Lab1

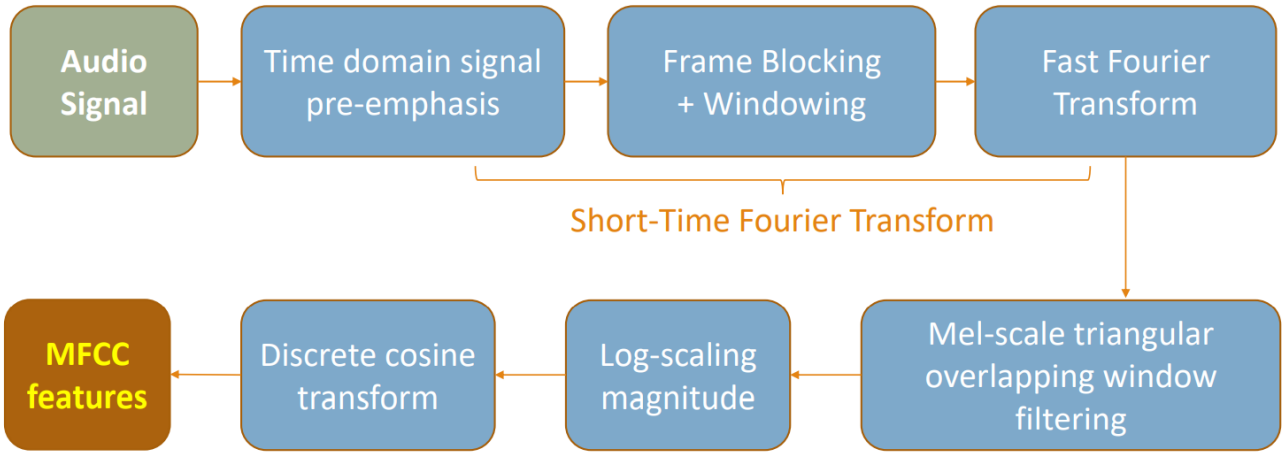
109061217 林峻霆

● Design Specification

實作一個能提取訊號的「梅爾倒頻譜係數」(Mel-scale Frequency Cepstral Coefficients，簡稱 MFCC) 的系統，並觀察系統中頻譜的變化並進行比較與討論。

● Design Implementation

這個部份我會著重於講解各個部分是如何實作以及他們背後各自的原理，產生的頻譜圖、MFCC 以及 bonus 的問題討論將留在 discussion 中探討。這個 Lab 要實作的系統如下：



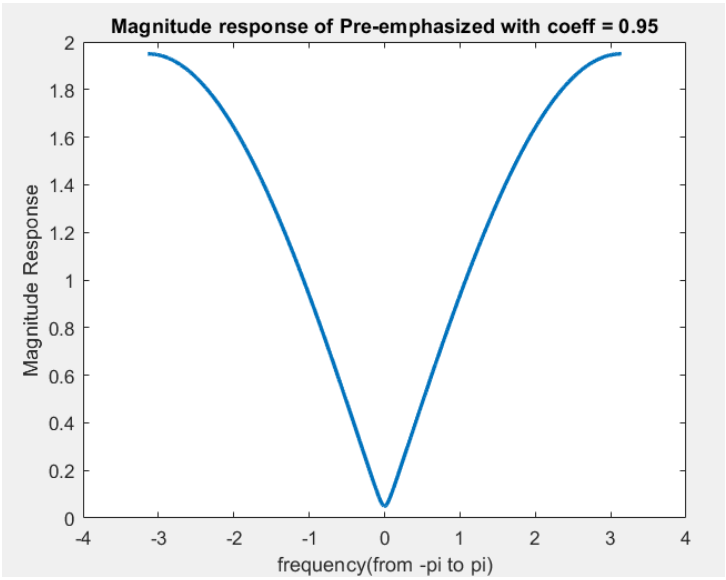
取自助教提供的講義

1. Pre-emphasis

聲音訊號需先經過一個 pre-emphasis filter，他的 impulse response 是

$$H[n] = 1 - \delta[n-1]$$

在頻域上，他就是一個 high-pass filter。



這個步驟的目的在於補償訊號受到發音系統(ex. 聲道、嘴唇)所壓抑的高頻部分，以及凸顯高

頻處的共振峰(語音訊號中帶有較多資訊的部分，能提供語音辨識所需要的資訊)

2. Short-Time Fourier Transform(STFT)

STFT 可分為兩個步驟：windowing + FFT。相較於傳統的將整個訊號丟下去做 FFT，STFT 將訊號透過 windowing 切成一個一個 frame，再將每個 frame 拿去做 FFT。這樣做的好處在於我們能更清楚的得到一些瞬時的資訊，而不會因為長時間而被其他資訊中和掉。在數學上，各個 frame 的頻譜可以表示為：

$$X[m,k] = \sum_{n=0}^{N-1} x[n+mH] \cdot w[n] \cdot e^{-j2\pi\frac{k}{N}}$$

$$m = \ index\ of\ frame\ , N = \ \#\ of\ FFT\ quantization\ , H = frame\ length$$

另外，每個 frame 之間是會有重疊的(ex. 這次 Lab 的 frame length 為 512 但 frame step 為 256)，使 frame 之間連續性提升，避免因變化過大在頻譜上產生原先不存在的頻率。選擇合適的 window 也能減緩此問題。

