多媒體資訊概論

**前導練習**(20%)

* + 將 a.wav 音訊檔讀出, 未作更改, 寫入 b.wav
  + 學習使用buffer進行 binary file 讀寫的技巧
  + 想像是一個 “copy”指令程式？
  + 使用 a.wav = “Nello.wav” (8bit, mono) 測試

import wave

from scipy import \*

from pylab import \*

filename = 'Nello.wav'

wavefile = wave.open(filename,'rb') # open for writing

nchannels = wavefile.getnchannels()

sample\_width = wavefile.getsampwidth()

framerate = wavefile.getframerate()

numframes = wavefile.getnframes()

comptype = wavefile.getcomptype()

str\_data = wavefile.readframes(numframes)

print(nchannels)

print(sample\_width)

print(framerate)

print(numframes)

wavefile.close()

path = 'copy.wav'

copyfile=wave.open(path,'wb')

copyfile.setnchannels(nchannels)

copyfile.setsampwidth(sample\_width)

copyfile.setframerate(framerate)

copyfile.setnframes(numframes)

copyfile.writeframesraw(str\_data)

copyfile.close()

copyfile=wave.open(path,'rb')

print(copyfile.getparams())

copyfile.close()

因為沒有很好的讀懂wave，所以用了比較笨的方法一步一步來。讀懂之後用了更聰明的方法，但也遇到setnnframes()沒有成功把值進去的情況，之後把資料另外讀進去就解決了。

import wave

from scipy import \*

from pylab import \*

filename = 'Nello.wav'

wavefile = wave.open(filename,'rb') # open for writing

#讀取wav檔案的四種資訊的函式。其中numframes表示一共讀取了幾個frames

params = wavefile.getparams()

str\_data = wavefile.readframes(params.nframes)

print(wavefile.getparams())

wavefile.close()

path = 'copy01.wav'

copyfile=wave.open(path,'wb')

copyfile.setparams(params)

copyfile.writeframesraw(str\_data)

copyfile.close()

copyfile=wave.open(path,'rb')

print(copyfile.getparams())

copyfile.close()

**離散傅利葉正逆轉換**(30%)

* + 將 a.wav 音訊檔讀出, 進行DFT轉換, a[n] 🡪 A[k]+jB[k]
  + 不經任何處理, 得 C[k]+jD[k] = A[k]+jB[k],
  + 再進行逆轉換 C[k]+jD[k] 🡪b[n], 然後寫入 b.wav
  + 將聲音訊號切割分段處理再組合

嘗試不同長度的分段，令分段長度 N = 128, 256, 512

* + 必須展示任選一段訊號的頻譜 (A2[k]+B2[k])1/2，以證明實作完成
  + 使用 a.wav = “Nello.wav” (8bit, mono) 測試

