Music Information Retrieval HW3 Report

102062209邱政凱

這次的作業要求我們用Non-negative Matrix Factorization實作Source Separation的演算法，並且做各種分析和比較。

B1、

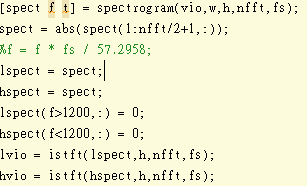
第一個問題是要我們用bss\_eval這個toolbox來測試Source Separation中經常用來當作度量標準的SDR ( Source Distortion Ratio)在各種不同音檔以及設定下的結果。第一列是TARGET音檔，第二列是true source音檔，第三列則是算出來的SDR值。Where a = 01\_vio.wav b = 01\_cla.wav c= 01\_mix.wav n = randn

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [c’;c’]/2 | [a’;b’] | [b’;a’] | [2\*a’;2\*b’] | (a+0.01\*n)’ | (a+0.1\*n)’ | (a+n)’ | (a+0.01\*b)’ | (a+0.1\*b)’ | (a+b)’ |
| [a’;b’] | [a’;b’] | [a’;b’] | [a’;b’] | a’ | a’ | a’ | a’ | a’ | a’ |
| 0.2725  -0.2924 | 256.7621  246.9101 | 256.7621  246.9101 | 256.7621  246.9101 | 20.1376 | 0.1666 | -18.840 | 40.2861 | 20.2848 | 0.2725 |

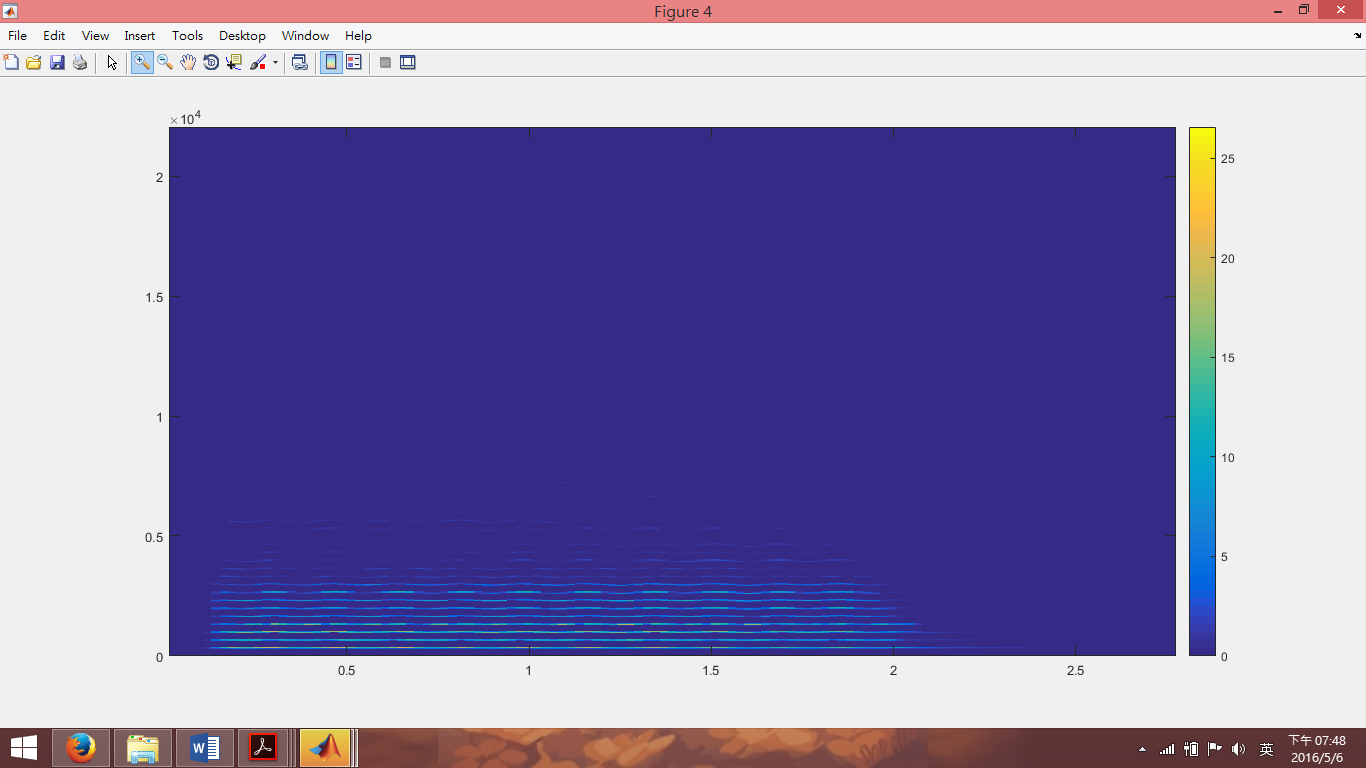
從表中可以看出，混合音檔跟單獨小提琴的音檔是比較像的(正值)，而跟單獨單簧管的音檔則是偏向於不像的(負值)。然後從第二行跟第三行可以看出，bss\_eval在計算SDR時會自動找出跟每個source最相近的target來輸出SDR值，所以TARGET音檔的順序就不是重點了。第四行我們可以看出bss\_eval在計算SDR時應該是有經過Normalization的程序，因為就算把target音檔直接乘上兩倍算出來的SDR值還是跟沒有強化過的是一樣的。第五到第七行可以看出，音檔裡面其實只需一些些微的噪音就可以讓SDR值大幅下降。第八行到第十行則可以看到，相較於random的noise，儘管a和b兩個音檔是不同樂器和不同的音高，但也許是因為都是Harmonics的關係，把b當成雜訊加進a裡面造成的SDR下降不像random noise那麼明顯。

B2、

這題是要我們試試看Inverse Short Time Fourier Transform的效果。把vio\_64.wav音檔轉換成spectrogram的部分，我使用的是matlab內建的spectrogram函式，並且設定Window size為2048，Hop size為1024，number of fft point設為跟window size一樣。實作的部分，把spectrogram中對應於取出的frequency vector中裡面大於或小於1200的部分直接設成0然後再做istft。

