Introduction to Multimedia Homework #2

作者：103062318 蔡尚倫

Q1. Create your own FIR filters to filter audio signal

Discuss how you determine the filters.

首先，理解兩個參數的意義 N 代表 filter 的大小，fcutoff 在不同 filter 中扮演不同腳色，以 low 為例，它代表保留 fcutoff 以下的訊號。觀察 outputFilter 的 spectrogram，我嘗試過將 N 設為 1001 得到的 filter 在邊界上(意即 300HZ) Transition band 頗大(呈現傾斜的遞減)，並不是理想上垂直的遞減，並將 N 設為 10001 時，Transition band 趨近於 0，符合理想上的 low pass filter，trade off 就是需要花大概 10 倍的時間 run code。而 fcutoff 的抉擇，在於觀察 input signal 高低起伏，來做粗略的判斷，剛開始選定 low pass 300 以下，band pass 450~700，high pass 800 以上，N 使用 1001，至於中間為何會有空掉的頻率，是因為 N 選擇太小的關係，Transition band 的出現造成 output signal 前後會有重疊的部分(混音)，所以只能放棄掉被混音的部分，捨棄一些資訊保留核心，如果 N 使用 10001，就能夠以很乾淨的切法，切 low pass 400 以下，band pass 400~800，high pass 800 以上，所以綜合上述觀察，如果不需要太準確的音樂，容許一些資訊的流失，可以選擇 (run code 快)

N=1001 low pass 300 以下 band pass 450~700 high pass 800 以上

如果你需要很準確的音樂，不容許資訊的流失，可以選擇 (run code 慢)

N=10001 low pass 400 以下 band pass 400~800 high pass 800 以上

我選擇下方的，所以 run code 的時間蠻長的(5~10 分鐘)

How you implement the filter and convolutions to separate the mixed song.

我是照著助教的hint與slide#80完成filter，過程分別是

-> Normalization

N = N-1; 因為N輸入值為奇數，減一變為偶數，取middle會剛好整除，而且在取N/2~-N/2時，由於有0的關係，outputFilter會變成奇數長度

fcutoff = fcutoff/fsample;

middle = N/2; 取中間值用於之後的for-loop

-> Create the correspond filter according the ideal equations(Low,High,Bandpass)

依照不同的參數，產生不同的filter，這裡已Lowpass為例(過程相同套用公式不同)

```
for n = N/2:-1:-N/2   防止 outputFilter change size on every iteration 的warning
        if(n==0) 防止除以0的狀況
            outputFilter(middle+1) = 1; 修改index +1 因matlab array從1開始
        else
            outputFilter(n+middle+1) = sin(2*pi*fcutoff*n)/(pi*n); 理由同上
        end
    end
outputFilter(middle+1) = 2*fcutoff;   記得填回outputFilter(middle+1)的正確值
```

-> Create the windowing function and Get the realistic filter(Blackmann only)

```
for n = 1:N
        outputFilter(n) = outputFilter(n) * (0.42 - 0.5*cos((2*pi*(n-1))/(N-1)) +
0.08*cos((4*pi*(n-1))/(N-1)));   注意outputFilter的index與Blackmann公式中的n是相差一，記得要減一才能算出正確的Blackmann window function
end
```

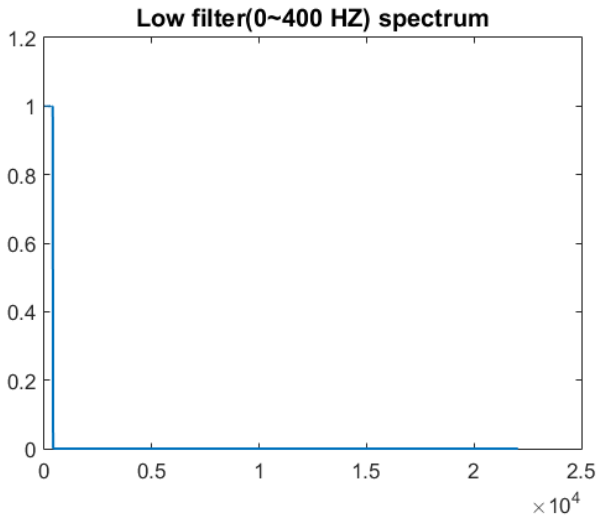
-> Filter the input signal in time domain(means conv)