**頻域的變音處理**(30%)**：**

* + 實作將 a.wav 音訊檔讀出, 進行DFT/ Filtering/ IDFT處理,
  + 將結果寫入 b.wav
  + 設濾波器為 H[k]= 0.98^k 以及 H[k]=0.98^(N-1-k) 或其他
  + 最佳之分段長度，以及最佳之濾波器函數請自行調整
  + 結果佔(20%)，實驗過程佔(10%)，其他處理可獲加分
  + 書面報告討論實作方法、結果與心得
* #include <iostream>
* #include <complex>
* #include <fstream>
* #include <math.h>
* #include <cstring>
* #include <stdio.h>
* #include <stdlib.h>
* #include <windows.h>
* using namespace std;
* #define \_USE\_MATH\_DEFINES
* #define pi 3.141592608
* class DFT\_Project{
* public:
* DFT\_Project(int n);
* ~DFT\_Project();
* void openReadFile(char const name[]);
* void openWriteFile(char name[]);
* void SaveFile(int arr[],int num);
* int\* LoadFile(int arr[],int num,bool option);
* void GetHeader();
* void GetBody(int n);
* void GetEnd();
* complex<float> \*DFT(int arr[],int num);
* complex<float> \*InverseDFT(complex<float> arr[],int num);
* complex<float> \*low\_pass(complex<float> arr[],int num);
* int\* Float\_to\_Int(complex<float> arr[],int num);
* void SetVolumn(float f);
* float \*Change(float arr[],int num);
* void Main();
* void Block(int num);
* private:
* int TotalLength,HeaderLength,BodyLength,EndLength;
* int N;
* int \*header,\*out2,\*body,\*end;
* complex<float> \*out;
* float VolumnFactor;
* ifstream input;
* ofstream output;
* };
* DFT\_Project::DFT\_Project(int n){
* N=n;
* HeaderLength=44;
* header = new int[HeaderLength];
* body = new int [N];
* out = new complex<float> [N];
* out2 = new int [N];
* VolumnFactor=0.8;
* }
* DFT\_Project::~DFT\_Project(){
* delete header;
* delete body;
* delete end;
* delete out;
* delete out2;
* input.close();
* output.close();
* }
* void DFT\_Project::openReadFile(char const name[]){
* input.open(name,ios::in | ios::binary);
* if(!input){
* cout<<"input failed!";
* }
* }
* void DFT\_Project::openWriteFile(char name[]){
* output.open(name,ios::out | ios::binary);
* if(!output){
* cout<<"output failed!";
* }
* }
* void DFT\_Project::SaveFile(int arr[],int num){
* for(int i=0;i<num;i++){
* output.put(arr[i]);
* }
* }
* int\* DFT\_Project::LoadFile(int arr[],int num,bool option){
* char buffer;
* unsigned char ch2;
* for(int i=0;i<num;i++){
* BodyLength=(option)?BodyLength-1:BodyLength;    // reading header and end,length--
* input.get(buffer);
* ch2=buffer;
* arr[i]=(int)ch2;
* }
* return arr;
* }
* void DFT\_Project::GetHeader(){
* printf("geting header\n");
* header=LoadFile(header,HeaderLength,0);
* BodyLength=(int)(header[40]+header[41]\*pow(16,2)+header[42]\*pow(16,4)+header[43]\*pow(16,6));
* TotalLength=(int)(header[4]+header[5]\*pow(16,2)+header[6]\*pow(16,4)+header[7]\*pow(16,6))+8;
* EndLength=TotalLength-BodyLength-HeaderLength;
* }
* void DFT\_Project::GetBody(int n){
* body=LoadFile(body,n,1);
* }
* void DFT\_Project::GetEnd(){
* printf("geting end\n");
* end = new int[EndLength];
