R E P O R T

[Sound processing - Octave]

학 과	컴퓨터공학부 컴퓨터공학전공
교수님	서경룡 교수님
학 번	201911608
이 름	김지환
제출일	2022.06.19

부경대학교

목 차

● Term Project

Sound processing

FFT를 활용하여 음성 데이터를 처리한다.

□ 프로젝트 설명

· 음원(한 옥타브 도~시, 노래 한 소절, 배경음악 등)의 주파수를 분석
· 두 신호를 합성한 후 주파수 분석
· Filter를 사용하여 분리 및 주파수 분석
· 주제 선정 후 FFT를 이용한 프로젝트 수행

□ 구성

· Octave를 이용한 Project 수행
· fft를 활용
· 음성 분석
· 주파수 분석
· 음성 합성
· 음성 분리

□ 준비

· Octave 설치
· 피아노 음계 C -> B 음원
· 노래 한 소절 음성 파일
· random 주제 선정

□ 프로젝트 순서

· Project - 1
· Project - 2
· Project - 3

Project - 1

---

1. 한 옥타브의 음원을 준비한다.
   도, 레, 미, 파, 솔, 라, 시
   1-2초 , Sr은 음원에 따라 선정
2. 각 음을 주파수 분석한다.
3. 2개의 음을 합성한 다음 주파수 분석 한다
4. Filter를 사용하여 합성한 음을 분리한 다음 주파수 분석한다.
   - filter를 바꾸면서 필터에 따른 주파수 분석결과를 보인다.
5. 기타 작업을 수행한다.

---

1. C4 ~ B4 (2옥타브 도~시) 음원을 준비 (경로 : 옥타브 내 같은 폴더)

도.wav	2022-06-17 오후 4:04	WAV 파일	208KB
라.wav	2022-06-17 오후 4:05	WAV 파일	239KB
레.wav	2022-06-17 오후 4:05	WAV 파일	217KB
미.wav	2022-06-17 오후 4:05	WAV 파일	230KB
솔.wav	2022-06-17 오후 4:05	WAV 파일	244KB
시.wav	2022-06-17 오후 4:04	WAV 파일	185KB
파.wav	2022-06-17 오후 4:04	WAV 파일	239KB

https://thomson.tistory.com/584 (음계 mp3 file 다운로드)

https://convertio.co/kr/mp3-wav/ (mp3 -> wav 파일 변환)

2. 각 음의 주파수 분석

notes	notes	0	1	2	3	4	5	6	7	8
도	C	16.35	32.70	65.41	130.81	261.63	523.25	1046.50	2093.00	4186.01
도#	C#	17.32	34.65	69.30	138.59	277.18	554.37	1108.73	2217.46	4434.92
레	D	18.35	36.71	73.42	146.83	293.66	587.33	1174.66	2349.32	4698.64
레#	D#	19.45	38.89	77.78	155.56	311.13	622.25	1244.51	2489.02	4978.03
미	E	20.60	41.20	82.41	164.81	329.63	659.26	1318.51	2637.02	5274.04
파	F	21.83	43.65	87.31	174.61	349.23	698.46	1396.91	2793.83	5587.65
파#	F#	23.12	46.25	92.50	185.00	369.99	739.99	1479.98	2959.96	5919.91
솔	G	24.50	49.00	98.00	196.00	392.00	783.99	1567.98	3135.96	6271.93
솔#	G#	25.96	51.91	103.83	207.65	415.30	830.61	1661.22	3322.44	6644.88
라	A	27.50	55.00	110.00	220.00	440.00	880.00	1760.00	3520.00	7040.00
라#	A#	29.14	58.27	116.54	233.08	466.16	932.33	1864.66	3729.31	7458.62
시	B	30.87	61.74	123.47	246.94	493.88	987.77	1975.53	3951.07	7902.13

https://cosmosproject.tistory.com/255 (표 참고)

-> 현재 준비한 음원의 옥타브는 C4~B4이다.

