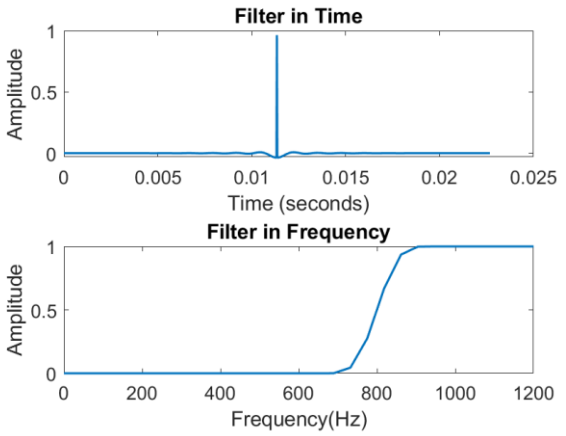


圖 7 high-pass filter 的 Shape 和 Spectrum

F_{cutoff}=800Hz

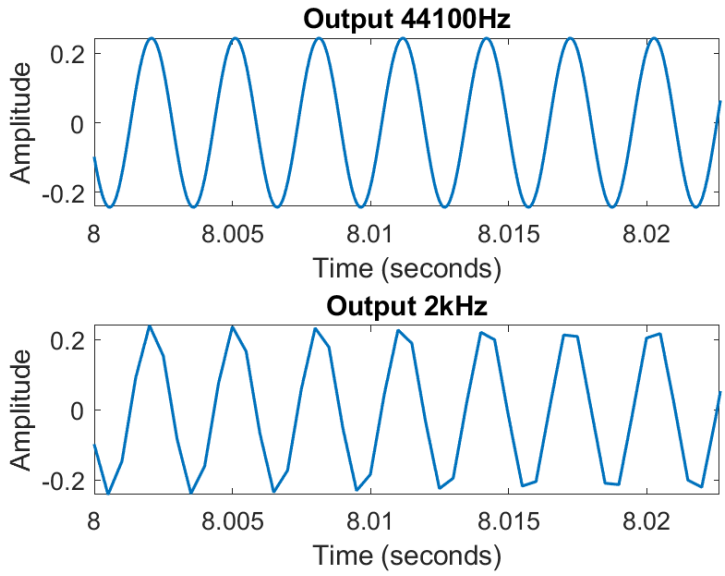


圖 8 對低頻的音檔(圖 2)做 resample 的 Shape 和 Spectrum

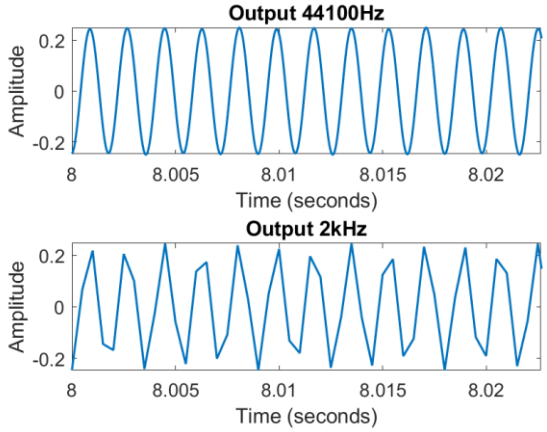


圖 9 對中頻的音檔(圖 4)做 resample 的 Shape 和 Spectrum

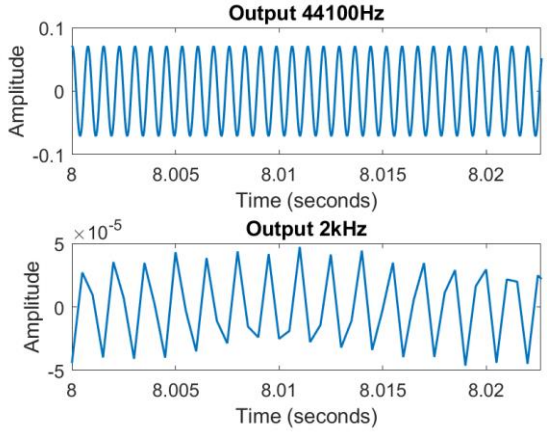


圖 10 對高頻的音檔(圖 6)做 resample 的 Shape 和 Spectrum

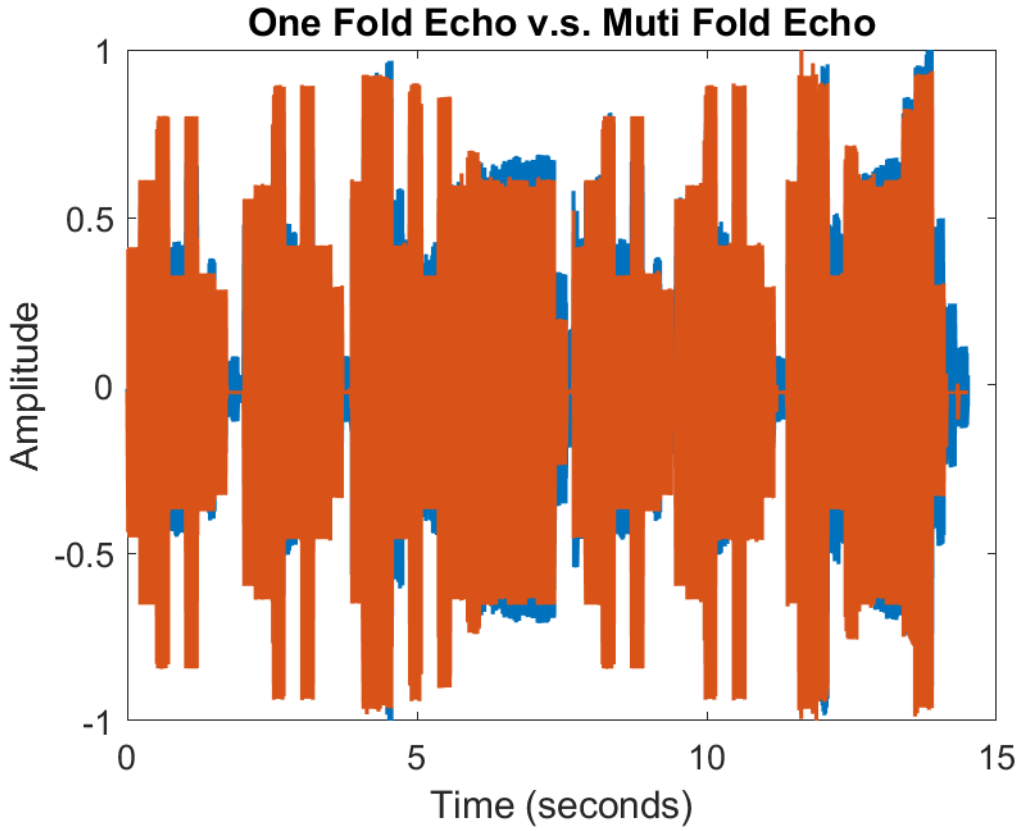


圖 11 One Fold Echo(橘色) v.s. Muti Fold Echo(藍色)
(為了看出連續的變化，取了 50000 個 time step)

分析以及討論

1. 三個 filter 間 Spectrum 和 Shape 的差異