* end = LoadFile(end,EndLength,0);
* }
* complex<float>\* DFT\_Project::DFT(int arr\_origin[],int num){
* complex<float> f,ck;
* complex<float> \*arr\_transform = new complex<float> [num];
* for(int k=0;k<num;k++){
* f=0;
* for(int n=0;n<num;n++){
* f.real(f.real()+cos(2\*pi\*k\*n/num)\*arr\_origin[n]);
* f.imag(f.imag()+sin(2\*pi\*k\*n/num)\*arr\_origin[n]);
* }
* arr\_transform[k]=f;
* }
* return arr\_transform;
* }
* complex<float>\* DFT\_Project::InverseDFT(complex<float> arr\_transform[],int num){
* complex<float> f;
* complex<float> \*arr\_origin = new complex<float> [num];
* std::complex<float> temp2;
* for(int n=0;n<num;n++){
* f=0;
* for(int k=0;k<num;k++){
* temp2.real(cos(2\*pi\*k\*n/num));
* temp2.imag(0-sin(2\*pi\*k\*n/num));
* temp2\*=arr\_transform[k];
* f+=temp2;
* }
* f.real(f.real()/num);
* f.imag(f.imag()/num);
* arr\_origin[n]=f;
* }
* return arr\_origin;
* }
* complex<float>\* DFT\_Project::low\_pass(complex<float> arr[],int num){
* complex<float> \*arr\_filtered = new complex<float> [num];
* std::complex<float> factor;
* for(int k=0;k<num;k++){
* factor.real(pow(0.98,k));
* factor.imag(pow(0.98,k));
* arr\_filtered[k] = arr[k]\*factor;
* }
* return arr\_filtered;
* }
* int\* DFT\_Project::Float\_to\_Int(complex<float> arr[],int num){
* int\* int\_arr = new int [num];
* for(int i=0;i<num;i++){
* int\_arr[i]=arr[i].real();
* }
* return int\_arr;
* }
* void DFT\_Project::SetVolumn(float f){
* VolumnFactor=f;
* }
* //float\* DFT\_Project::Change(float arr[],int num){
* //  float \*arr2 = new float [num];
* //  arr2[0]=arr[0];
* //  arr2[1]=arr[1];
* //  for(int k=2;k<num;k++){
* //      arr2[k]=arr[k-1];
* //  }
* //  return arr2;
* //}
* void DFT\_Project::Block(int num){
* GetBody(num);
* //printf("Start DFT transform\n");
* out=DFT(body,num);
* //printf("Start low\_pass\n");
* out=low\_pass(out,num);// 開啟後極為低通濾波
* //  printf("change voice"); not work!!
* //  out=Change(out,num);
* //printf("Start invDFT transform\n");
* out=InverseDFT(out,num);
* out2=Float\_to\_Int(out,num);
* //printf("Writing back file\n");
* SaveFile(out2,num);
* }
* void DFT\_Project::Main(){
* GetHeader();
* SaveFile(header,HeaderLength);
* printf("start calculating\n");
* do{
* Block(N);
* //printf("%d\t",BodyLength);
* }while(BodyLength-N>0);
* printf("writing end\n");
* if(EndLength>0){
* GetEnd();
* SaveFile(end,EndLength);
* }
* printf("finish!\n");
* }
* int main(){
* // set input file
* string wav\_url = "Nello.wav";
* char const \*wav\_url\_arr = wav\_url.c\_str();
* // set output file
* string out\_name = "Nello3.wav";
* char name[out\_name.length()+1];
* strcpy(name, out\_name.c\_str());
* int n = 512;    // k have same value as n
* DFT\_Project fftpr(n);
* fftpr.openReadFile(wav\_url\_arr);
* fftpr.openWriteFile(name);
* fftpr.Main();
* return 0;
* }