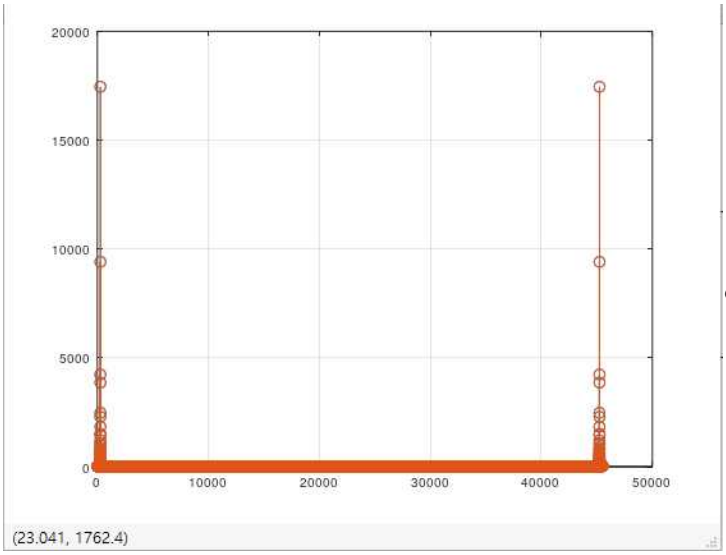
도 - 261.6Hz, 레 - 293.7Hz, 미 - 329.6Hz, 파 - 349.23Hz,

솔 - 392Hz, 라 - 440Hz, 시 - 493.9Hz 이다.

2-1 Octave를 이용해 주파수 분석

- C4 (2옥타브 - 도, 261.6Hz)

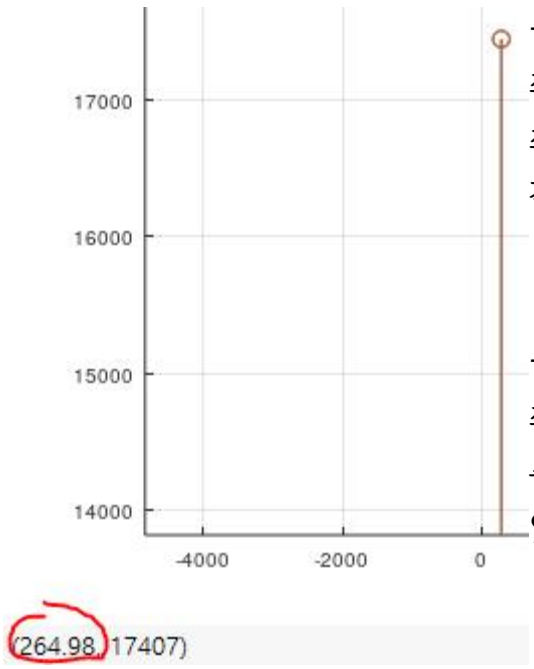
```
c = "C.wav"; % 경로 설정 (현재는 같은 경로)
[scale_C, Cr]=audioread(c);
len_c = length(scale_C); % data양
fc = fft(scale_C,len_c/2); % 채널이 두개이므로 나누기 2
figure(1); stem(abs(fc)); grid on % fft로 인한 허수 부분을 없애기 위해 abs
```



->

좌, 우로 듣기 위해 양쪽의 값이 같음.

```
>> abs(max(fc))
ans =

   1.7451e+04   1.7451e+04
```



-> 약 17,451로 위 Stem 그래프의 최댓값과 유사함을 알 수 있음.
최댓값이 2개로 좌,우로 듣기 위해 값이 2개임을 증명

->

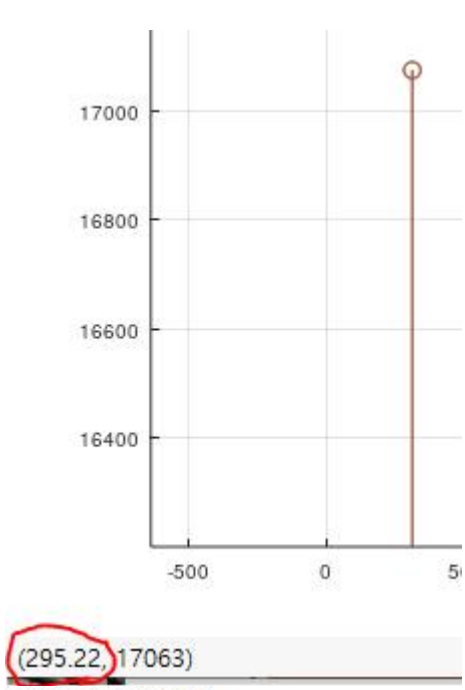
최댓값의 x축이 약 264.98로 C4의 주파수인 261.6Hz와 근사함을 알 수 있음. 마우스 위치에 따른 문제.