   在 Time domain 上的 Shape 很難看出該 Filter 有甚麼意義，但轉到 Frequency 上的 Spectrum 就有很明顯的意義。如圖 3、5、7 所示，可以很看出三個 filter 分別在低、中、高三個頻域有值，因此和這些 filter 做 convolution 後，就可以得到在低、中、高三個頻域的值。

2. 經過 reducing sampling rate 後，訊號的差異

   如圖 8、9 所示，在低頻(0~350Hz)或是中頻(420~650Hz)的情況下，都還可以保持頻率，儘管 Shape 有改變。

   但圖 10 中，在高頻(800Hz~)的情況下，波頻率已經和原本的不同。

   這是因為採樣頻率只有 2000Hz，如此能夠被完整還原的只有 0~1000Hz，超過 1000Hz 的就無法還原，所以低頻(0~350Hz)或是中頻(420~650Hz)受到的影響很小，但對高頻(800Hz~)就會明顯有幾個音節消失。

3. One Fold Echo 和 Muti Fold Echo

   One Fold Echo 聽起來像是加重了每個單音，從 Shape 上也可以看出這點，橘色的部分有時會是幾乎沒有震幅，剛好那也是音樂中斷的地方。

   Muti Fold Echo 聽起來就會有連續的感覺，從 Shape 上也可以看出它沒有中斷的痕跡，在橘色是趨近於 0 震幅時，藍色仍然也有震幅。這是因為 echo 是來自過去的 output，這些值是會隨時間逐漸累積的。