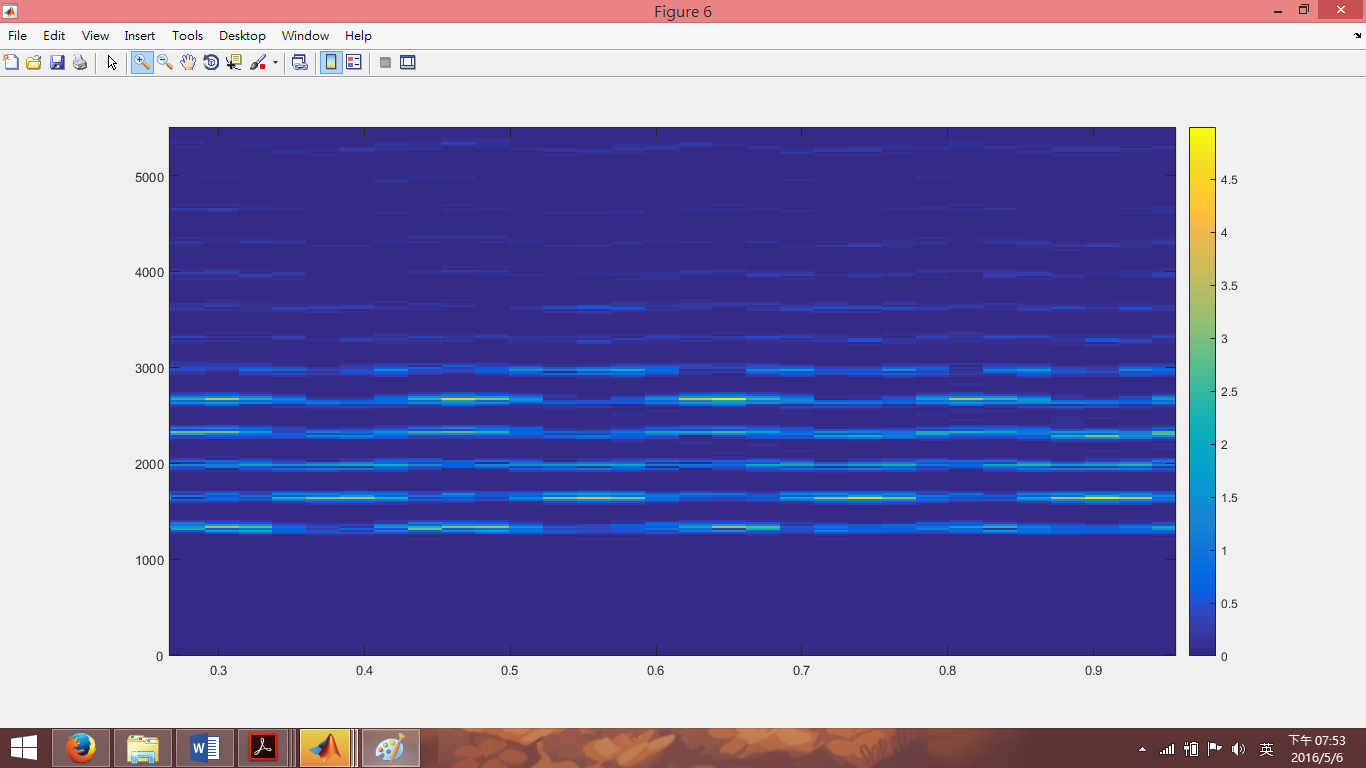
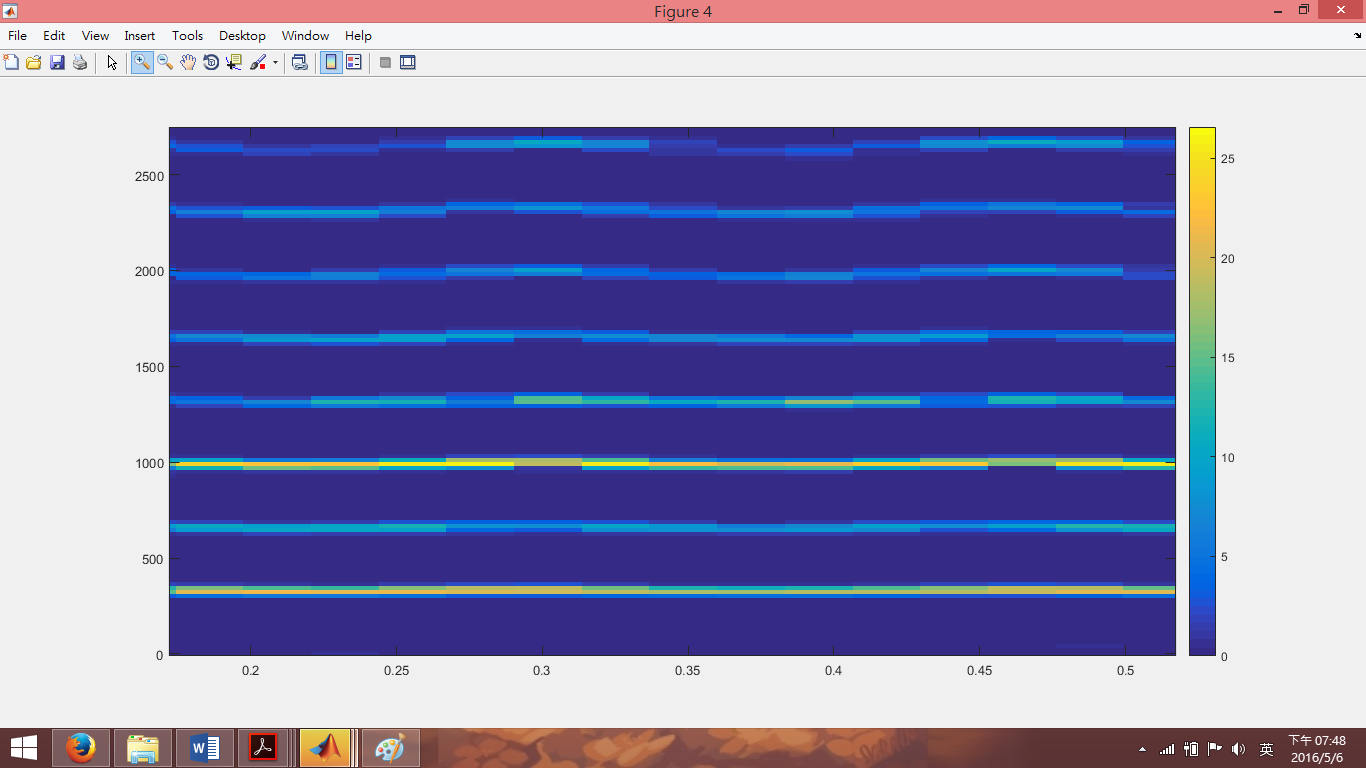