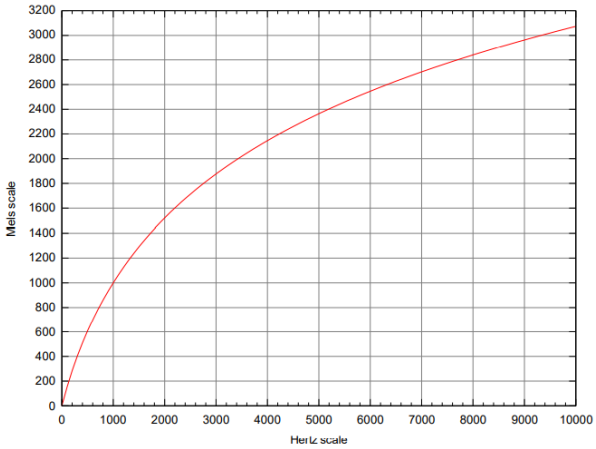
3. Triangular window filtering

首先先介紹 mel-scale，mel-scale 是根據人對不同頻率的的聲音感知不同所設計出的「新頻率」，設計的理念來自於人耳對低頻的聲音較敏感、區分能力強，在高頻則相反。與 Hz 的轉換如下：

- Conversion:

$$Mel = 2595 \times log_{10}(1 + \frac{f}{700\ Hz})$$

- Human perceptual scale of pitches

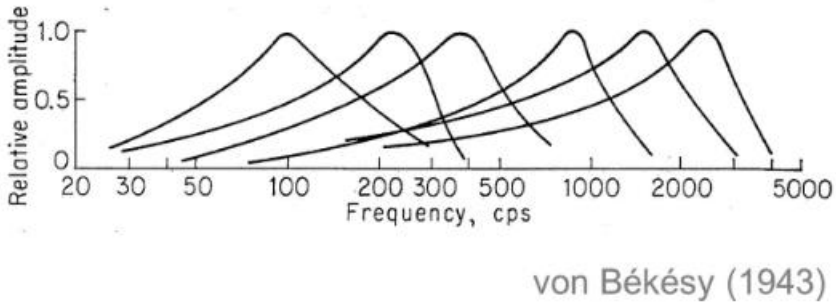


取自助教提供的講義

這部分的實作我認為是這個 Lab 最複雜的部份。首先，我們得先將頻率轉換成 mel-scale(低頻時變化較快；高頻時變化較慢)，並在 mel-scale 下將頻率切成 K 等分，K 是 triangular filter 的數量。接下來，再從 mel 轉回 Hz，並將頻率做 quantization(因為我們採用 FFT，(freq_min, freq_max) -> (0, N – 1))。最後再透過 quantize 過後的 index 去畫三角形。這部分可以透過 Numpy 函式庫中的 linspace function 來實現。

Triangular window filter 的設計想法來自於前人對耳朵構造的研究，發現其對不同頻率的單

音的頻率響應如下圖：



取自劉奕汶教授提供的講義

4. Log-energy and Discrete-Cosine Transform(DCT)

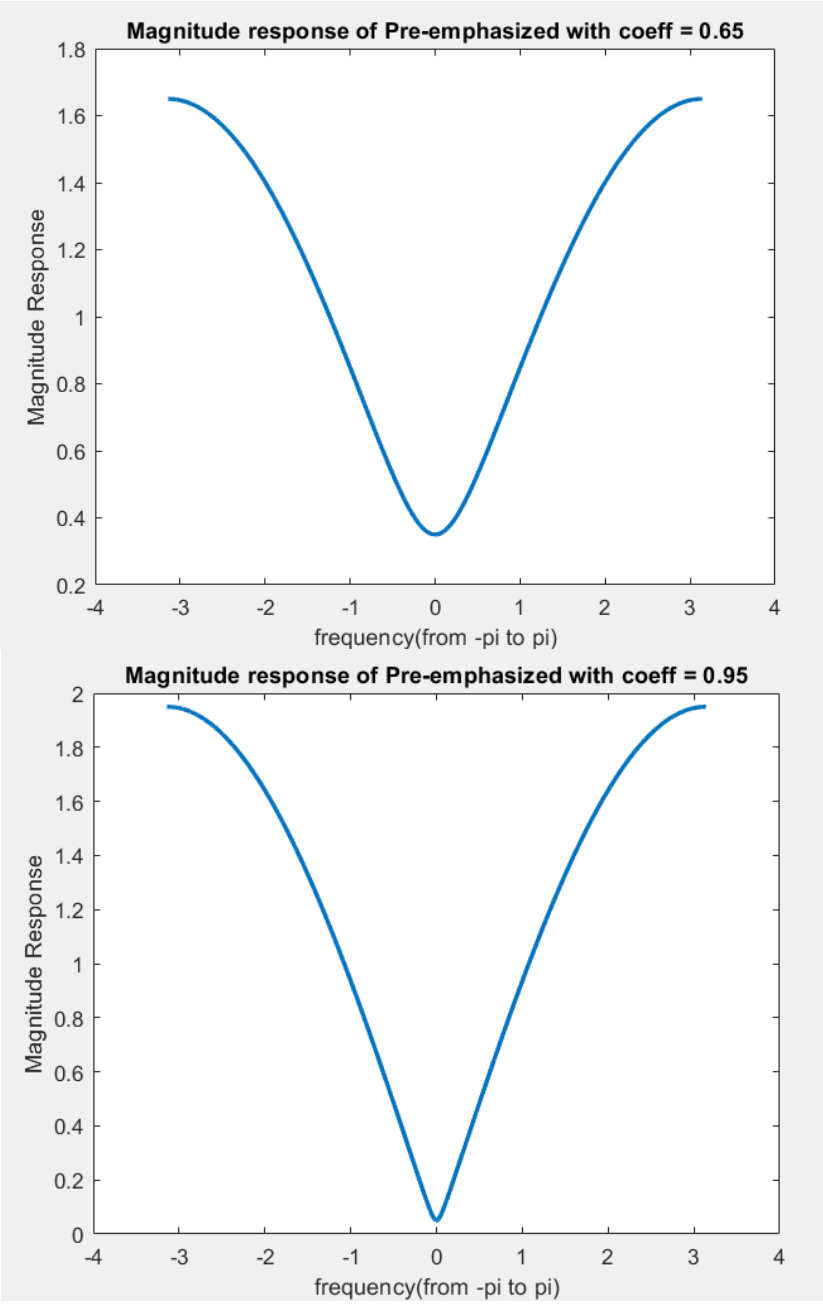
將通過 triangular window filter 的訊號取其的 log scale energy，在通過 DCT 轉換到 Quefrency domain(倒頻譜)，倒頻譜中各 quefrency 的 amplitude 就是這個 frame 的 MFCC

