102062111 林致民

多媒體技術概論 HW3 Report

Q1.

這一題是要把混在一起的三首歌拆開來，那稍微用耳朵聽了一下，加上工具的輔助，發現到三首個的頻率分布在這三個範圍：[～ 441]、[441, 800]、[800 ～]，所以第一個我選擇用low-pass，第二個用band-pass，第三個用high-pass。

首先要先做出三個filter：low-pass, band-pass, high-pass，然後經過windowing function後，再跟原始音樂檔做convolution。

filter的部分，參數我基本上使用N = 501, Fs = 44100，windowing function 採用rectangular windowing function = w(n) = 1

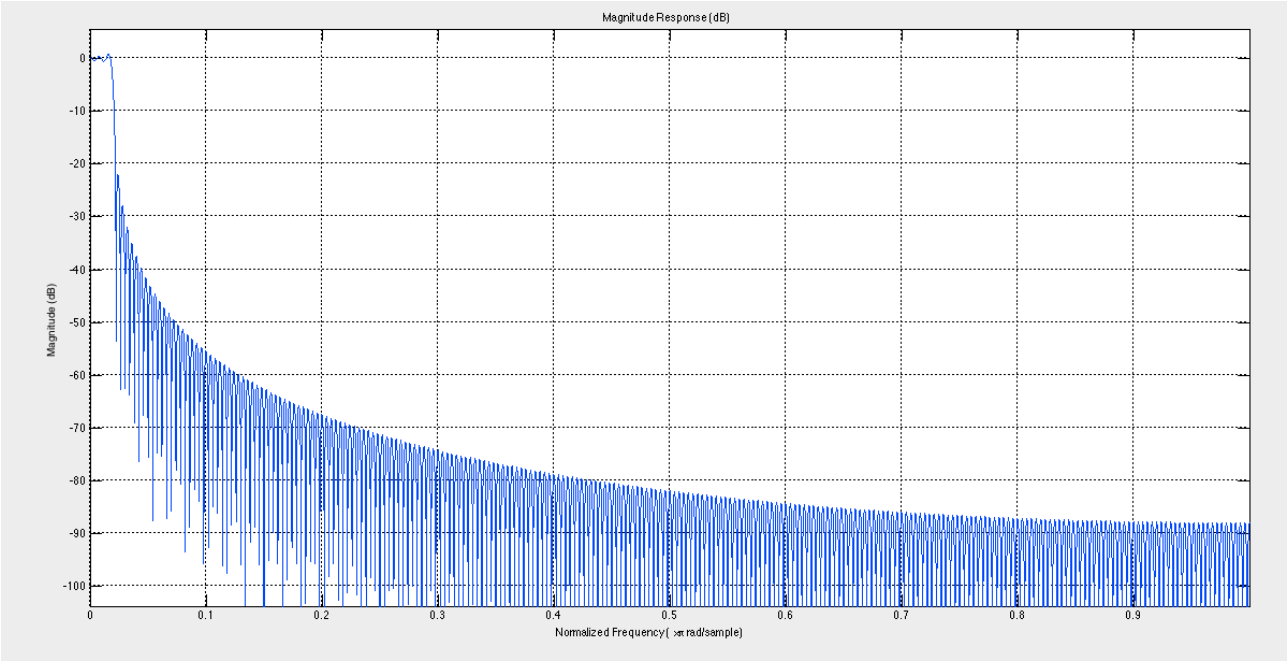
1. low-pass的實作，fc = 441 ：

```
middle = floor(N/2);

for n = (-middle):middle,
    if n == 0
        fltr(middle+1) = 1;
    else
        fltr(n+middle+1) = sin(2 * pi * f_c * double(n)) / (pi * double(n));
    end
end

fltr(middle+1) = 2 * f_c;
```