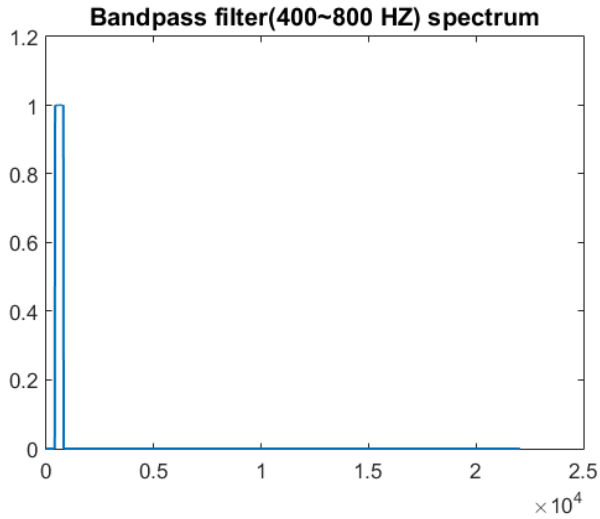
參考官方https://www.mathworks.com/help/matlab/ref/conv.html，由裡面範例得知，1D的conv可以視作多項式相乘(取到與input size一樣即可)，跑兩層for-loop即可完成

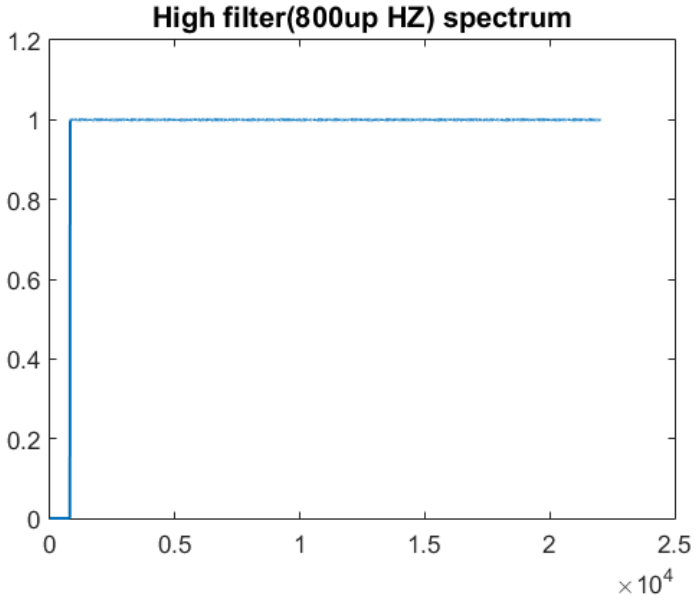
```
for i = 1:length(inputSignal)
    for j = 1:length(outputFilter)
        if(i-j+1>0) 確認index一定要大於0才不會出問題
            outputSignal(i) = outputSignal(i) + inputSignal(i-j+1)*outputFilter(j);
        else 其餘狀況就代表不用繼續
            break;
        end
    end
end
```

Compare spectrum and shape of the filters.