存成音檔之後，再次使用spectrogram函數，把lvio和hvio兩個低高頻的音檔轉換成頻譜圖，並且用imagesc的函數來跟原本的音檔比較。

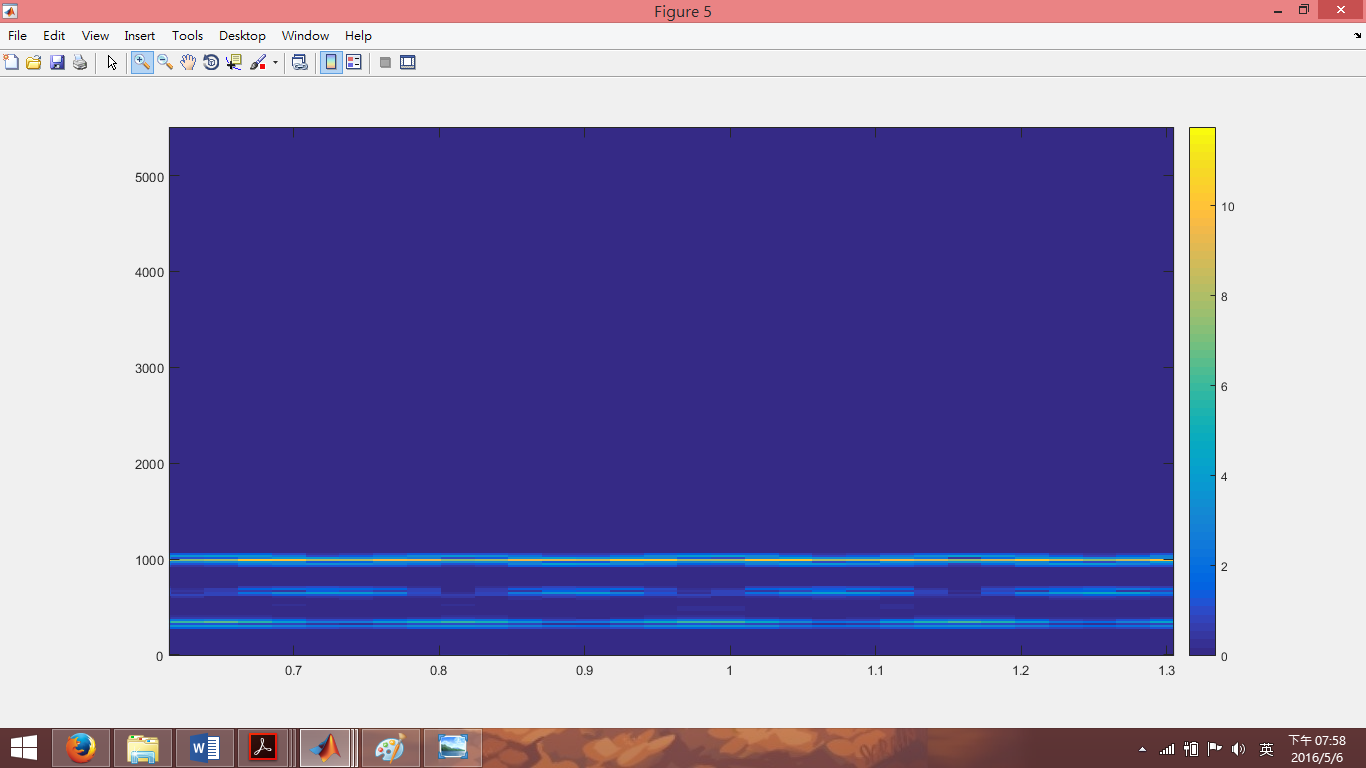


這是原本音檔的spectrogram。

上圖放大的結果。Pitch id 64對應到的基頻是330，可以看到跟此圖吻合

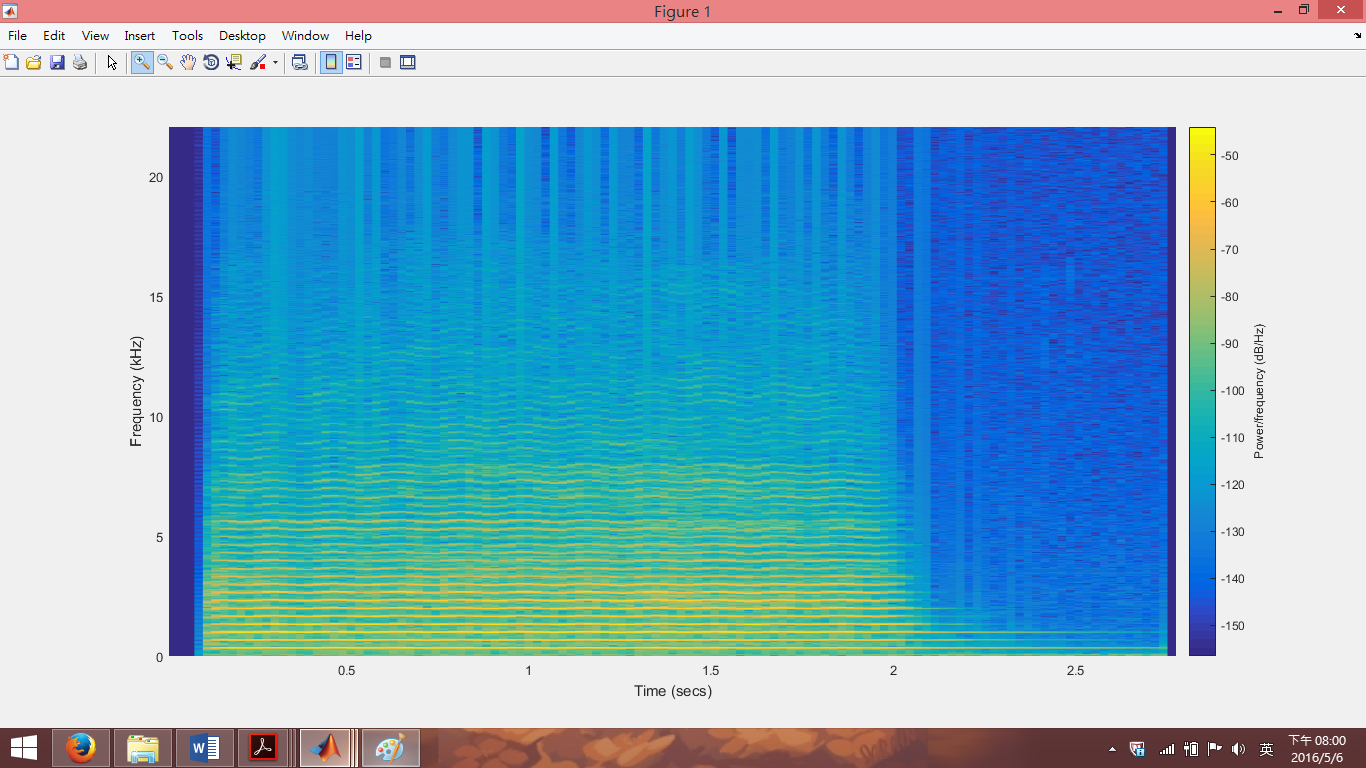


這是把小於1200頻率部分取掉之後的頻譜圖



這則是把大於1200頻率取掉之後的頻譜圖(注意刻度差異)