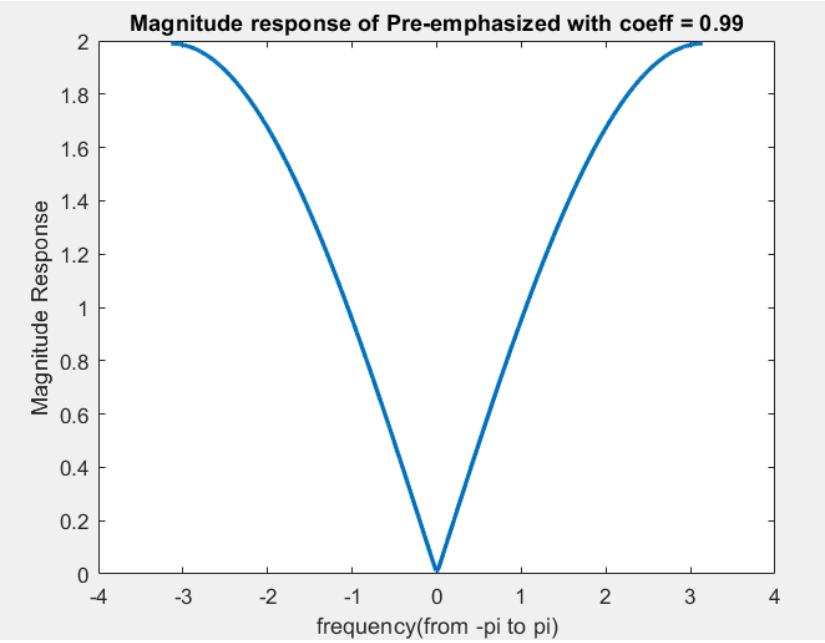
● Discussion

這部分我們就來比較我們得到的一些結果(頻譜圖)，並討論一些實作時的問題。

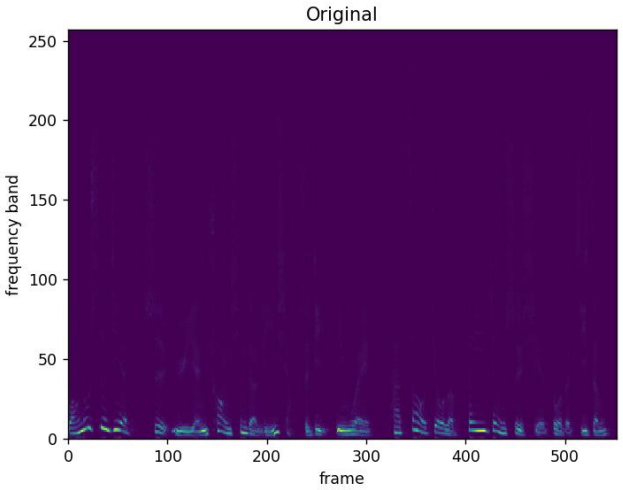
1. Pre-emphasis

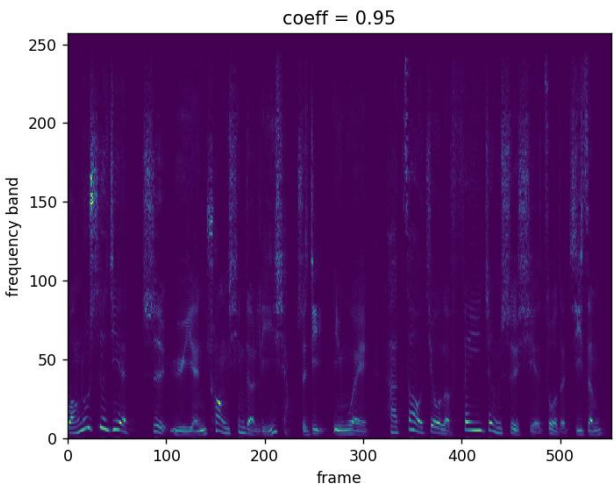
在這個步驟中，我們將前後兩個 sampling point 相減，當作我們要處理的訊號。直觀上來看，這個動作就是在消除訊號中的直流成分(前後 sampling point 不變的部分)，而從數學的角度來看，它的 frequency response(如下圖)確實是一個 high-pass filter。當係數α越大，high-pass 的特性越明顯。如同前一部分所述，這個步驟的目的在於補償訊號受到發音系統(ex. 聲道、嘴唇)所壓抑的高頻部分，並以及凸顯高頻處的共振峰(語音訊號中帶有較多資訊的部分，能提供語音辨識所需要的資訊)