Frequency analysis：

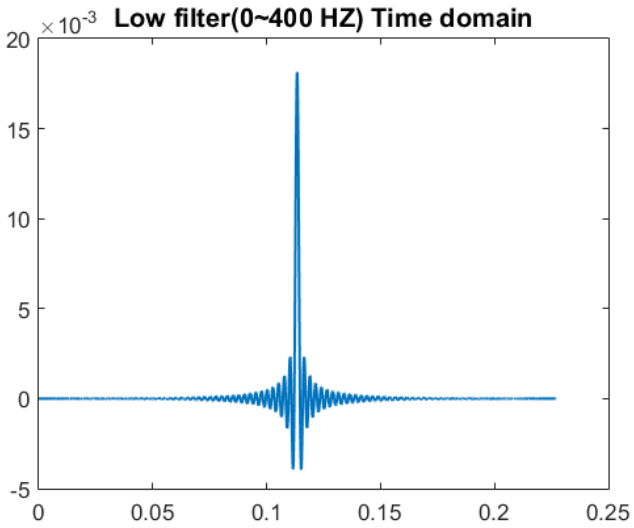


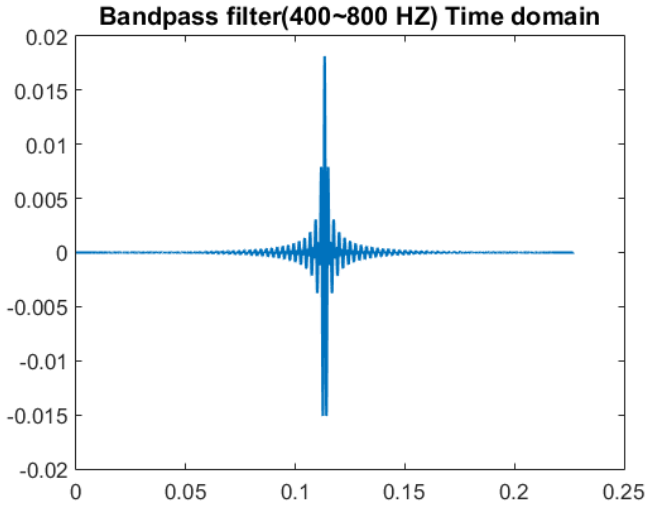


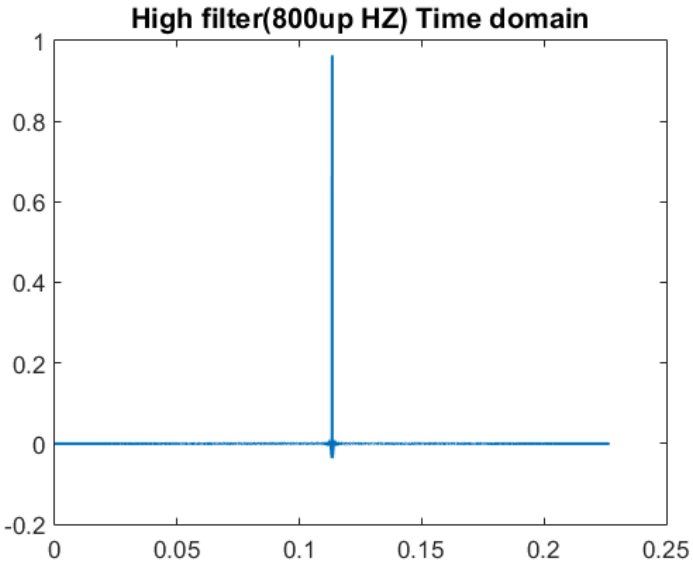


由上圖所示，如果將 input signal 轉為 frequency domain 時，與上述圖片分別做點乘(By convolution theorem)，很明顯的 low pass filter 將 400HZ 以上的頻率全部點乘為 0，band pass filter 將 400HZ 以下與 800HZ 以上的頻率全部點乘為 0，high pass filter 將 800HZ 以下的全部點乘為 0，三種 filter 的差別就很明顯，各自將不同頻率的波段過濾