從上面我們大致可以看到，Pitch id = 64對應的330為基頻以及其泛音所產生的頻譜圖。另外值得一提的是，我試著把imagesc函數中的尺度改成log scale之後再次把原音檔的spectrogram畫了一次，得到下圖的結果。可以看到在很多時間點的地方都有出現橫跨所有頻率的低能量雜訊。

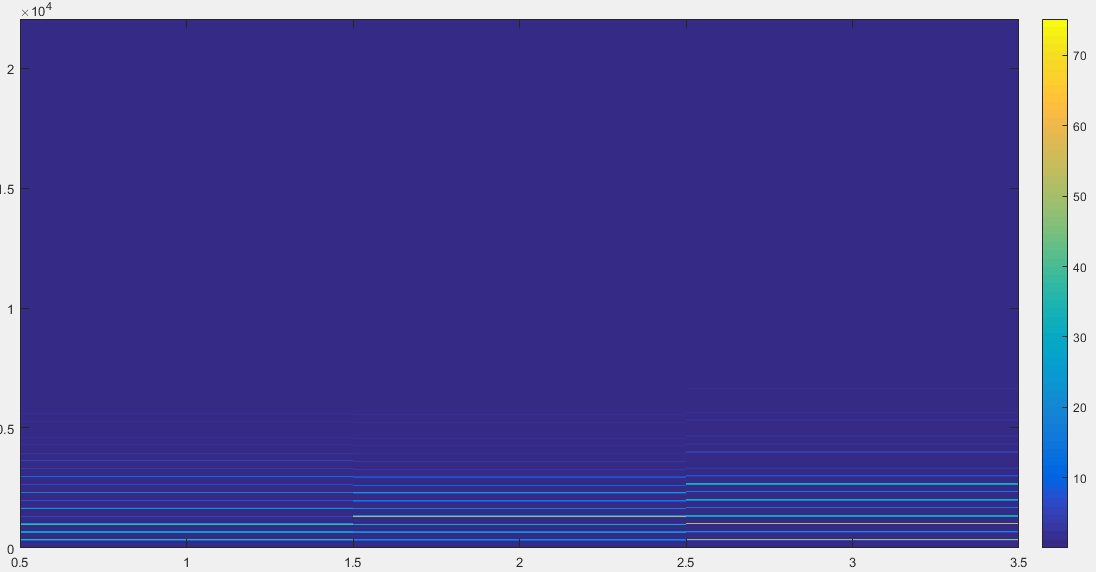
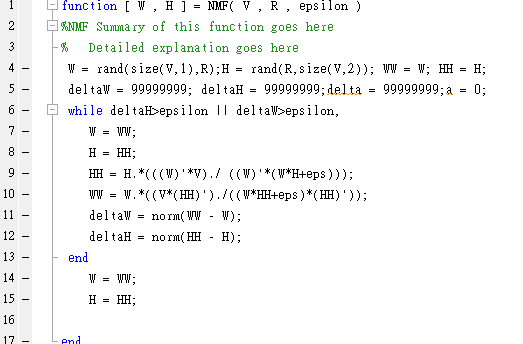


至於三個音檔之間，直接聽起來的話可以感覺到vio\_64\_hp只有細微、高頻而且還有明顯快速震動的vibrato的聲音。vio\_64\_lp聽起來則只剩下較低頻率的嗡嗡聲。然而兩者其實都已經聽不太出來是小提琴的聲音了(高頻的還多少比較可以聽出來)，可以猜測樂器的音色其實跟整個泛音的結構都有關係，刪除基頻和附近的泛音或者是只保留基頻和附近的泛音都會讓音色出現很大的變化。

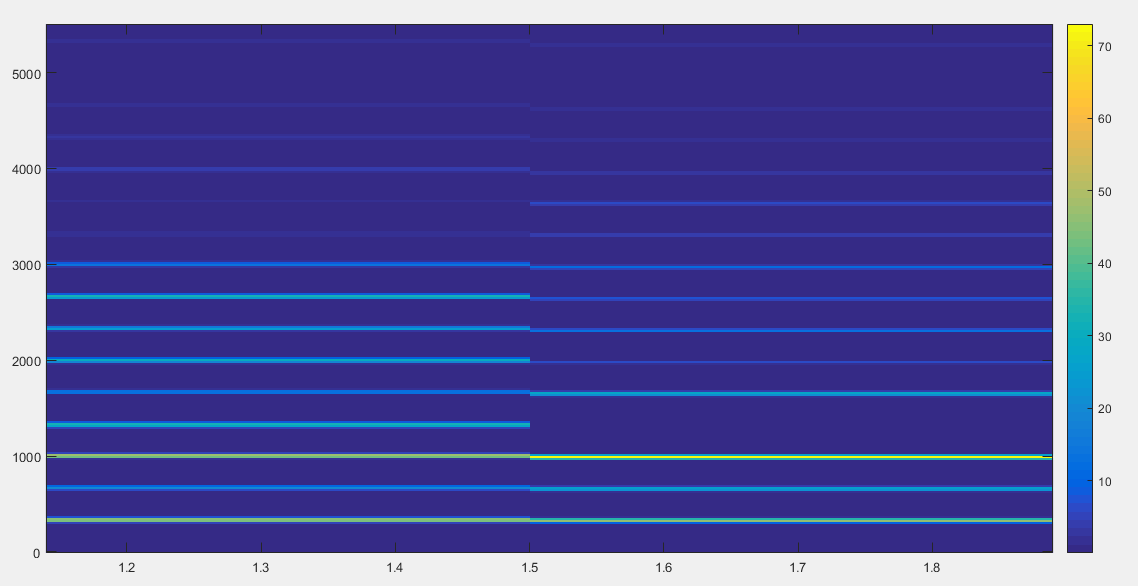
vio\_64\_hp和vio\_64\_lp兩者和原本音檔之間的SDR分別為30.6156、34.0651，可以得知兩者和原本音檔的相似程度差不多，然而我還測試了直接把原本音檔做stft -> istft後再跟原本音檔做比較，發現SDR僅有73.9281，我認為這應該是在轉換過程中的誤差，導致得出的SDR不如第一題中我們直接把兩個相同音檔拿去算SDR所得到的值。