2. Audio Dithering and Noise Shaping

實作步驟

1. 讀入 Tempest.wav，得到震幅數值的 inputAudio(1-D Array)和採樣頻率 fs。
2. 將 inputAudio 放入 BitReduction()得到 bitReductionAudio，其 BitReduction()裡面步驟 ：
   (1) inputAudio 先往上 shite 1，其範圍從[-1,1]變成[0,2]
   (2) inputAudio 每個 element 乘上 2^7，其範圍變成[0,256]
   (3) 對每個 element 做四捨五入，如此每個 element 的值都可以使用 8-bit 來表示
   (4) 每個 element 除 2^7，其範圍變成[0,2]
   (5) inputAudio 往下 shite 1，其範圍從[0,2]變成[-1,1]，如此便得到 bitReductionAudio
3. 將 bitReductionAudio 放入 AudioDithering()得到 ditheredAudio，其 AudioDithering()裡面步驟 ：
   (1) 每個 noise 都是隨機的，其範圍為[-1,1]

$$\text{noise} = (\text{round}(255 * \text{rand}) - 127)$$

$$\text{noise} = \begin{cases} 1 & , noise = 128 \\ \dfrac{noise}{127} & , noise \neq 128 \end{cases}$$

   (2) 再將 noise 除 127，讓它不會過於影響波形
   (3)

$$\text{ditheredAudio(i)} = \text{bitReductionAudio(i)} + \text{noise(i)}$$

   (4) 對 ditheredAudio 做 normalize 處理，將其範圍再固定回[-1,1]
4. 將 ditheredAudio、shapingFeedback 放入 NoiseShaping()得到 noiseShapingAudio，其 NoiseShaping()裡面步驟 ：
   (1) 從第一個 element 開始做，假設做到第 i 個
   (2) 計算前一個的 err

$$\text{err(i} - 1) = \begin{cases} 0 & , i = 1 \\ ditheredAudio(i) - noiseShapingAudio(i) & , i \neq 1 \end{cases}$$

   (3) 計算當前 element

$$\text{noiseShapingAudio(i} + 1) = \text{ditheredAudio(i} + 1) + \text{shapingFeedback} * \text{err(i} - 1)$$

   (4) 再對$\text{noiseShapingAudio(i} + 1)$做 quantized(使用前面的 BitRedution)
   (5) 等做完所有 element 後，對 noiseShapingAudio 做 normalize 處理，將其範圍再固定回[-1,1]
5. 利用 Q1 做的 myFilter()來做出 Low-pass filter，並和 noiseShapingAudio 做 convolution 得到 filteredAudio

6. 將 filteredAudio、threshold 放入 AudioLimiting()得到 limitingAudio，其 AudioLimiting()步驟 ：

   (1) 對每個 element 做檢查，如下

$$\text{limitingAudio}(i) = \begin{cases} \text{threshold} & , filteredAudio(i) > \text{threshold} \\ -\text{threshold} & , filteredAudio(i) < -\text{threshold} \\ filteredAudio(i) & , else \end{cases}$$

7. 將 limitingAudio、Min(=-1)、Max(=1)放入 Normalize()得到 recoverAudio，其 Normalize()步驟 ：

   (1) 計算 inputRange

$$\text{inputRange} = \max(\text{limitingAudio}) - \min(\text{limitingAudio})$$

   (2) 計算 outputRange

$$\text{outputRange} = \text{Max} - \text{Min}$$

   (3)

$$\text{recoverAudio} = \text{Min} + \text{outputRange} .* \left(\frac{\text{input} - \min(\text{input})}{\text{inputRange}}\right)$$