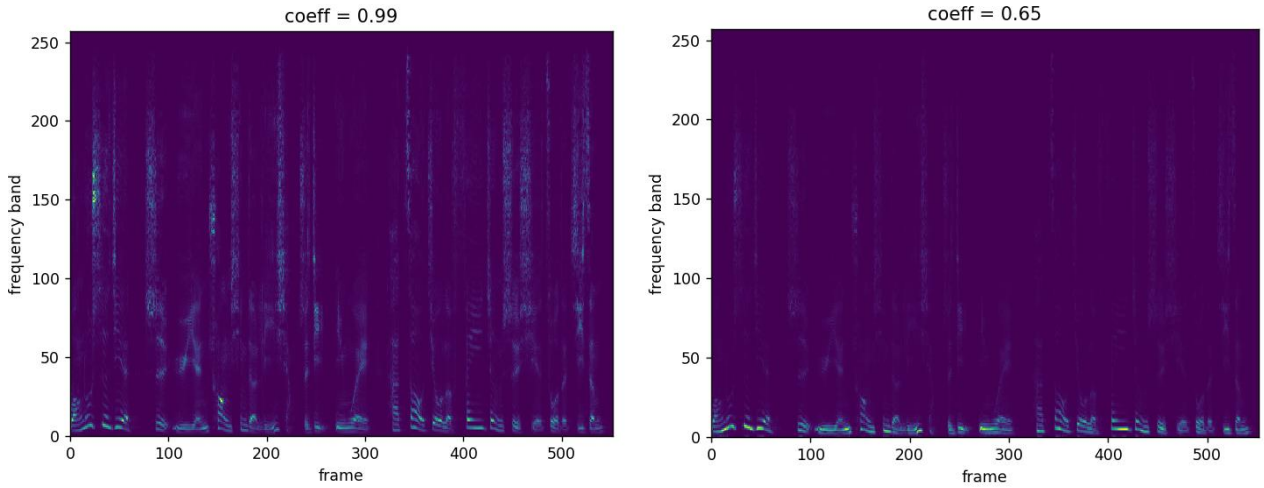

Magnitude response of Pre-emphasized with coeff = 0.65
Magnitude Response
frequency(from -pi to pi)
Magnitude response of Pre-emphasized with coeff = 0.95
Magnitude Response
frequency(from -pi to pi)



從 STFT 得到的頻譜圖可以看出 pre-emphasis 有凸顯高頻共振峰的特性。比較原始訊號的頻譜以及係數為 0.95 的頻譜可以發現，在原先頻譜中看不清楚的峰值部分，在係數 0.95 的頻譜中區隔出來，清楚許多。而比較原先頻譜、係數 0.65 頻譜以及係數 0.95 的頻譜可以發現，pre-emphasis 中的係數決定高頻共振峰的強化程度，係數越高，越明顯。