B3、

這題則是要我們實作NMF演算法然後試著去分解音檔vio\_64.wav。我的實作方法基本上就跟講義上寫到的pseudo code一樣。

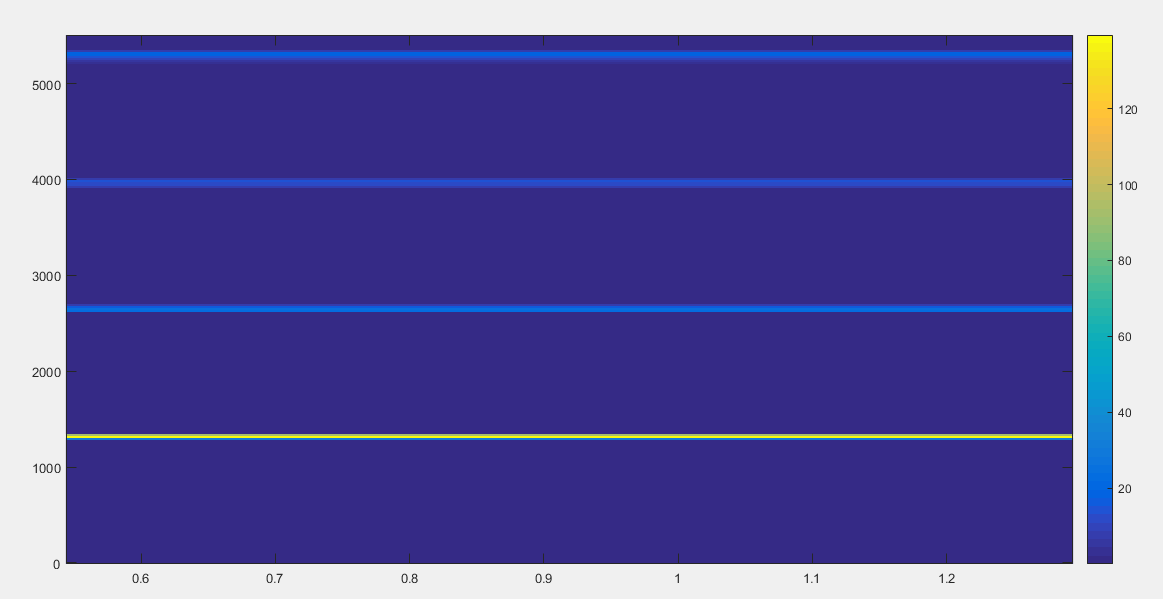
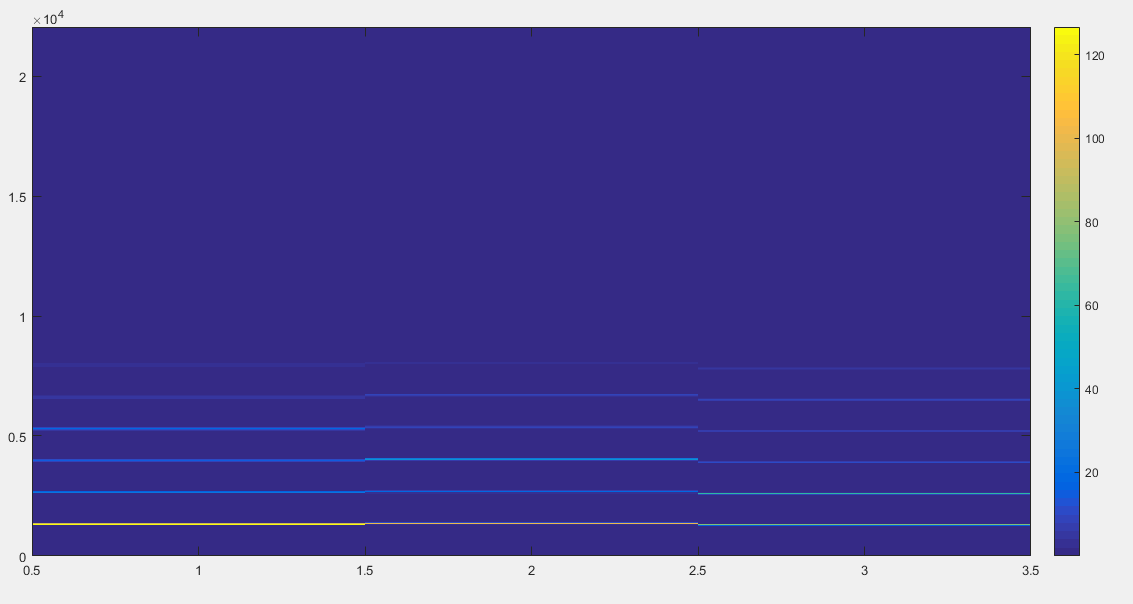
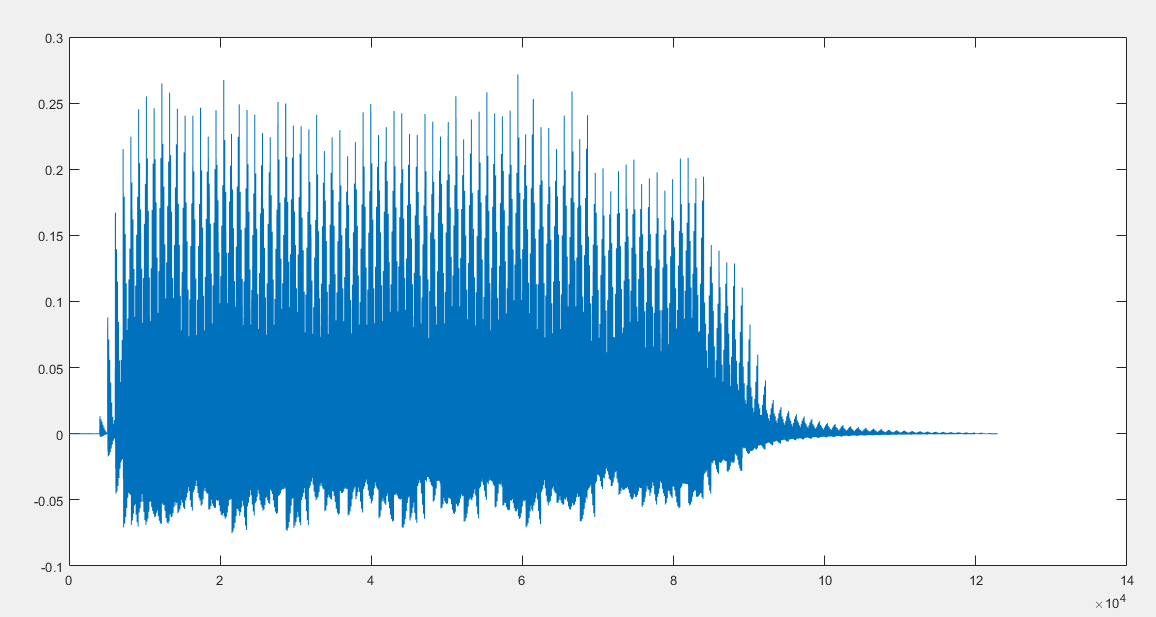