結果圖片

為了方便觀察，跟時間有關的圖只會顯示 100 個 time step(約 0.00227 秒)，

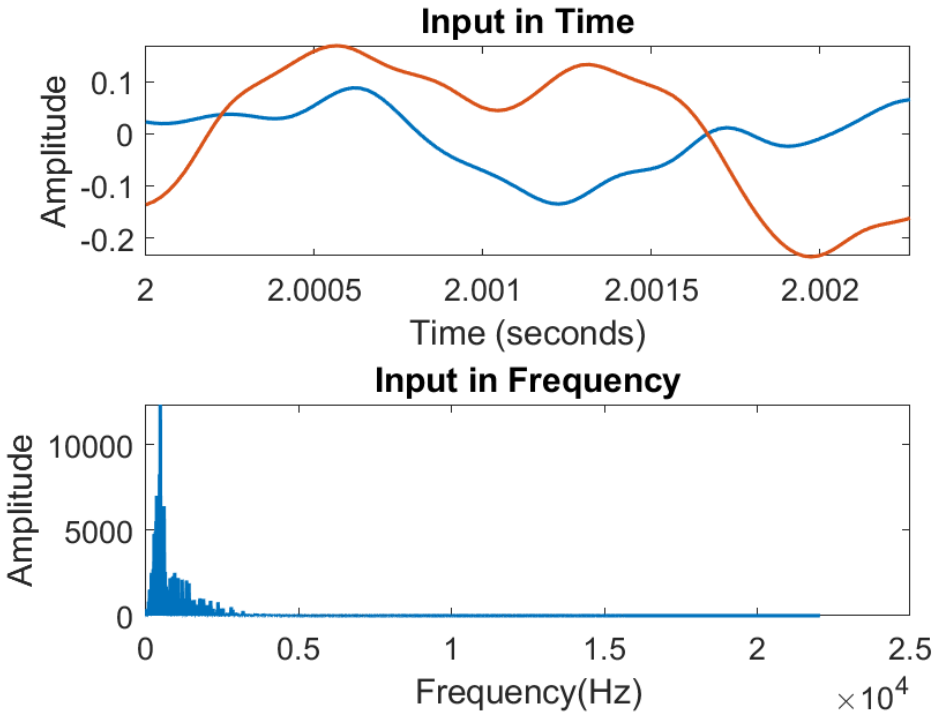


圖 12 input 的 shape 和 Spectrum

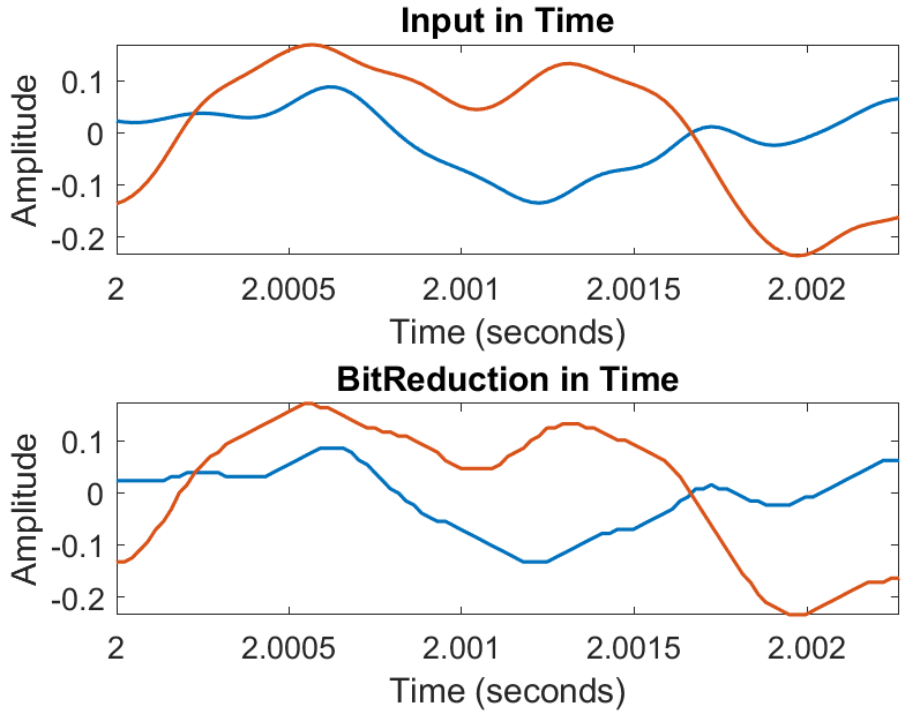


圖 13 input v.s. after bit Reduction

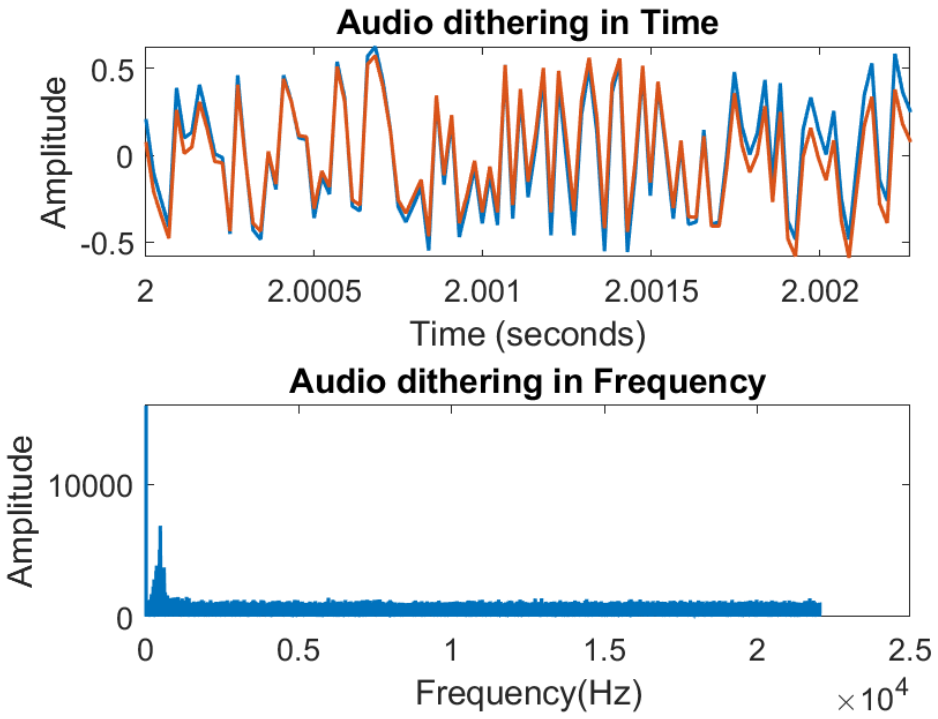


圖 14 經過 Audio dithering 後