然後以下是結果（使用fvtool(filter)這個API）：



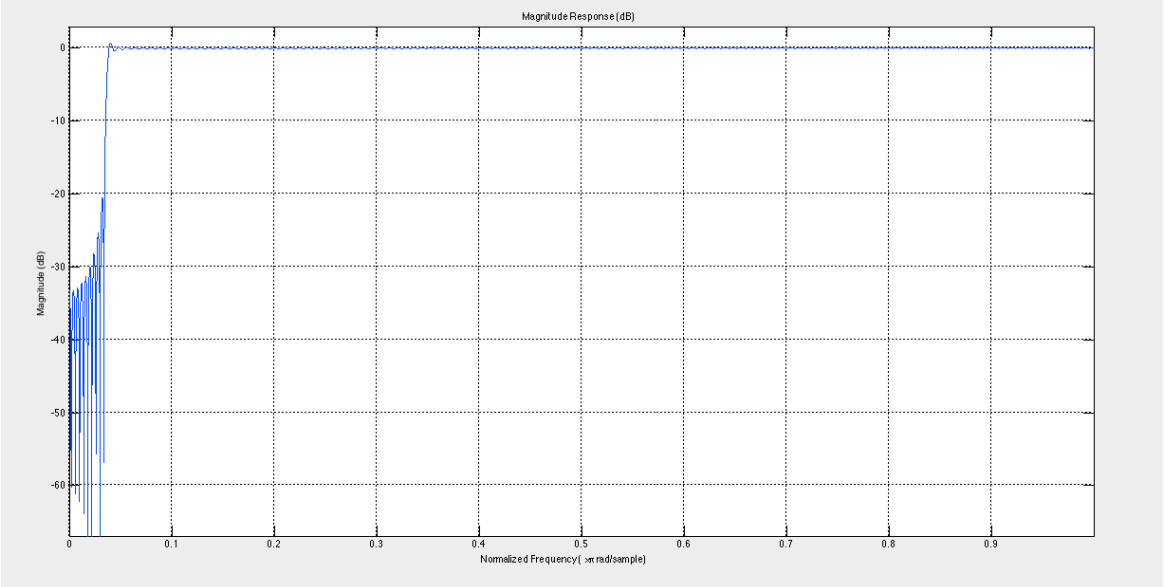
在frequency domain上，發現到低頻的部分在高點，而高頻的部分則會被篩掉。

2. high-pass 實作，fc = 800：

```
for n = (-middle):middle,
    if n == 0
        fltr(middle+1) = 1;
    else
        fltr(n+middle+1) = -sin(2 * pi * f_c * double(n)) / (pi * double(n));
    end
end

fltr(middle+1) = 1 - 2 * f_c;
```

結果：



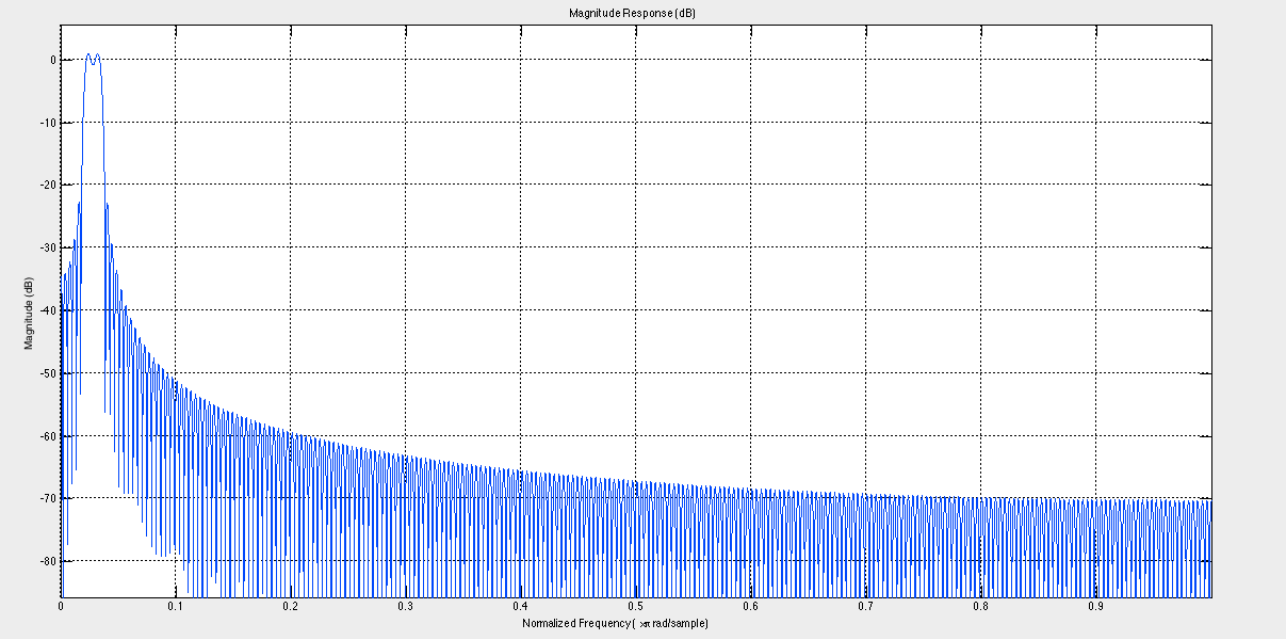
3. band-pass：fa = 441, fb = 800

```
for n = (-middle):middle,
    if n == 0
        fltr(middle+1) = 1;
    else
        fltr(n+middle+1) = (...
                                sin(2 * pi * f2 * double(n))...
                               -sin(2 * pi * f1 * double(n))...
                            )/ (pi * double(n));

    end
end

fltr(middle+1) = 2 * (f2 - f1);
```