vio\_64.wav用NMF算出來的3個template



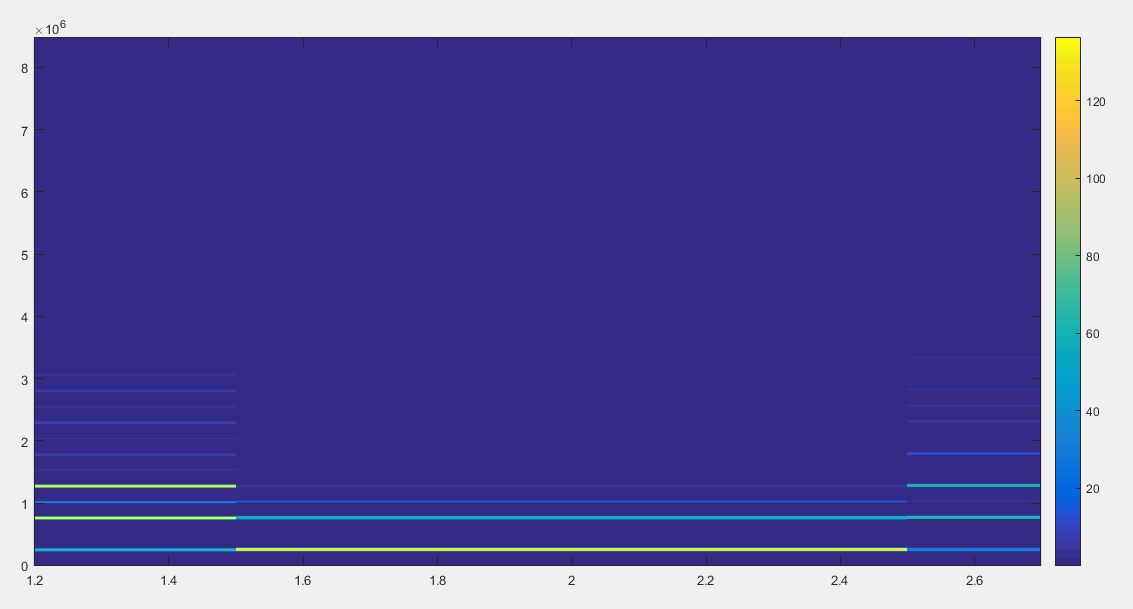
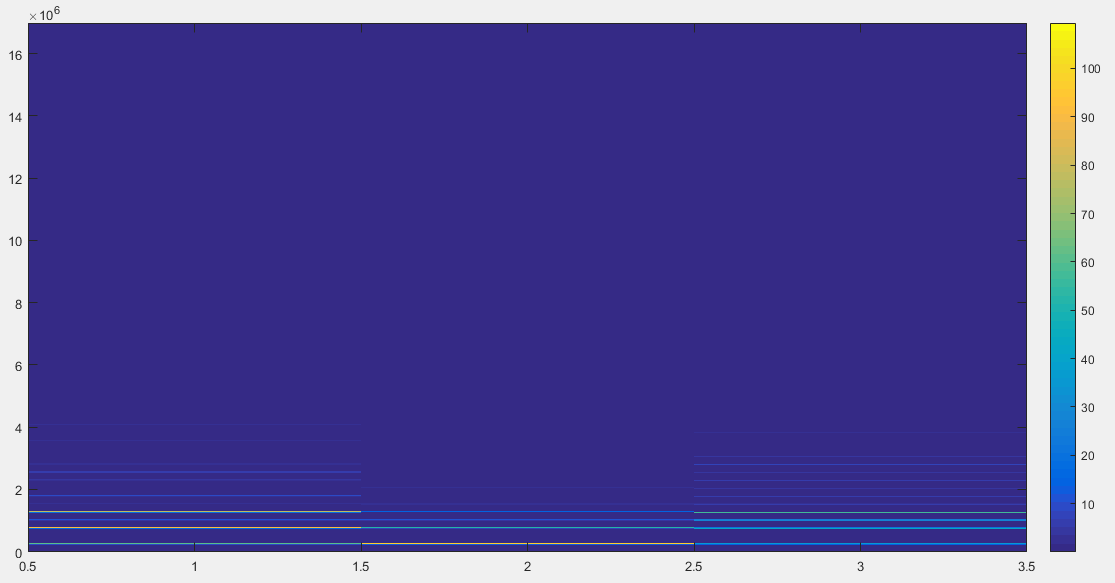
放大觀看

從上圖可以看出，其實算出來的template其實大致是有和原音檔吻合的(Pitch id = 64，基頻為330的音色)。不過三個template之間大致上雖然幾乎一樣但是還是有一些些微的差異。如果把重構後的音檔(W\*H)用ISTFT轉回時域之後畫出來，大致上會像下圖。



Vio\_88.wav做NMF分解的三個template

不難看出，因為pitch id =88比64還要高上不少，因此基頻的位置自然比vio\_64.wav的template還要高。



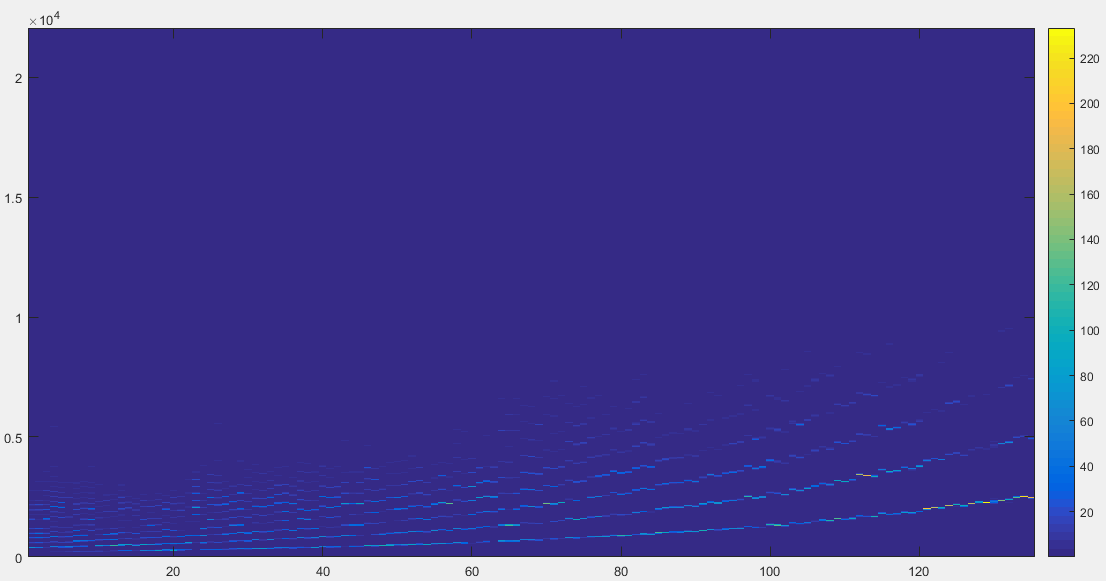
cla\_64.wav的template

可以看出儘管同樣是pitch id =64的音檔，但是單簧管學出來的template卻跟小提琴學出來的頻譜不太一樣。很明顯，因為單簧管是Open Tube的樂器，第1、3、5…倍的泛音基本上是會消失(或者是很明顯較弱)在學出來的template

