Q1. Create your own FIR filters to filter audio signal

此題目的是要將三種混合音訊利用 FIR filter 成功區分，由 input signal 的 spectrum 中(透過 function – makeSpectrum()可得)，我們可以發現此三種混合音訊存在不同 frequency 的區間，因此可以利用此特性將音訊進行切割。在這裡，我透過不斷嘗試，最後選擇使用[400]、[500-750]、[900]將音訊區分為適用 low-pass、band pass 與 high-pass filter，這是我嘗試以來覺得輸出後雜訊與音質較佳的切割方法。

在 myFilter function 的實作中，首先，就是選擇 N(filter size)，在多次嘗試中，我自己覺得提高 N 的 size 有助於 filter 音訊的雜音減少，因此我最後選擇將 N 設置為 999。接下來，就是根據不同型態的 filter 方式來做相對應的處理了，在這裡我的實作方法，是參考講義中如下的公式，利用 if-else 的方法檢查每一個 n 是否於 0，並 assign 相對應的公式後即可完成。要特別注意的是，當我們用 N/2 取 mid 值的時候，因為 N 為奇數，因此會使用 floor()與 ceil()來去除餘數的干擾。

TABLE 5.2 Equations for Ideal Impulse Responses for Standard Filters, Based on Cutoff Frequency f_c and Band Edge Frequencies f_1 and f_2

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	$\frac{\sin(2\pi f_1 n)}{\pi n} - \frac{\sin(2\pi f_2 n)}{\pi n}$	$1 - 2(f_2 - f_1)$

接下來，我們就要建立 window function，因為此題指定的 window 為 Blackmann，因此根據講義的 Blackmann window function，我們即實作出需要之結果。但是比較不同的是，因為我的 for 迴圈是從 1 跑到 n-1，因此應該把 0.5 前的係數改為負號才可以得到正確的結果。

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

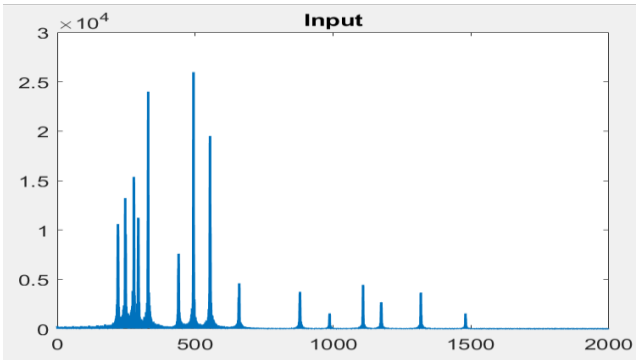
最後，我們要將 low-pass 的結果進行 one-fold 與 multiple echo，在這裡，只要寫出兩種 echo 需要的 matrix ak 與 bk，並搭配講義中給的公式進行實作，並搭配上 filter function 即可完成。

One-fold echo: a_k = [1, 0, 0, 0, ..., 0, 0.8] , b_k = [1]
(That is, 3199 zeros between 1 and 0.8.)
The output of the filter is: **y[n] = x[n] + 0.8*x[n-3200]**

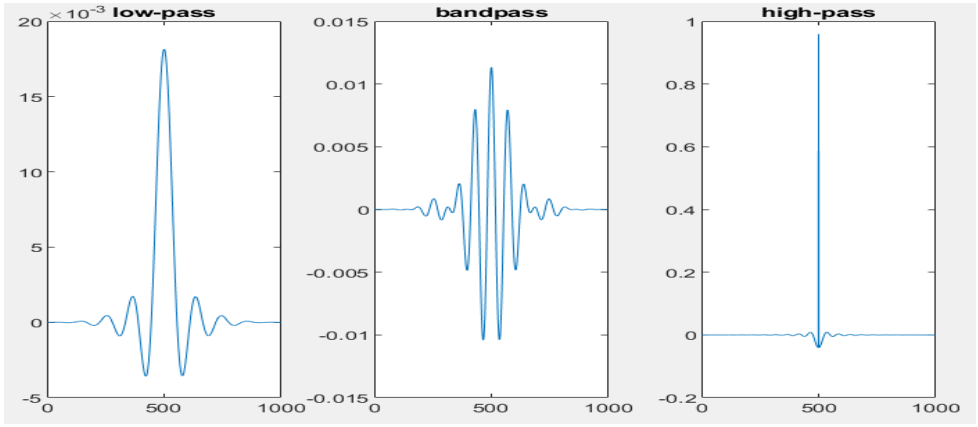
Multiple-fold echo: a_k = [1], b_k = [1, 0, 0, 0, ..., 0, -0.8]
(That is, 3199 zeros between 1 and -0.8.)
The output of the filter is: **y[n] = x[n] + 0.8*y[n-3200]**