Time domain analysis：







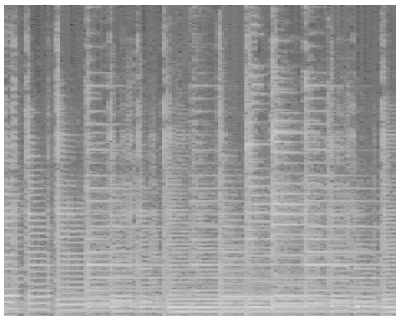
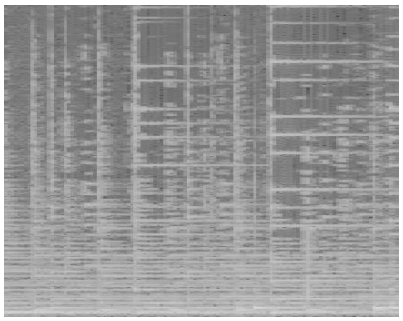
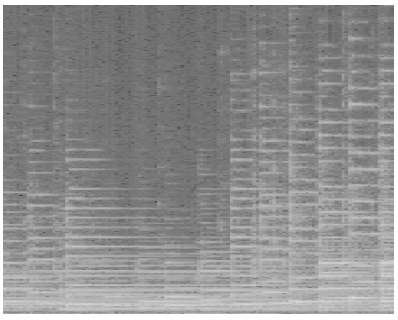
根據頻率的定義，一秒中有幾個波，再由上圖的波型可以發現，high pass filter 比較集中，也就是一秒鐘之內能塞入很多個波，low pass filter 則比較分散，也就是一秒鐘之內能塞入的波就比較少，而 band pass filter 則位於前面兩者中間，再經由 convolution(意義上是某個函數在另外一個函數上的加權疊加)，high pass filter 直覺上假如是一個正的值(周圍是正的機率也很高)，與 high pass filter 做疊加會比原來正很多，而如果是負的值(周圍是負的機率也很高)，那麼疊加後會比原來負很多，所以就會產生上下快速震盪，也就是高頻，low filter 也可以用這種觀點來看，加權相加後相對而言就沒那麼大，所以波與波之間的時間會拉得很開，直觀上就是低頻，而 band pass filter 則夾在中間。

Q2. Music classification through spectrograms and human eyes:

Discuss what you observe on spectrograms of different classes of instruments.

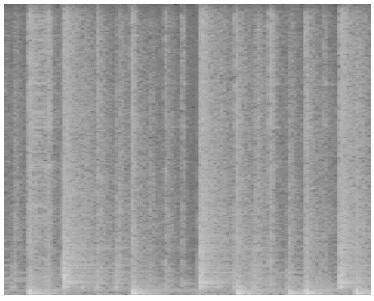
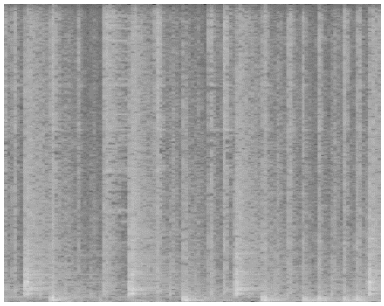
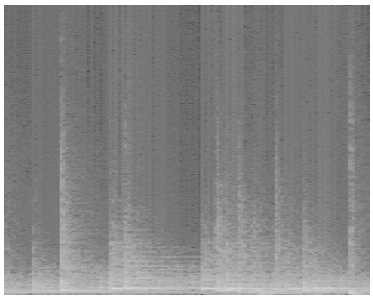
Guitar spectrogram

有彈奏過吉他的人就會知道，吉他譜有些是由許多和弦組合出來的，所以前後會有一段時間某頻率強度相同，就是我們現在看到 Guitar spectrogram 上看到有彈奏的白色條紋狀，而其他頻率則呈現黑色沒有強度，最明顯能區別的特徵就是會以白色長條為分割的帶狀感覺，理由我猜是因為彈奏吉他是會切換和弦，所以頻譜中間會有一長條的白色，每個頻率上都有強度使和弦轉換，聽起來更為順暢。

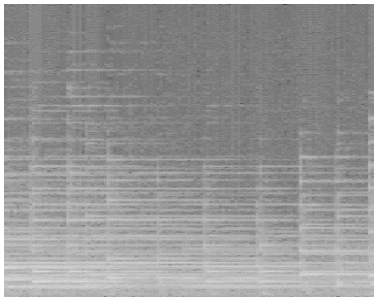
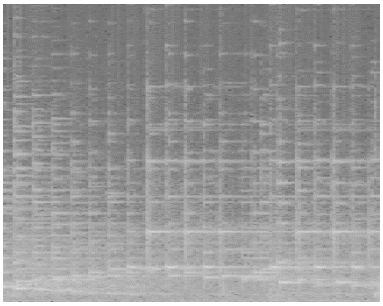
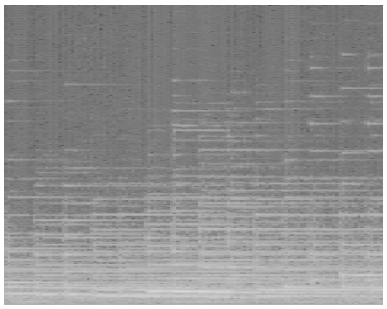
Drum spectrogram