上述程式碼為做完傅立葉轉換後再根據有無低通濾波而完成

波形圖code:

from scipy.io.wavfile import read

# 讀取 WAV 檔案

rate, data = read("Nello3.wav")

# 取樣頻率

print("Sample rate: {} Hz".format(rate))

print("Data type: {}".format(data.dtype))

import matplotlib.pyplot as plt

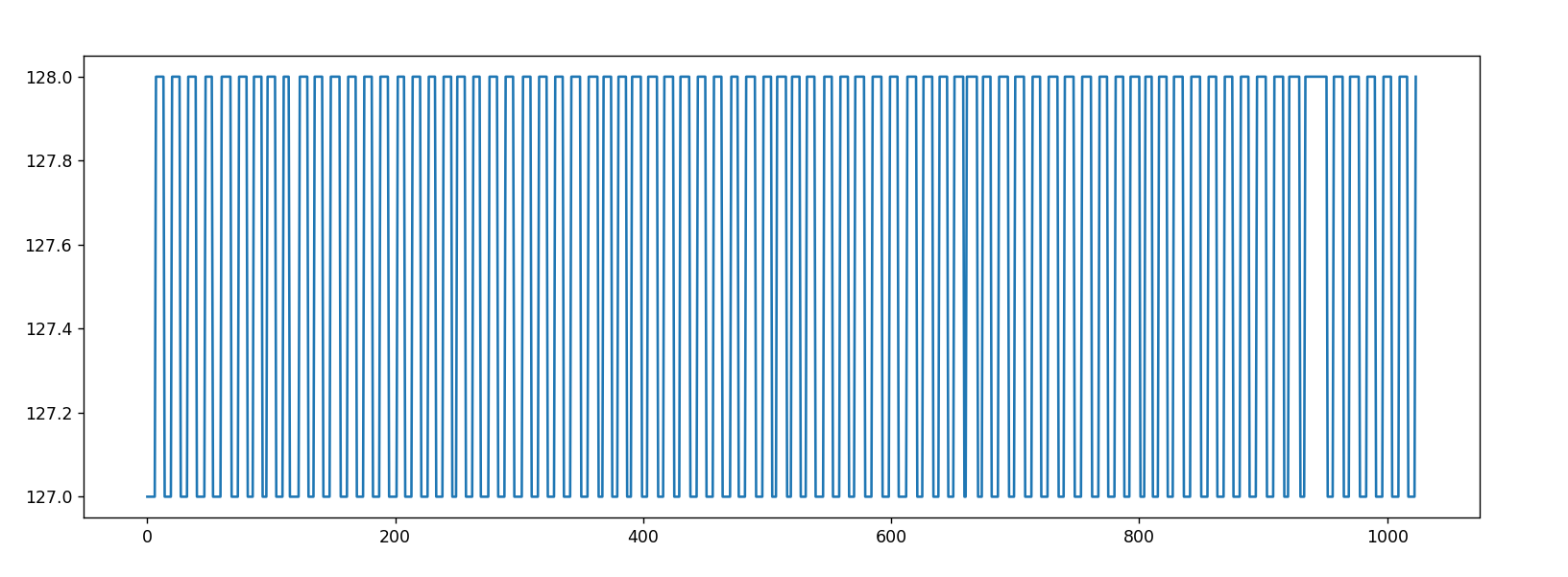
# 繪製前 1024 點資料的波形圖

plt.figure(figsize=(15, 5))

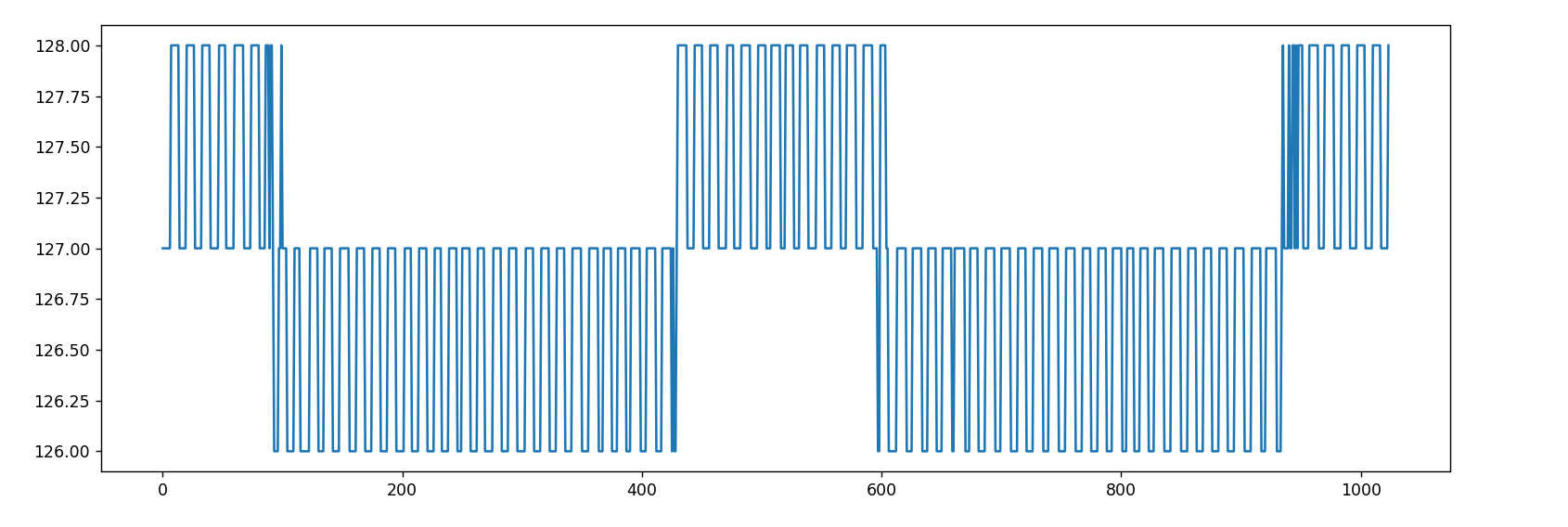
plt.plot(data[0:1024])

plt.show()