ditherdAudio 的 shape 和 Spectrum

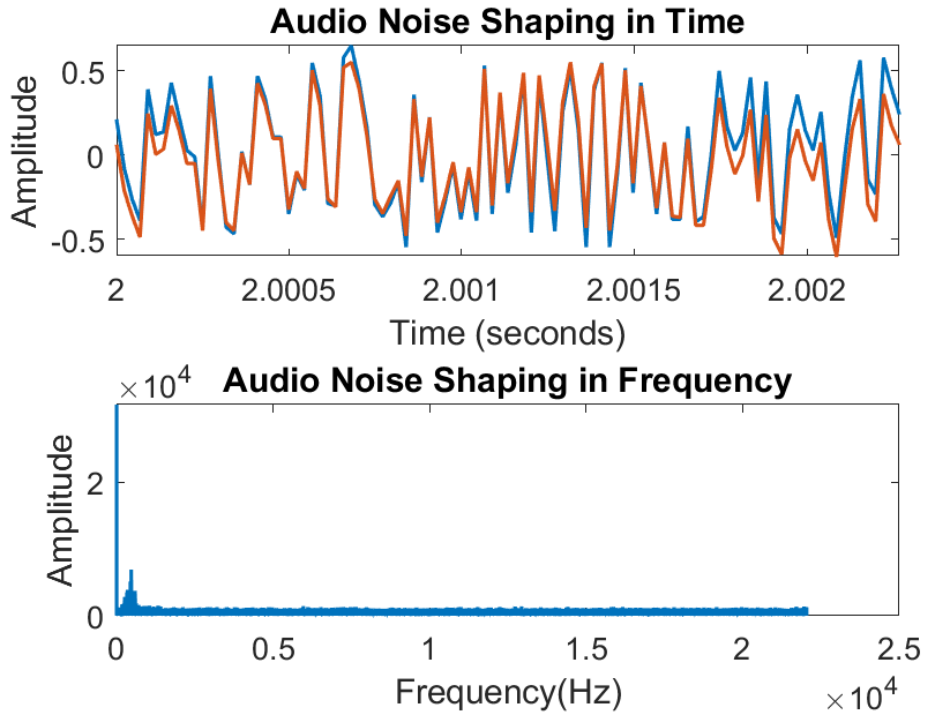


圖 15 經過 Noise Shaping 後

noiseShpaingAudio 的 shape 和 Spectrum

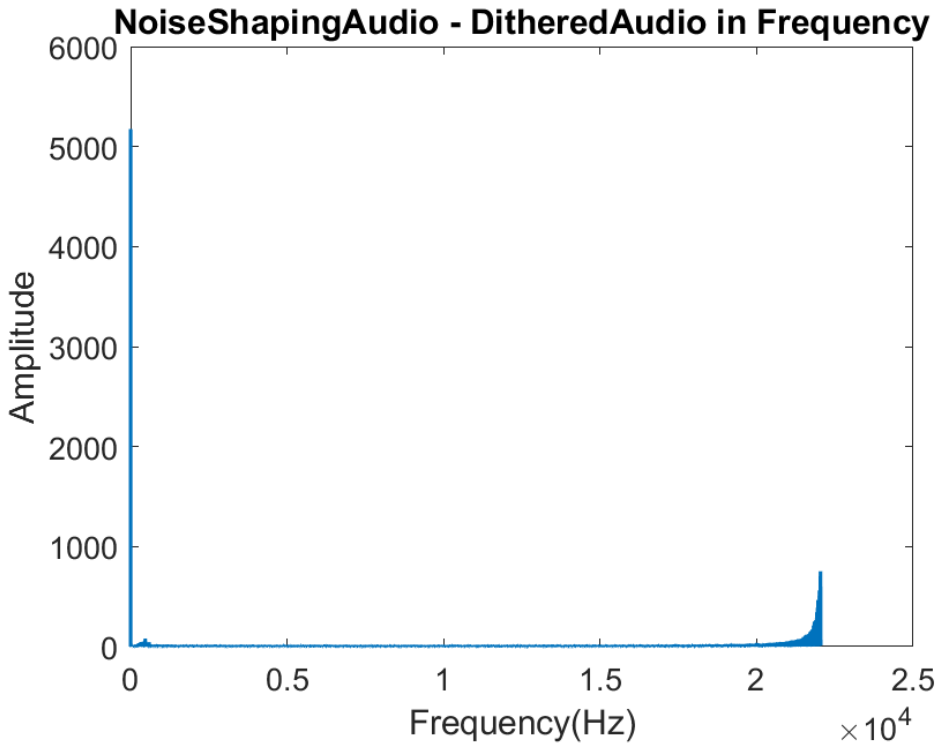


圖 16 NoiseShapingAudio 的 Spectrum 減去 DitheredAudio 的 Spectrum

(可以明顯看出高頻多出一段，那段是雜訊)