實作結果：

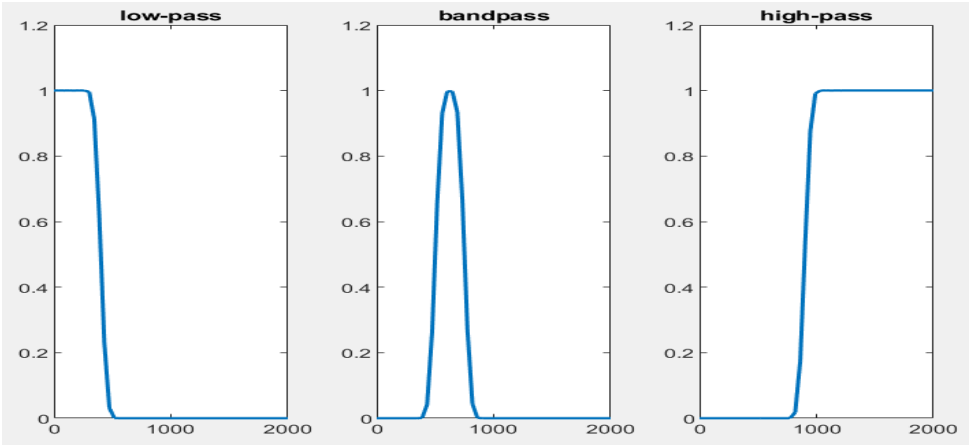
The spectrum of the input signal：



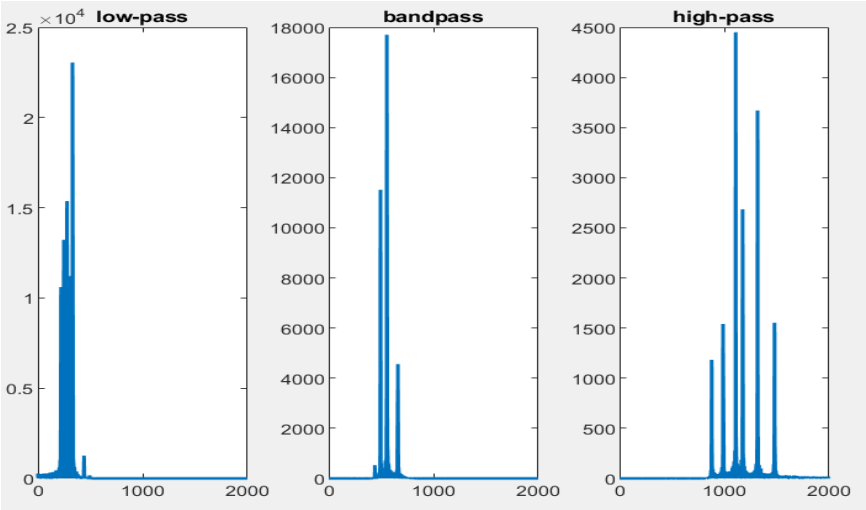
The shapes of the filters (time domain)：



The spectrums of the filters(frequency domain)：



The spectrums of the output signals. (before echo)：



在 filter spectrum 的結果中，我們可以清楚 input 的混合波頻率和濾波後的頻率和範圍，可以發現我們有成功的將混合音訊分為低頻、中階、高頻的差別，顯示完成了分離音訊的工作。

從 filter shape 我們可以看出，low-pass filter 的 y 軸差距最小，起伏最為平緩，high-pass 橫跨的數值最大，而 band pass 介於兩者之間，又因為 high-pass 與 low-pass 在公式上是單純的 sin 函數又只差了一個係數的正負號，因此比起 band pass，都有在有數值的期間則會使曲線特別凸出，而左右兩端則有漸漸向 0 趨近的趨勢。

Q2. Audio dithering and noise shaping

這題希望我們將 input 的 signal 由 16 bits 轉為 8 bits，由於這樣的 bit reduction 會造成音質降低與雜音出現的結果，就需要透過 audio dithering 以及 shaping 等步驟來嘗試復原此音訊。
首先，在 bit reduction 的部分，由於原本我們 audioread()得到的型態為 double，因此得到的值會介於 -1~1 的區間，透過 round(input*(2^7))/2^7，我們就可以完成 bit reduction，在這裡，round(input*(2^7))可以將 16 bits 轉為 8bits，將-128~128 的範圍 mapping 到我們 8bits 的 256 格上，而最後的 /2^7，是還原成原本的值，如此才可以正確地使用 audiowrite()不出錯，寫出成為.wav 檔案。
我使用 bit reduction 後的 input 來進行 dithering 和 shaping。在 dithering 的部分，可以透過 rand 來 random 變數、產生 uniform distribution，最後將 reduction 完畢的值 input2 加上 random 值，就完成 audio dithering。