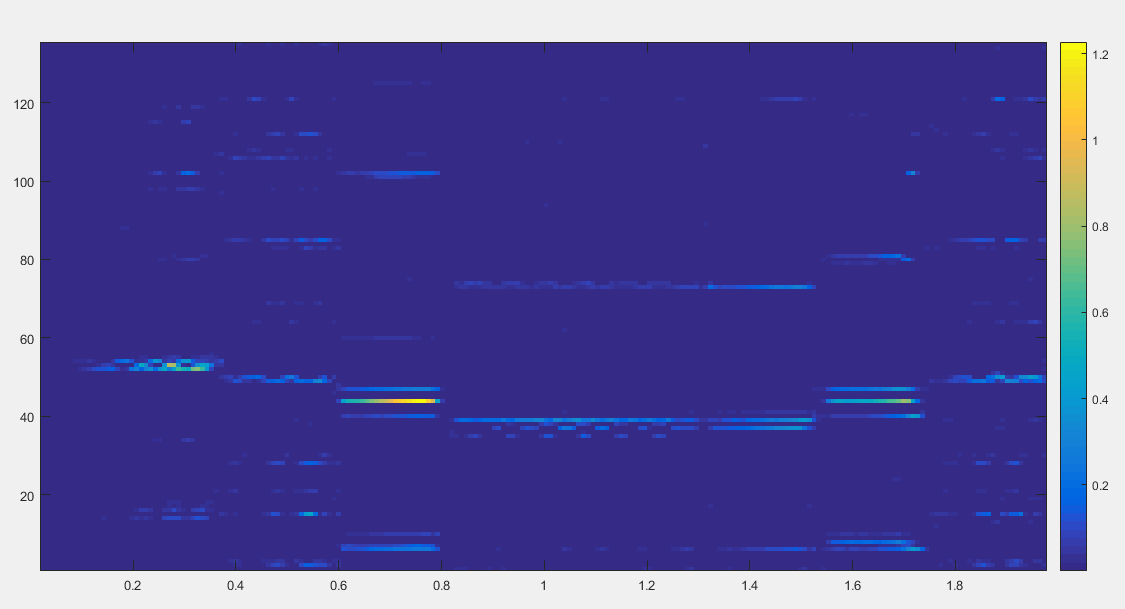
之中的(如上圖箭頭所示)。

R1、

這題是教我們更改剛剛寫出來的NMF演算法去從小提琴所有Note的音檔中個別學出三個template，總共135個template的dictionary出來，並且用這個dictionary去拆解validation set中的01\_vio.wav這個檔案。看看固定template matrix時去算出的來的activation matrix是否可以讓兩者相乘有效的還原出音檔。因為這題必須要固定W只更改H，所以稍微把NMF的演算法做了些修改。



小提琴的Dictionary



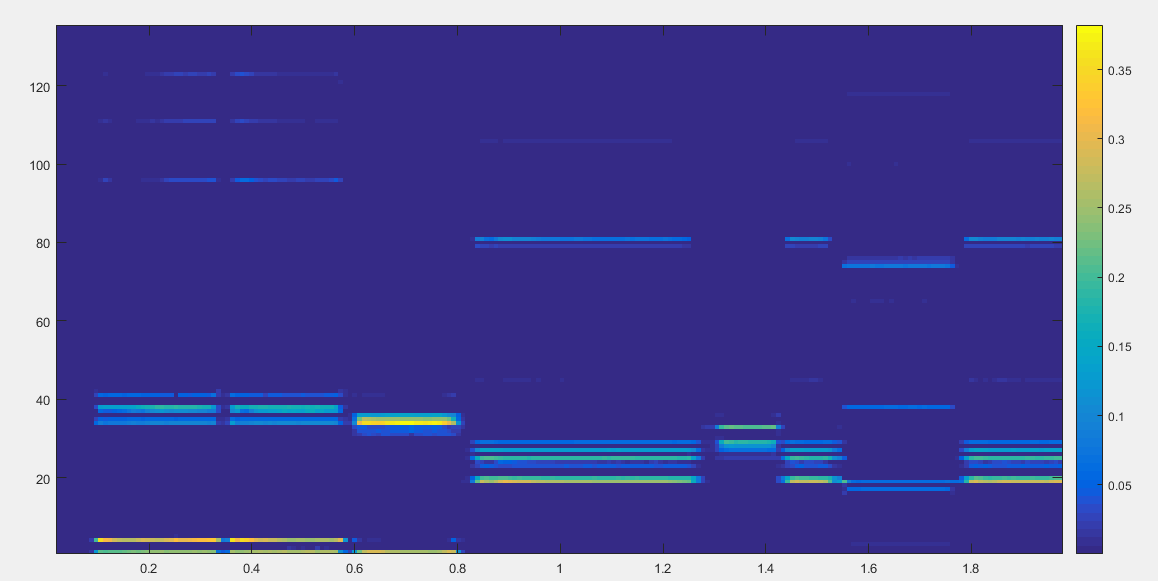
用上圖的dictionary去拆解01\_vio.wav所算出來的activation matrix

從Dictionary中可以看到，從左到右整體的note的頻譜逐漸升高，而且上升的方式是非線性的Pitch Scale。而Activation matrix裡，紅線框起來的部分可以很明確的跟音檔中的旋律吻合。把兩個矩陣相乘，接著再把從原曲的頻譜鐘用angle()函數提取出來的phase資訊加回去，接著做ISTFT即可得到用template和activation matrix重建出來的音檔。

重建出來的音檔，音色可以很明確地聽出來是小提琴的音色，旋律也可以聽得出來跟原本的音檔01\_vio.wav是一樣的，只是會有雜音，而且是在每次有Vibrato(抖音)的技巧的時候雜音會特別明顯。SDR則是9.7051。如果在做ISTFT之前不把Phase加回去的話效果會明顯地變差很多，雜訊量大幅增加，SDR也變成-12.8497。