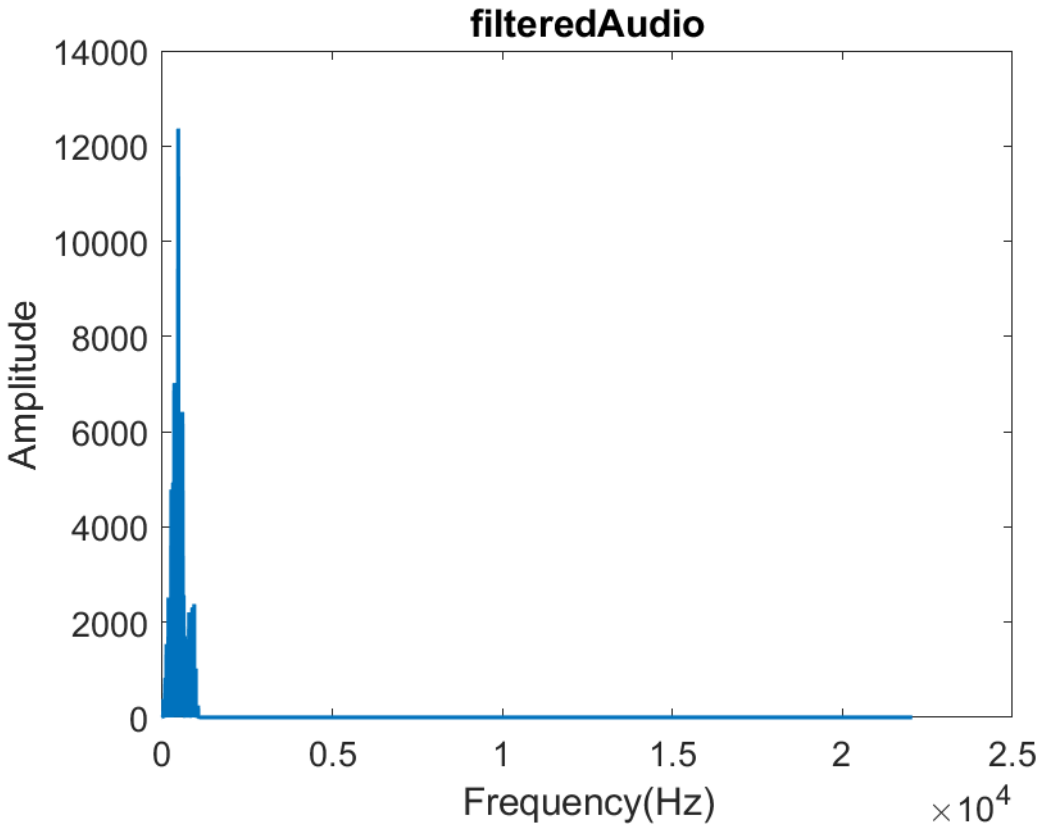


圖 17 NoiseShapingAudio 經過 Low-pass Filter 後得到

filterAudio 的 Sepctrum

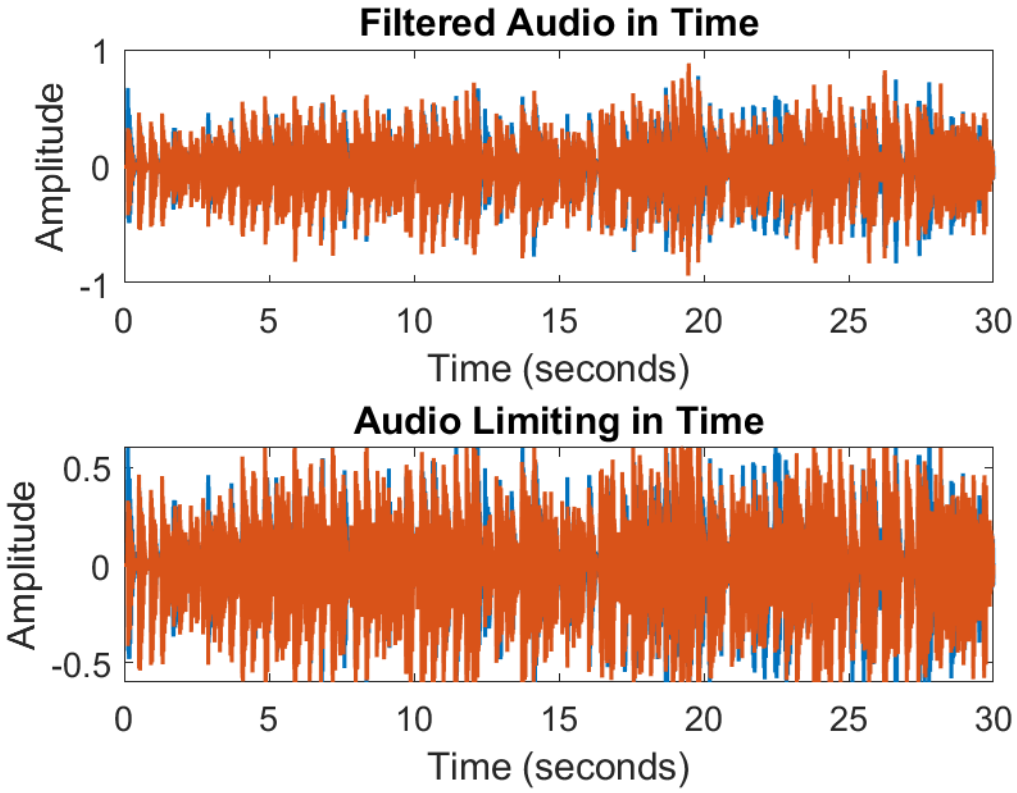


圖 18 fiteredAudio 經過 Audio Limiting 後得到