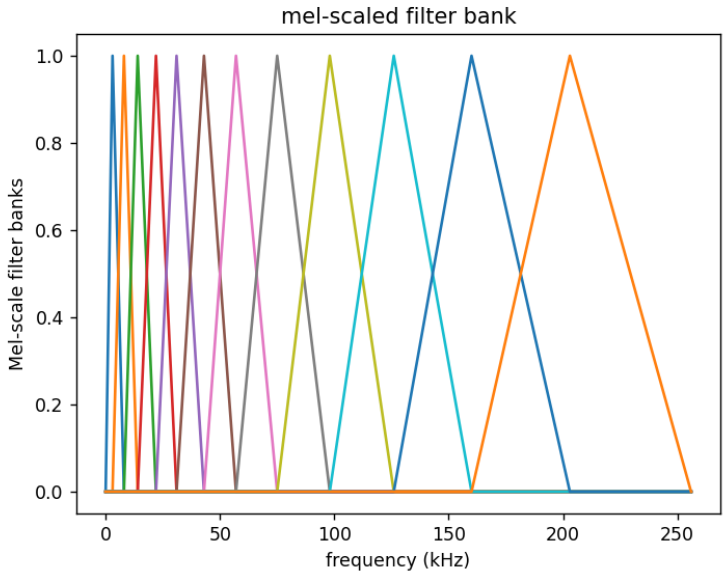


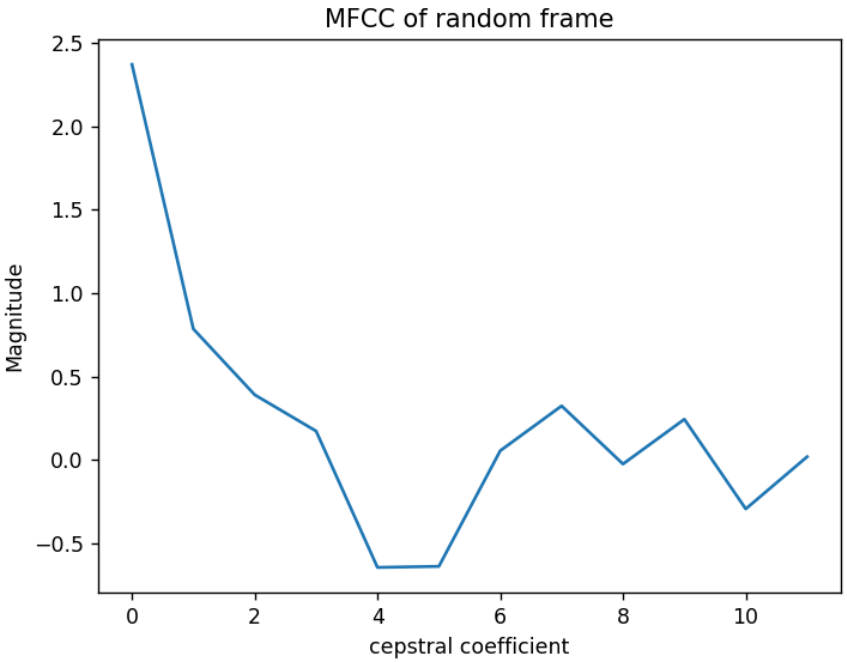


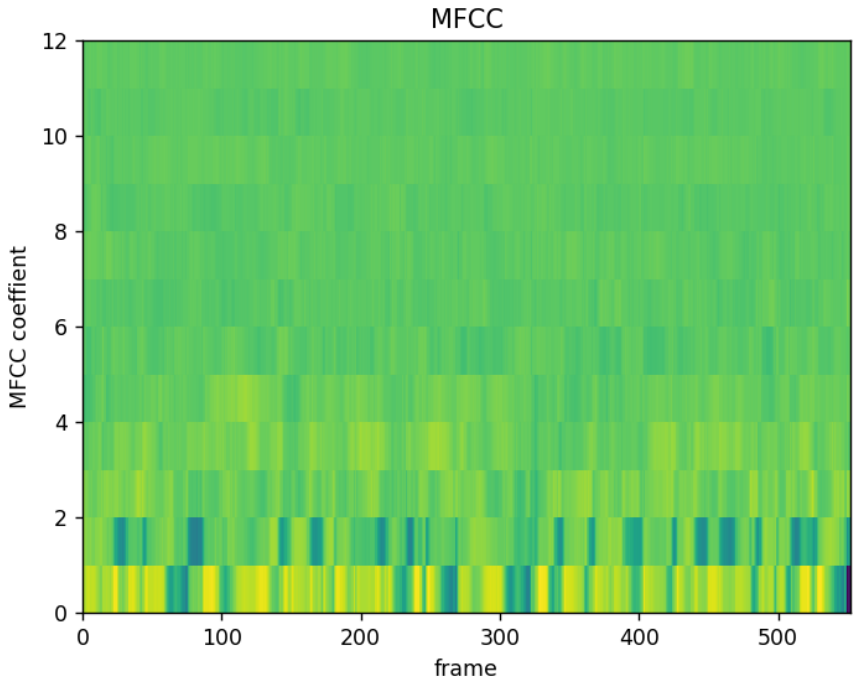
2. Triangular window filter

這部分有兩個問題要討論 : 1. filter bank 以及 MFCC 的數量要挑多少？ 2. FFT quantization 的數量(512)跟 frequency frame(257)的差異來自哪裡？

首先是 Lab 要 demo 的頻譜以及 MFCC：







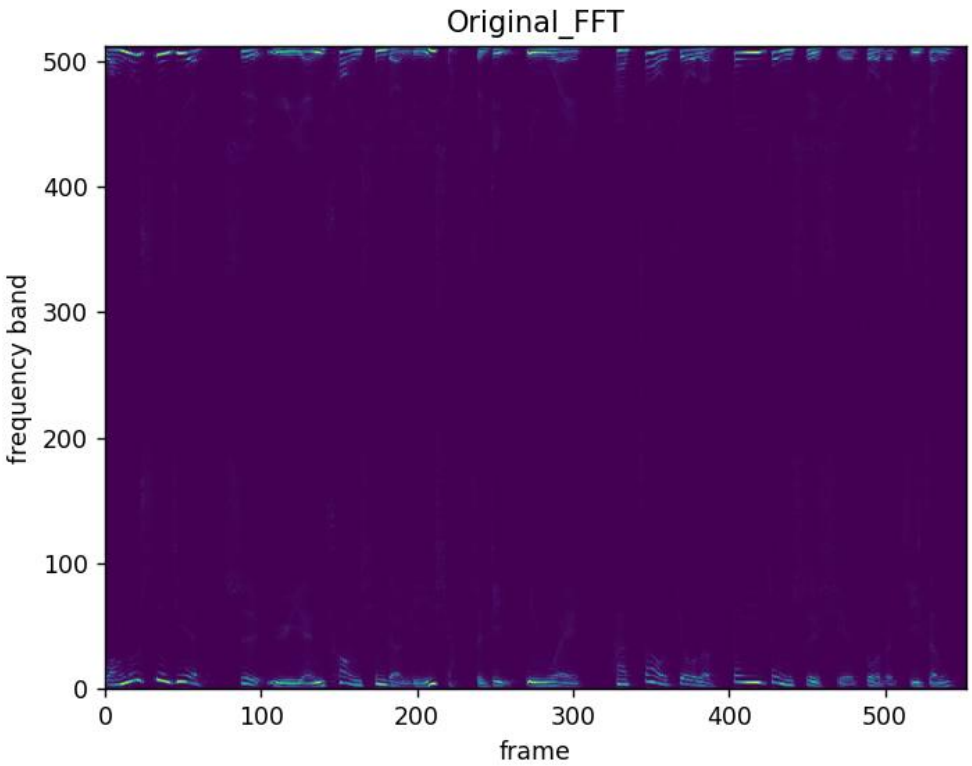
接著我們來解決上面提到的兩個問題，首先是「FFT quantization 的數量(512)跟 frequency frame(257)的差異來自哪裡？」。首先我們看 FFT 的數學公式：