其特色是每條帶狀中，每個頻率的前後強度大致相同，由白至黑，代表震幅越來越小，鼓面上的震度幅度也越來越小，所以聲音會越來越小，直到下次的敲擊才會變為白色

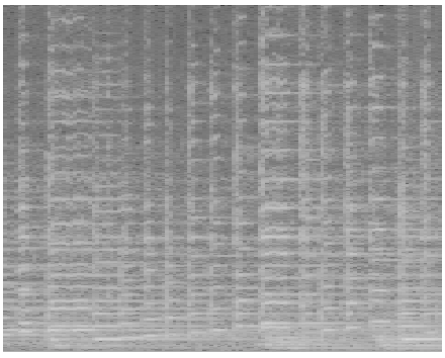
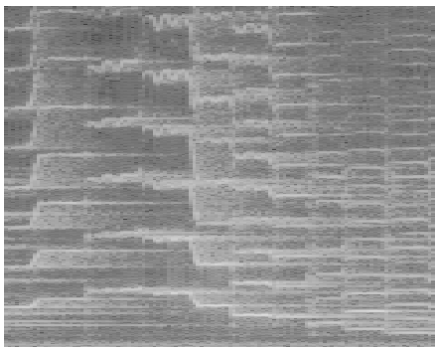
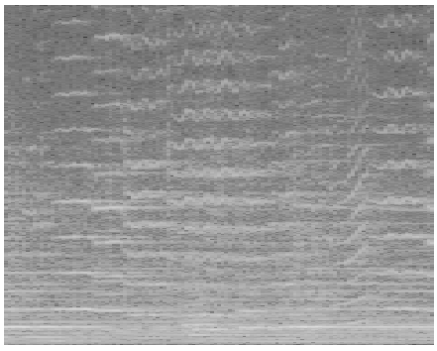
Piano spectrogram

Piano 與吉他還蠻相向的，可是差別我覺得在於音域的大小，能從 spectrogram 看到幾乎每個頻率的聲音都能彈奏，有別於吉他只能在固定幾個頻率上彈奏，固在第二張圖片的下方會呈現階梯狀往上，也就是他在某一時段的內能彈奏的(白色橫條紋)很多頻率不同的組合，piano 像是流水圖，而 guitar 像是格子圖

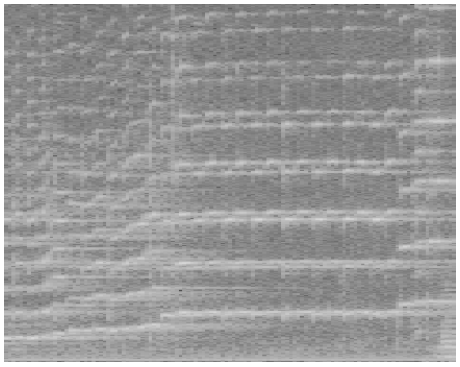
Violin spectrogram

Violin 的 spectrogram 呈現海浪型，帶狀就不太明顯，所以是裡面最好區別的。相信聽過 violin 彈奏，會發現音樂好像有連續的感覺，也就是在短時間內，從 A 頻率變換到 B 頻率，中間用類似 Transition band 填補，中間聽起來就不太會有顆粒狀的感覺

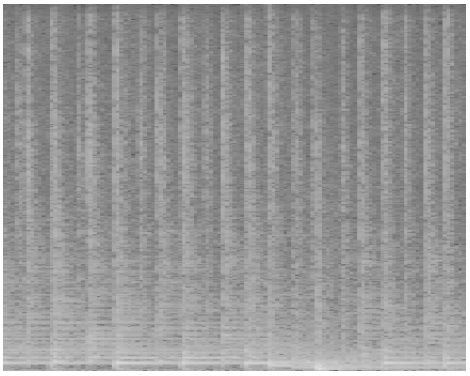




Classify Test Spectrogram

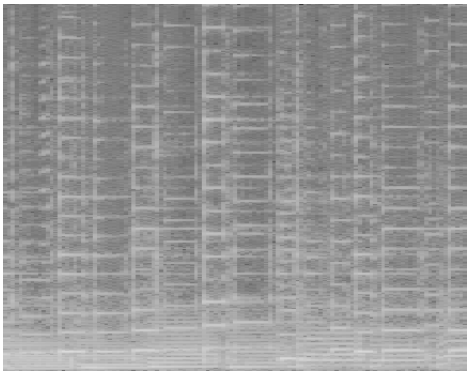
先從最簡單的開始區分，這個有波浪狀的，很明顯就是 violin，所以 test_spectrogram_3 就是 violin