至於在 audio shaping 的部分，我參考講義中的公式來完成我的 for 迴圈，random 變數 Di 的部分，因為在上面的 dithering 步驟已經加過，因此這裡就不再重複做了。在這裡，我選取的 c 值為 0.9，並根據公式定義：當 i 值不為 1 時，Ei = f_in (i,j) - f_out(i,j);否則，Ei = 0。最後的 f_out(i,j) = f_in(i,j) + 0.9 * Ei 即可得到需求結果。

- D_i is a random dithering value added to the ith sample.
- The assignment statement $F\_in_i = F\_in_i + D_i + cE_{i-1}$ dithers and noise shapes the sample. Subsequently, $F\_out_i = [F\_in_i]$ quantizes the sample.
- E_i is the error resulting from quantizing the ith sample after dithering and noise shaping.
- For $i = -1$, $E_i = 0$. Otherwise, $E_i = F\_in_i - F\_out_i$.

在 low-pass filter 的部分，只要將上一步驟的 f_out 帶入 Q1 的 myFilter()這個 function 並帶入正確變數即可完成，在這裡，我選擇的 N 值跟第一題一樣為 999、window name 一樣為”Blackmann”、filter name 為題目指定的 low-pass、cutoff frequency 則在幾次嘗試後選擇了 1500。

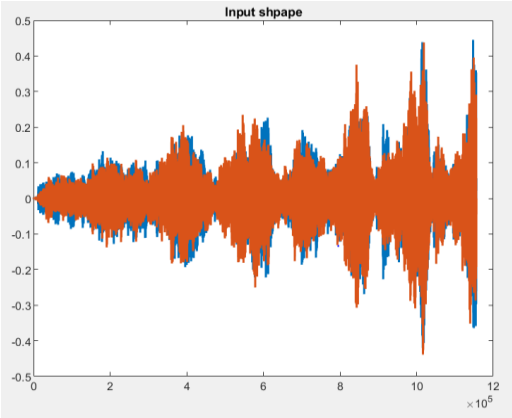
在 limiting 的部分，因為題目指定 hard limiting，按照講義中 hard clippping 的定義，f_out 中只要有超過我們規定 min ~ max 這個範圍，都要強制轉為 min 與 max 這兩個數值，我將 max 設為 0.45、min 設為 -0.45，並透過雙層迴圈將 f_out 強制轉為我們規定的範圍。

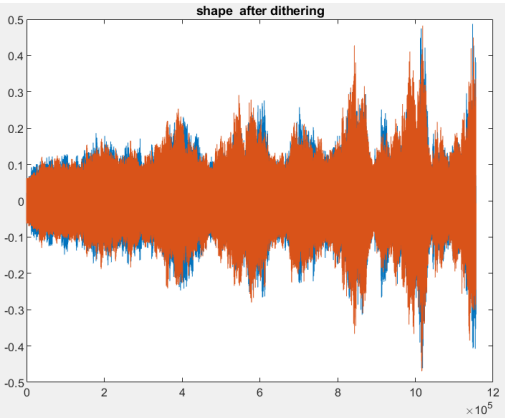
- Hard limiting (clipping)
  - cuts amplitudes of samples to a given maximum and/or minimum level.

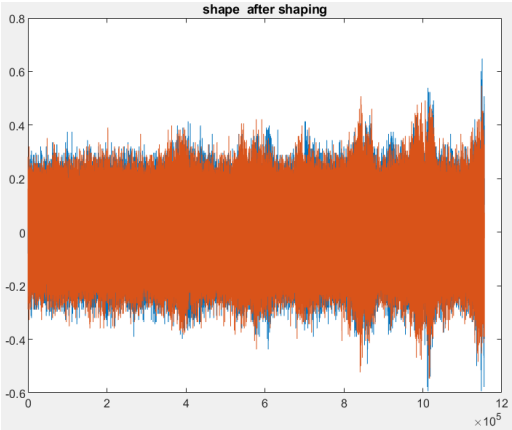
Normalization 的步驟也是按照講義所示，透過 max(f_in)、max(f_out)，我們可以得到 input 與 output 的 highest amplitude sample，gain 值則透過兩者相除可以得到，最後，只要將 f_out 同乘這個 gain 值，即可完成 raise all samples，完成 normalization。
最後，只要透過 audiowrite()，帶入最後的 f_out，即可得到修復完成的音訊檔案。

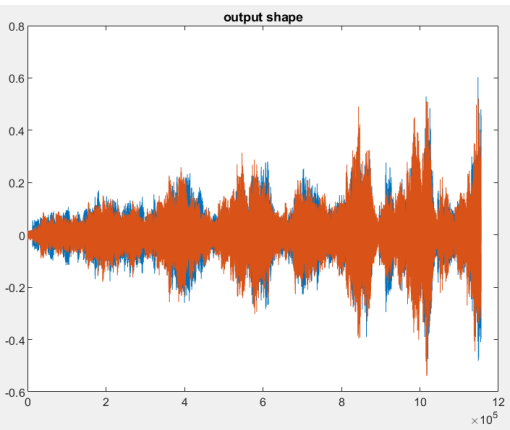
- Normalization steps:
  1. Find the highest amplitude sample in the audio selection.
  2. Determine the gain needed in the amplitude to raise the highest amplitude to maximum amplitude.
  3. Raise all samples in the selection by this amount.