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn} \qquad k = 0, 1, 2 \ldots N-1$$

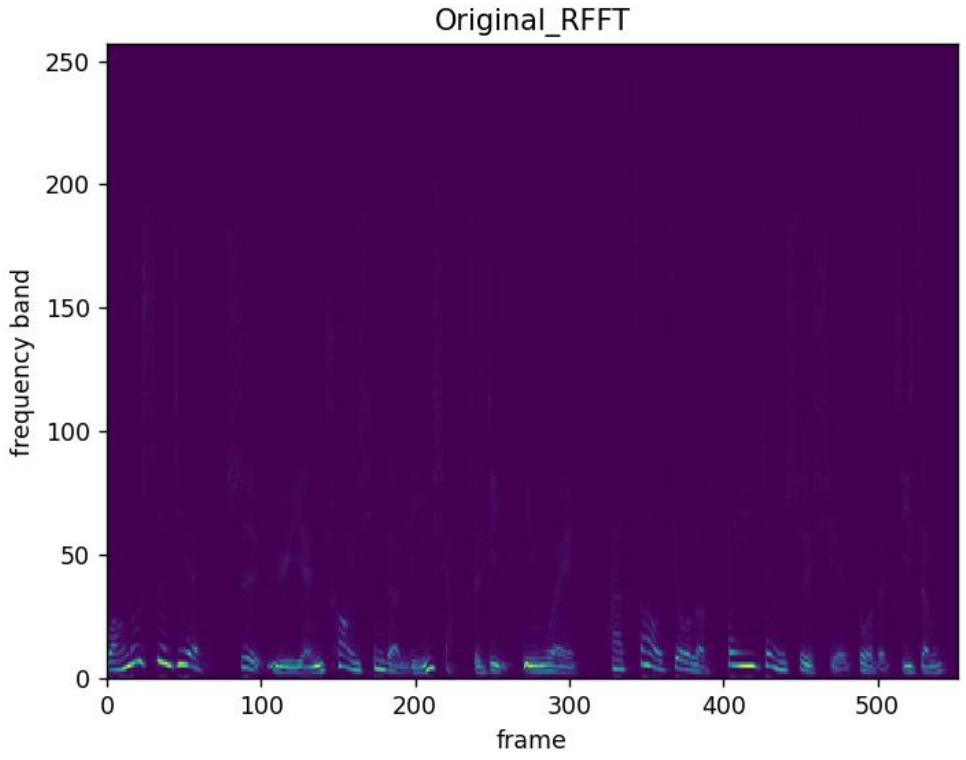
基本上 X[k]的數量與 FFT Quantization 的數量相同，都是 N。當 x[n]是一個 real signal 時，X[k]具有 conjugate symmetric 的特性，也就是 X[k] = X*[-k]，若只考慮能量我們就會有 | X[k] | = | X*[-k] | = | X[-k] |，在 MFCC 中我們只關注頻譜的能量，因此我們可以只取一半的頻譜來提取 MFCC。這也是為何我們的 code 裡面是用 np.fft.rfft

```
# FFT
complex_spectrum = np.fft.rfft(frames, num_FFT).T
```