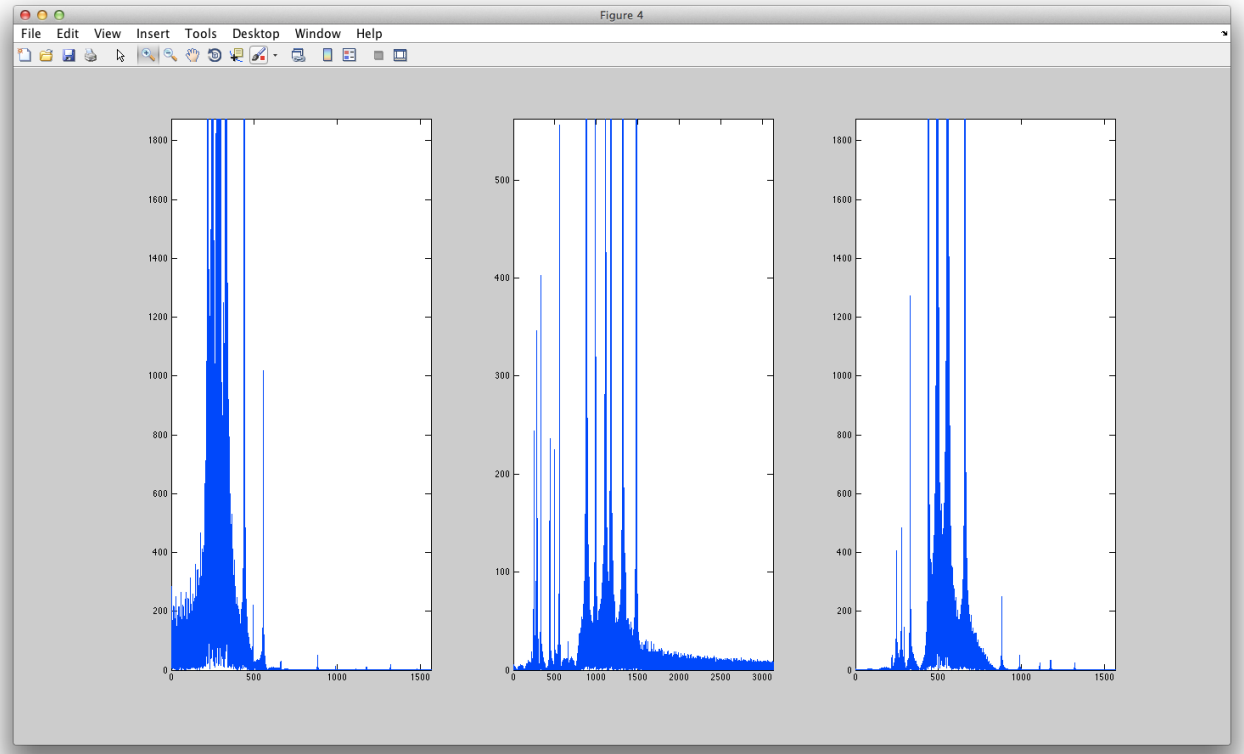
以下是結果：



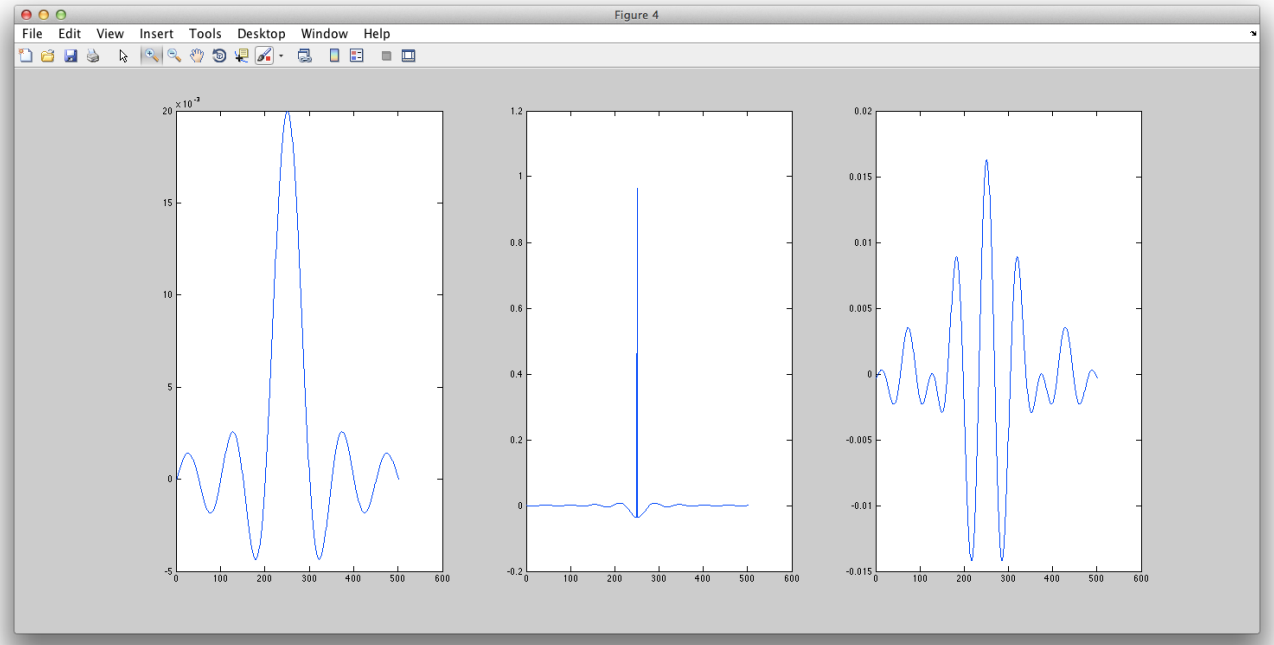
Convolution的話，則是利用幾個for loop就解決了（C code）：

```
untitled
1  for (int i = 0; i < sampleCount; i++ ){
2      y[i] = 0;                          // set to zero before sum
3      for (int j = 0; j < kernelCount; j++ ){
4          y[i] += x[i - j] * h[j];       // convolve: multiply and accumulate
5      }
6  }
```