R2、

這一題則是讓我們試著使用上一題學出來的小提琴Dictionary去分解單簧管的音檔01\_cla.wav。以下是用小提琴的Dictionary當W用NMF算出來的H。

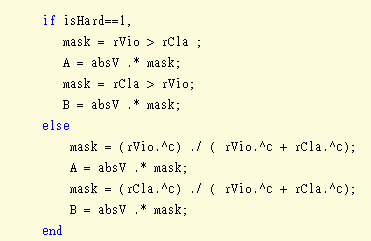


重建出來的activation matrix如上。可以看到紅線框起來的部分其實是跟原音檔一樣的旋律，只是很奇妙的是全部的note的上下都多出了很多其他的note同時出現，導致原本只有一個單音的地方全部都變成了數個note同時出聲。

用W\*H算出來V之後加回原本的Phase之後用ISTFT重建，可以得到用小提琴的Template去模擬出來的單簧管聲音。整個音檔從頭到尾都很不穩定，有很大的雜訊干擾(如同activation matrix所示，每個音出現時都是數個相近的note同時發聲)，不過仔細聽的話會覺得，其實背後好像有類似單簧管的聲音藏在裡面，只是有很強大的干擾。所以雖然這個音檔是用小提琴的Dictionary去重建出來的，但是聽起來音色卻是偏向單簧管的。

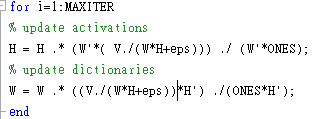
R3 & R4、

這題是要我們真正的去做Source separation的演算法了。首先照著之前學出小提琴dictionary的方式同樣去學出單簧管的dictionary。接著個別用兩個dictionary去對01\_mix.wav這個音檔做NMF(也就是用兩個Dictionary去當作固定的W去算出H)。接著把W和H相乘，接著用Wiener Filter去做Masking，算出個別音色的Spectrogram之後把Phase加回去再做ISTFT就可以得到分離後的兩種樂器的音檔了。我用以下的方式控制Wiener Filter的Soft/Hard以及參數c，以方便找出最適合的分離方法。



經過試驗後，發現Hard Mask的效果比較好(分離出來的音檔跟原音檔的平均SDR值較高)，所以最後的Wiener Mask也是用Hard Mask。C=1或C=2則幾乎沒什麼差，於是我都把C設成1。

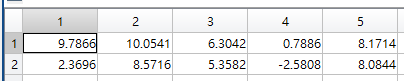
至於NMF的部分，我發現學習Dictionary時使用KL Divergence當作Cost function的效果會稍微好於用Euclid Distance當作Cost function。所以把NMF的iteration改成以下：



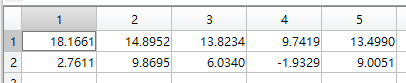
NMF分解時，每個Note的template數量的部分，我發現使用四個以上的template去解NMF的話會解不出來所以無法使用，使用兩個話效果差別不大，使用一個的話效果則會明顯變差，所以最後我還是保留每個Note學出來的Template數都是3個。

所以在R3跟R4中，我最後用的演算法都是如上所示。

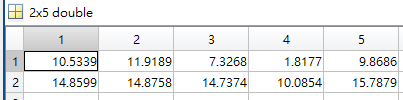
R3中，Validation set中的音檔分離出來的兩個樂器分別對應的SDR、SIR、SAR如下所示(第一列是小提琴，第二列是單簧管)：



SDRs



SIRs



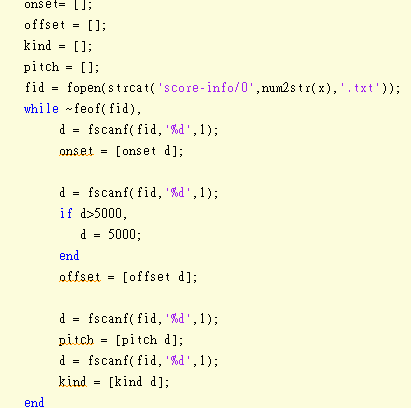
SARs

可以看到，在SDR(分離出來的跟原音檔有多像)的部分很明顯小提琴的表現優於單簧管的。也可以注意到第四首是最難分離的(聽了一下音檔我覺得應該是因為音檔的前半兩種音色的Pitch幾乎重疊了)。SIR(分離音檔跟其他音源有多像)的部分也是小提琴比較高。SAR(跟不是任何待分離音源的相似度)則是單簧管比較高。綜合以上觀察，可以很明顯地得出：小提琴的分離效果是優於單簧管的這樣的結論。

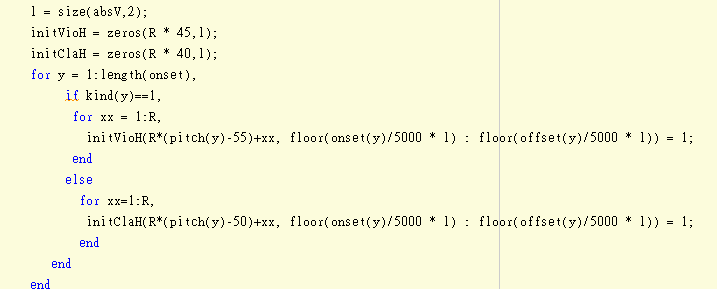
Test Set的部分，因為沒有分離好的音檔的關係當然就沒有辦法計算SDR值，就留待老師幫忙計算囉。

A1、

我還額外做了進階題的第一題。這題是要我們在有樂譜資訊的情況下去初始化Activation matrix，看看最後用這樣初始值去iterate出來的H的效果如何。

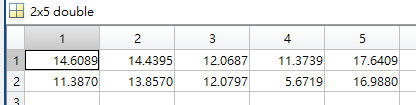


首先用以上的code把txt檔中的樂譜資訊讀入。

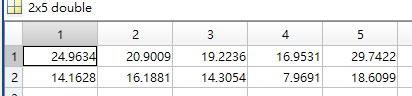


接著依照原本頻譜時間軸的長度把樂譜的時間長度和頻譜的x軸長度統一，並把介於Onset跟Offset時間內的initial H設成1。

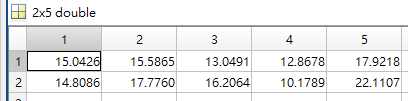
剩下的部分就跟前面R3的部分一樣了。Score informed NMF的結果(SDR、SIR、SAR)如下：



SDR



SIR



SAR

可以看到，整體的表現明顯都變好了許多。聽了一下音檔也可以發現，分離的效果真的得到了大幅的提升，除了少許的雜訊外幾乎是分離的很乾淨，可見做NMF時矩陣的初始值對整體效果的影響之大。