如果我們將這邊的 rfft 改成 fft，這樣跑出來的頻譜(只經過 pre-emphasis)會變成下圖：

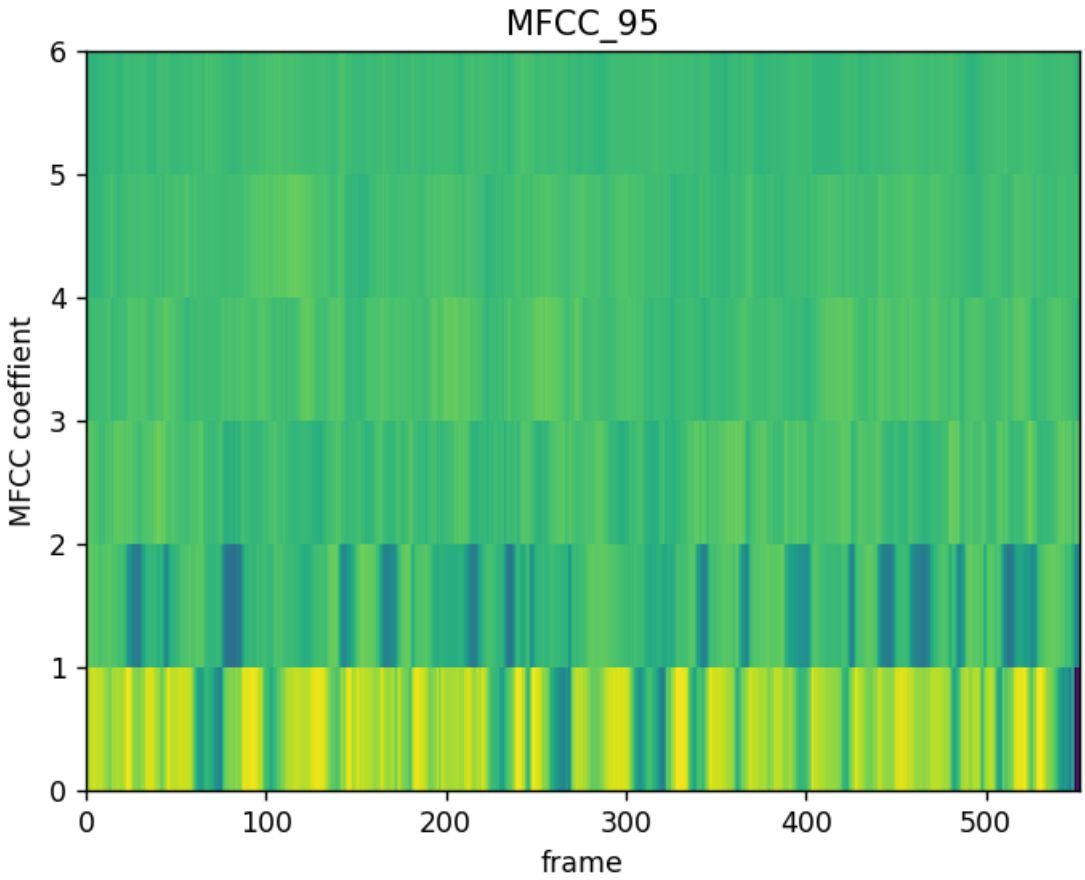


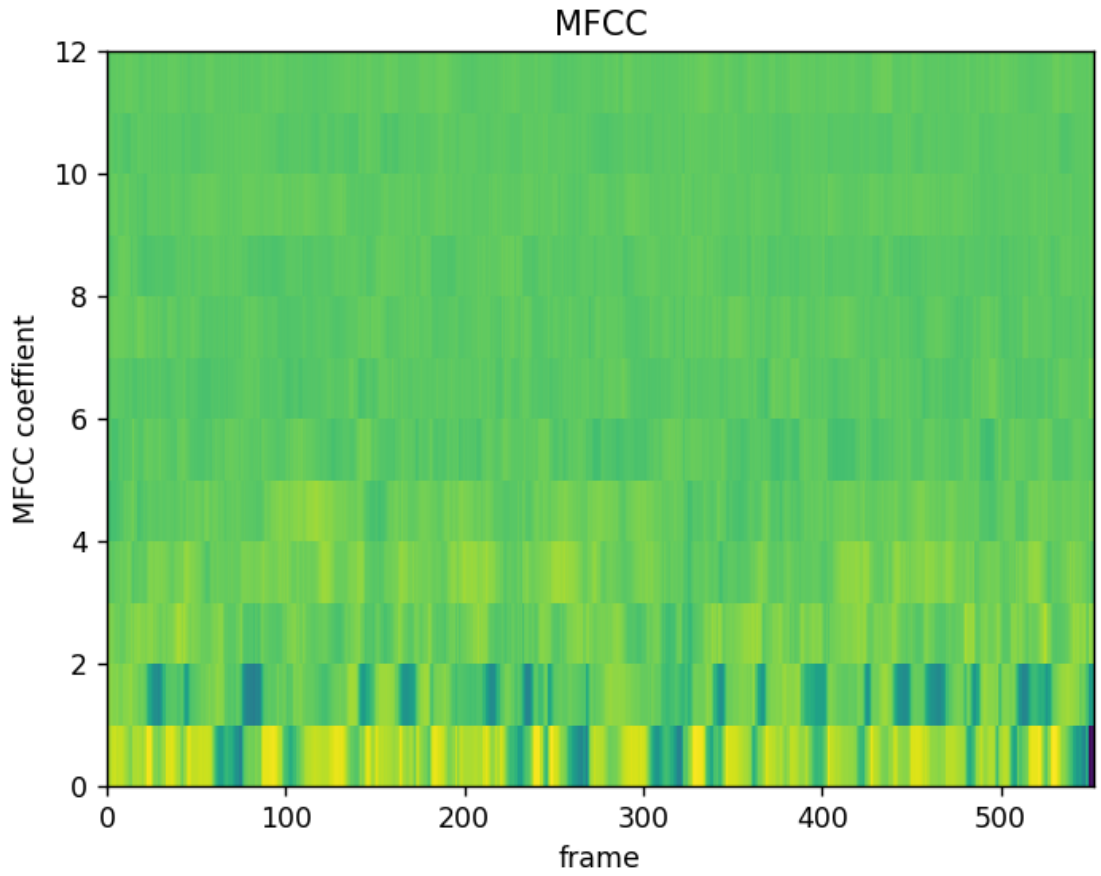
下方則是原先的頻譜(採用 rfft，只經過 pre-emphasis)：

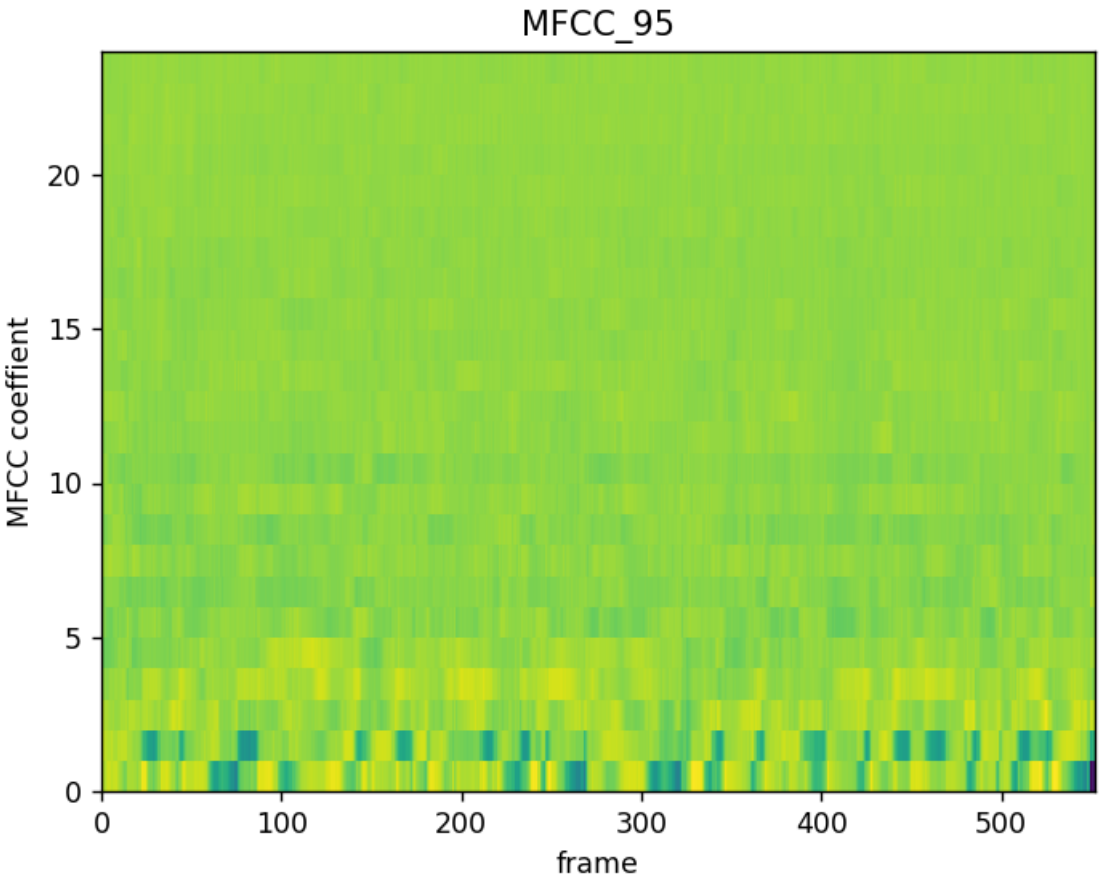


可以發現 fft 的頻譜上下對稱的，它在頻譜上對到的是(0, 2pi)，而 rfft 則為 fft 頻譜的下半部。因此我們可以只取一半(0, pi)來計算 MFCC。這就是為甚麼 FFT quantization 的數量(512)跟 frequency frame(257)之間差兩倍的原因。