* 三首歌的結果(從左到右：low-pass, high-pass, band-pass)：



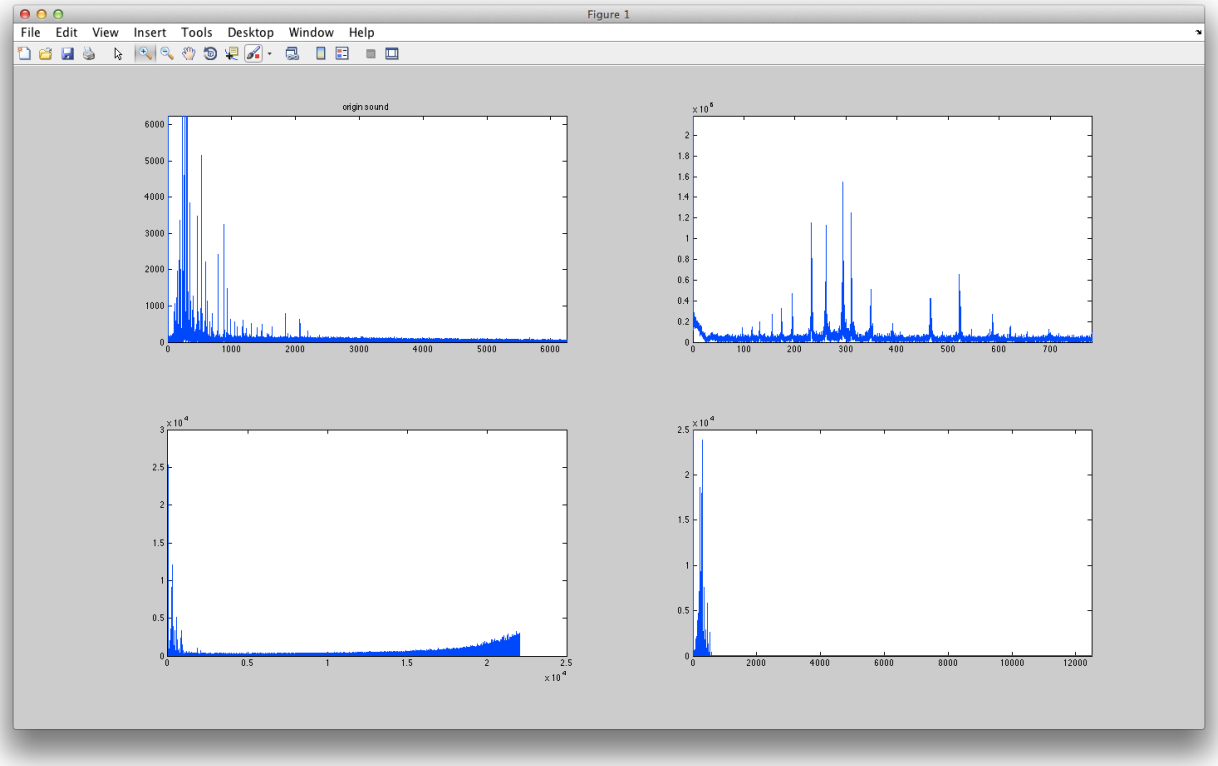
* filter以time_domain的方式繪製(從左到右low, high, band-pass)：



Q2.

這一題助教提供給我們的是從8 bit降到4 bit『雙聲道』的聲音檔，然後要用audio dithering & noise shaping & low-pass filter把聲音還原到跟8 bit一樣，會經過以下這幾個步驟：

1. 原來的聲音檔(左上)
2. 加入一些rand noise（右上）
3. Dithering & noise shaping（左下），把雜訊推到高頻率的地方
4. 經過low-pass filter後的音樂（右下）



至於做法就跟講義上的一樣，首先先把原聲音黨normalize 到 -128 ~ 128之間，之後套入rand_noise， 之後再做noise shaping，所以看起來的算是就像以下這樣（j 是聲道，i 是time domain的位置，c取0.8）：

```
Ei = F_in(i-1, j)/s - F_out(i-1, j);
F_out(i, j) = floor(noise(i, j) + c * Ei);
```

Dithering 只是對原始音樂檔加上一些noise，所以就只是噪音增加，而 noise shaping 則是把噪音往高頻率的地方移動，所以背景的噪音聽起來聲音就比較高。