limitingAudio 的 Sepctrum

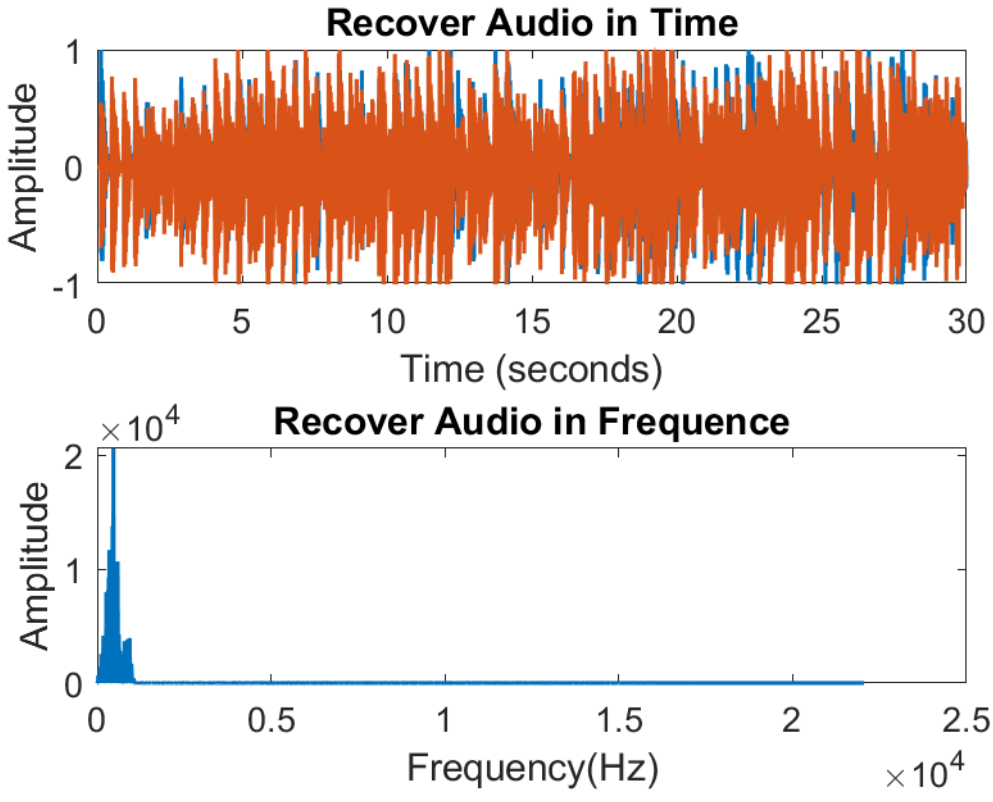
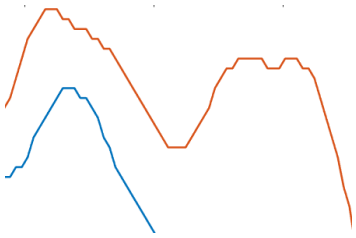
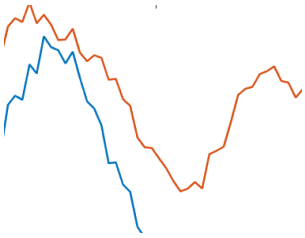
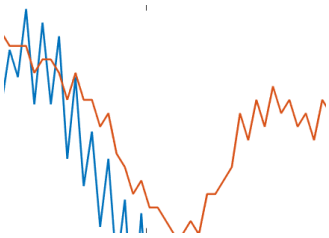