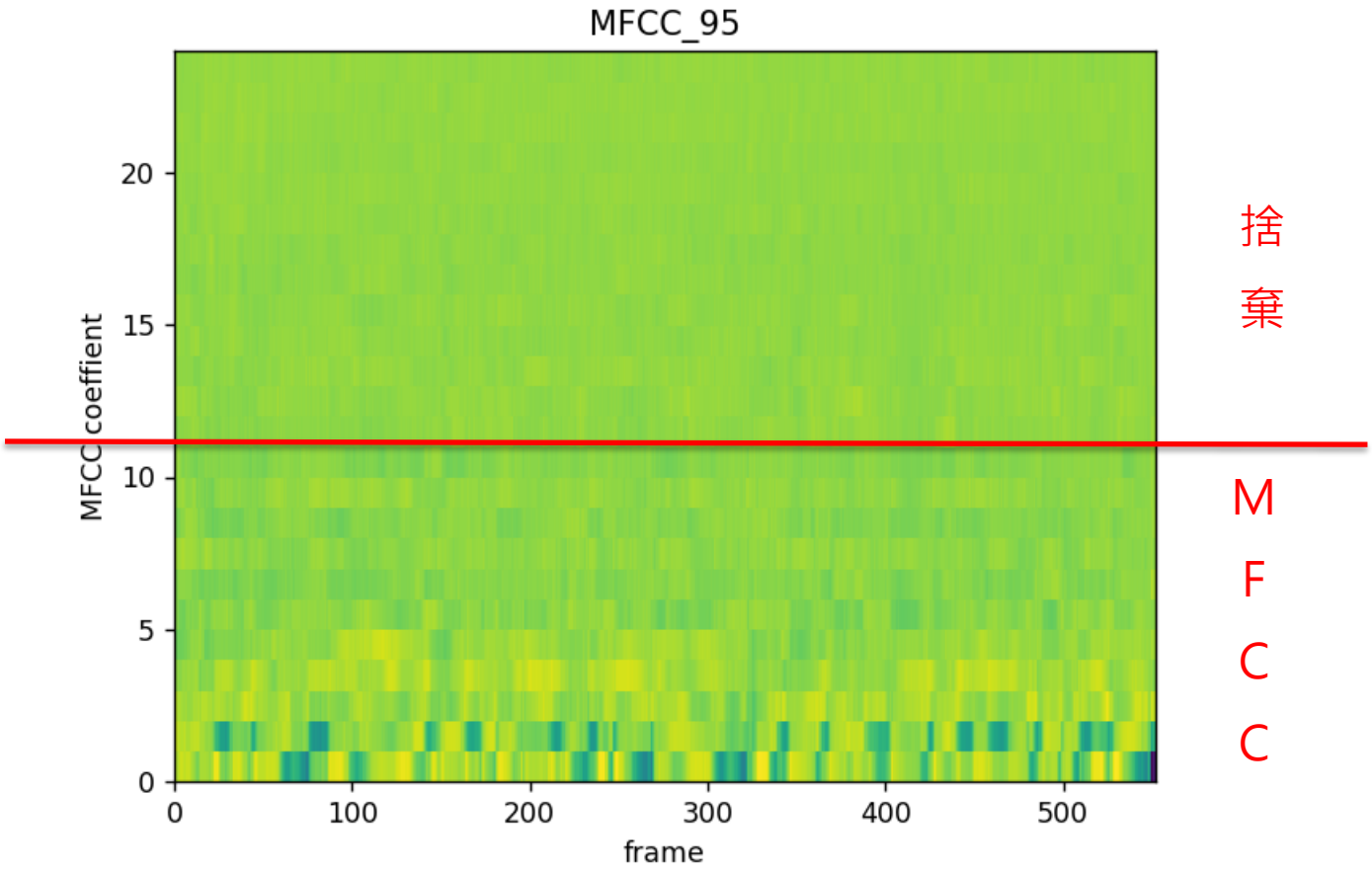
接著我們來解決另一個問題：「要取幾個 triangular filter 以及 MFCC？」我們先思考 filter bank 的數量與什麼有關，答案是 max_freq 與 min_freq 在 mel-scale 下的等分點的數量，也就是說提高 filter bank 的數量可以提高 frequency resolution，如果我們今天要分析的音訊其頻率十分靠近，那 filter bank 的數量可能就得提升，以區分這些頻率。此外，filter bank 的數量也跟我們後續希望得到的 MFCC 數量有關，filter bank 的數量要大於等於 MFCC 的數量。以這次 Lab 為例，我們跑 3 個 case：# of filter bank = 6, 12, 24




MFCC
MFCC coeffient
frame
0
2
4
6
8
10
12
0
100
200
300
400
500


MFCC_95
MFCC coeffient
frame
0
5
10
15
20
0
100
200
300
400
500

從圖上就可以看出，y 軸同樣是 0 到 pi(因為使用 rfft)，filter bank 的數目越多 frequency resolution 越好。我認為 frequency resolution 要能高到幫助我們看出 MFCC 頻譜有明顯差異，進而找到合適的 MFCC 數量。以這個 Lab 為例，在 # of filter bank = 24 的圖中我們可以發現，在第 11 個 frame 之上就基本上沒有變化了，表示後方的 MFCC 對於語音辨識沒有很大的資訊量。因此我們可以取 24 個 filter bank 以及 10~12 個 MFCC 來當作這個語音訊號的特徵，來幫助我們做進一步的語音分析。



決定 filter bank 以及 MFCC 的數量就是在 frequency resolution 以及計算複雜度之間做 trade off，frequency band 越多 frequency resolution 越好，但同時我們要做的計算量就越大，花費時間更長。

◆ Conclusion

這個 Lab 介紹一種常運用在語音處理、辨識的技術—MFCC，從人耳的構造對各頻率的頻率響應為初衷，建立起數學模型模擬人耳的聽覺感受，並運用 python 實作這個系統，提取語音特色 MFCC。實作過程中讓我一步步了解這個系統的原理，並去思考和討論一些關於這個系統的一些問題。

◆ References

- YouTube 頻道：Valerio Velardo - The Sound of AI
  https://www.youtube.com/c/ValerioVelardoTheSoundofAI
  完整介紹 MFCC 各流程的細節，以及各個 stage 背後的數學。
- Numpy 函式庫 https://numpy.org/doc/stable/index.html
  查詢函式庫的內容，幫助trace code和實作
- 教授與助教的講義