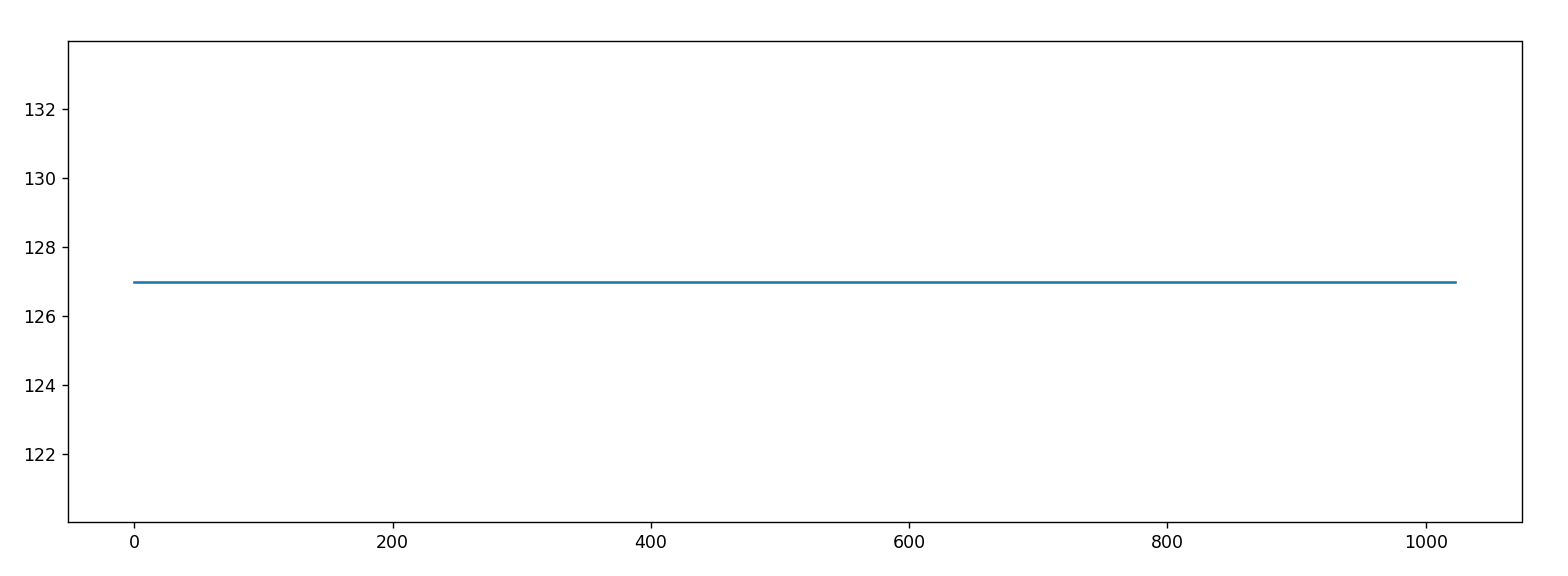
未轉換的波形圖



傅立葉轉換後的波形圖



開啟低通濾波且傅立葉轉換後的波形圖



**書面報告**(20%)

* + 作業題目、實作方法、工作進度、結果測試、參考資料、心得報告

這是我第一次使用python去完成作業並報告，選擇python的原因是聽聞同學分享用pthon做這題目是很好做的，但我本身沒有學過python也沒有甚麼python程式的coding經驗，於是我在周末時自己在家裡看著微軟教程按部就班的學習，雖然上面的內容都是基礎，但也同時讓我對python語法熟悉了許多，透過查找wave的函式庫及搜尋檔案讀取、輸出、編寫或讀取並完成了第一部分，以及上網尋找了抓出波形圖的程式，使用matlab函式庫讀取wave檔案畫出波形圖。

第二部分與第三部分就花費了相當多的時間，對於使用python完成我反而就不知道從何處下手才行，於是我花了些心思去尋找網路上所提供的解法，與同學的程式的程式教學，產生了傅立葉轉換及濾波器的程式碼。先將wave檔做一個fft轉換。第三部份我增加了濾波器的程式碼，直接在fft轉換上加上濾波，再反轉換回去，不得不說第二部分和第三部份都是高難度的部分，我們大家都在這上面花了不少時間去研究。過去聽說學長姊都是多人一組，但我們今年卻全部一人一組。雖然這是可以去讓每個人都實際的動手嘗試一遍，但對於個人的作業量還是挺有難度在的。不過透過這次作業更了解了wave檔的格式設定，也同時學會了python之奧妙，喜歡上了python這個語言。