圖 19 limitingAudio 經過 Normalize 後得到

recoverAudio 的 Shpae 和 Spectrum

分析以及討論

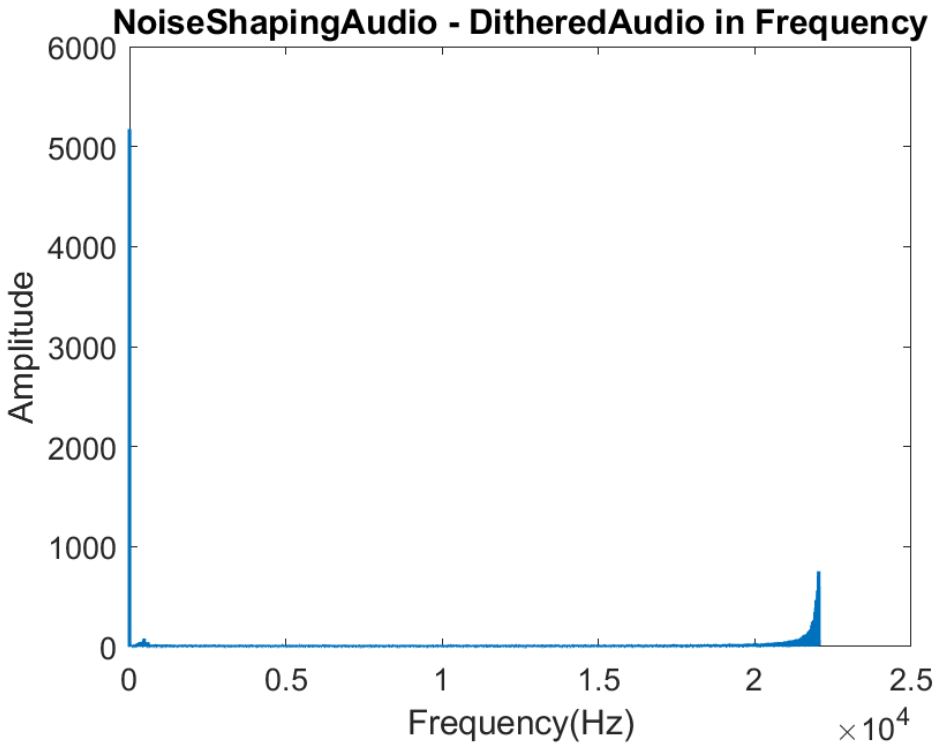
1. Discuss the effect of dithering and noise shaping according to the spectrums and shapes you plot

		
經過 bit Reduction 後	經過 audio dithering 後	經過 noise shaping 後

再經過 bit Reduction 後，可以很明顯看出 shape 裡有明顯的階梯狀，這導致我們聽的時候會有雜音。

經過 audio dithering 後，雖然還是有點陡峭，但是階梯狀的已經明顯減少。

而再經過 noise shaping 後，雖然波形看起來更陡峭，但看以下這張圖



可以看到雜訊被帶往高頻的地方，如此接著使用 low-pass filter，就可以將雜訊濾除。