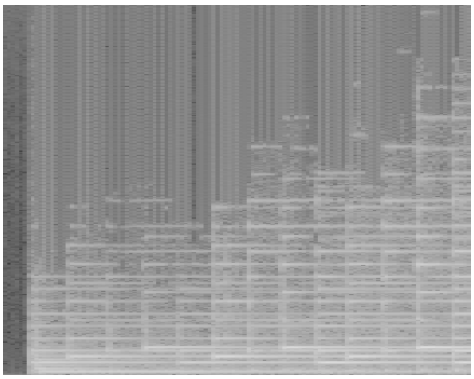
再來是這種帶狀，白色橫條紋並非橫跨整個帶狀，由白至黑，所以這是 drum，所以 test_spectrogram_4 是 drum



接著這張圖片，是帶狀，而且有白色橫紋橫跨整個帶狀，有可能是 piano 或者是 guitar，但是 piano 的話會呈現水流圖，並非這種格子圖，所以 test_spectrogram_1 應該是 guitar



最後這張圖片，是帶狀，而且有白色橫紋橫跨整個帶狀，有可能是 piano 或者是 guitar，但是圖片呈現流水型，與下方有階梯狀的感覺，所以 test_spectrogram_2 應該是 piano



Discuss how you implement “Short-time Fourier transform”

參考 https://www.mathworks.com/help/signal/ref/spectrogram.html
Supplement.pdf 以及 make_spectrum.m 過程分別是

->Parameter define

L1 = 2^nextpow2(segment_duration);　參考make_spectrum產生fft右邊的參數

x_len = length(x);　取得input歌曲的長度

hop = segment_duration-segment_overlap; 藉由傳入的兩個參數計算出跳躍的個數，可以跑到next segment的頭

ham_buildin = hamming(segment_duration, 'periodic'); 內建產生hamming window function

row = L1/2+1; 參考make_spectrum，助教在ilms上解釋為mirror性質，大致上就是圖形會對稱 y=constant ，所以只需L1的一半即可

segments_num = (x_len-segment_duration)/hop +1; 先別看第一個segment，之後的每個segment都是由一個hop再作跳躍，所以取得input歌曲的長度扣掉第一次的segment長度除以hop，得到的個數必須要再加上第一次被你扣掉的

S = zeros(row,segments_num); output matrix的產生，形式在檔案上方的hint有寫了，col的個數必須是所有segment個數(代表時間遞進)，row為剛定義的，因為mirror性質(對稱)

->Perform STFT(依照 Supplement.pdf 步驟)

```
index = 0;
for col=1:segments_num    對每個segment做STFT
    xw = x(index+1:index+segment_duration).*ham_buildin; 利用matlab矩陣slice的表示點乘hamming window
    index = index + hop; 跳到下一個segment的開頭
    fft_xw = fft(xw,L1); 內建fft，並且填入參數L1
    S(:,col) = fft_xw(1:row); 因為fft_xw的大小比S的row數還大，所以依舊使用matlab矩陣slice的方式填入
end
```

F = samplerate/2*linspace(0,1,row);　參考make_spectrum，以0以及1為起始和終點，切成row個點，然後再放大samplerate/2，放大該數是由於最大的頻率只會到smaplerate的一半，因row只取約L1/2，而非全取，轉成frequency domain看也就是只取到samplerate/2，這是從code中解讀出的原因。

T = (segment_duration/2:hop:segment_duration/2+(segments_num-1)*hop)/samplerate;　根據檔案上方的提示，取每個segment的中間時間點，所以是從segment_duration/2開始，以每hop數來數，最後直到egment_duration/2+(segments_num-1)*hop。係數為1/ samplerate是為了將這個T vector轉換成時間