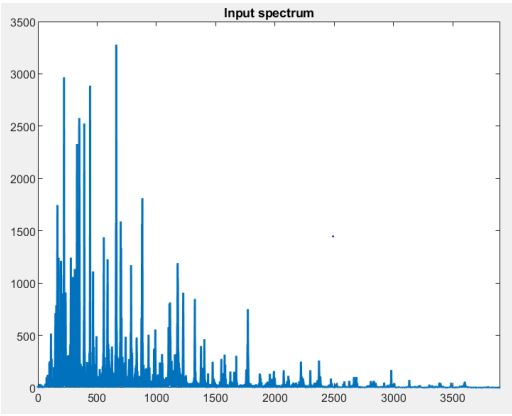
實作結果：

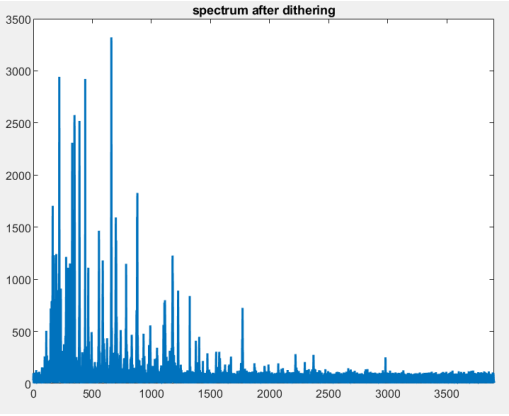

Input shpape
×10^5

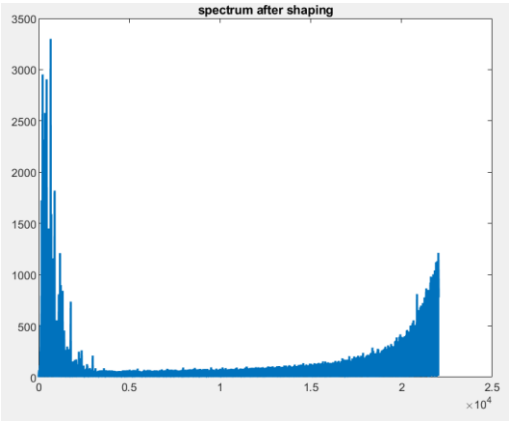

shape after dithering
×10^5

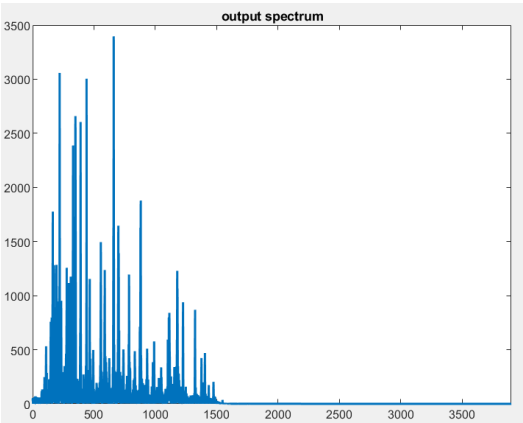

shape after shaping
×10^5


output shape
×10^5


Input spectrum


spectrum after dithering


spectrum after shaping
×10^4


output spectrum

比較 shape 可以發現，當把 input signal 加入 audio dithering 後，會使原本有棱角的偏矩形波形變得比較不那麼方方正正；加入了 audio dithering 後，高頻率的 frequency 增加；而經過 shaping 後，我們可以發現會使波形的高頻率部分上升，x 軸後端有一個尖峰出現，這樣的 shaping 有助於校正 quantization error，因為人耳對於高頻率的誤差敏感度較小。
在最後完成修復後，spectrum 回復跟 input 的相似，相較於原本 input 的波形，output 的波形較為柔順，不會是方方正正的偏矩形波，得到較佳的聲音結果。而由 spectrum 的圖輔助來看，也可以發現當我們最後完成時，會把高頻率部分去除，因此最後結果中，原本 input 中高頻率的部分會是平滑維持在 0 的線條，沒有向上起伏變化。

參考資料：

http://mirlab.org/jang/books/audioSignalProcessing/filterApplication.asp?title=11-1%20Filter%20Applications%20(%C2o%AAi%BE%B9%C0%B3%A5%CE)

https://www.mathworks.com/help/stats/prob.normaldistribution.pdf.html