마찬가지로 레, 미, 파, 솔, 라, 시까지 확인한다.

- Source

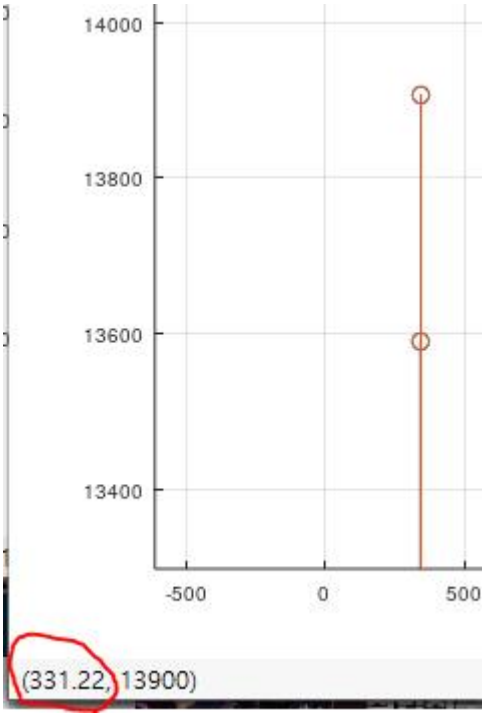
```
%clear
clear all; close all; clc;

%% scale 음원
c = "C.wav"; d = "D.wav"; e = "E.wav"; f = "F.wav";
g = "G.wav"; a = "A.wav"; b = "B.wav";

% data, Sampling Rate
[scale_C, Cr]=audioread(c); [scale_D, Dr]=audioread(d); [scale_E, Er]=audioread(e);
[scale_F, Fr]=audioread(f); [scale_G, Gr]=audioread(g); [scale_A, Ar]=audioread(a);
[scale_B, Br]=audioread(b);

len_c = length(scale_C); % data양
fc = fft(scale_C,len_c/2); %나이퀴스트 정리로 기존 대역폭을 찾으려면 /2
figure(1); stem(abs(fc)); grid on %fft로 인한 허수 부분을 없애기 위해 abs

len_d = length(scale_D);
fd = fft(scale_D,len_d/2);
figure(2); stem(abs(fd)); grid on

len_e = length(scale_E);
fe = fft(scale_E,len_e/2);
figure(3); stem(abs(fe)); grid on

len_f = length(scale_F);
ff = fft(scale_F,len_f/2);
figure(4); stem(abs(ff)); grid on

len_g = length(scale_G);
fg = fft(scale_G,len_g/2);
figure(5); stem(abs(fg)); grid on

len_a = length(scale_A);
fa = fft(scale_A,len_a/2);
figure(6); stem(abs(fa)); grid on

len_b = length(scale_B);
fb = fft(scale_B,len_b/2);
figure(7); stem(abs(fb)); grid on
```

-> 남은 음계들의 그래프 및 주파수 분석하기



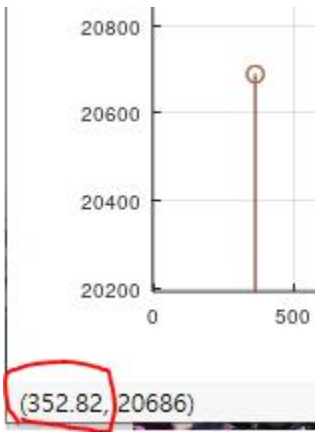

```
>> max(abs(fd))
ans =

   1.7073e+04   1.7073e+04
```

□ 293.7Hz, 약 1.5차이
(Max값 근사)-D4



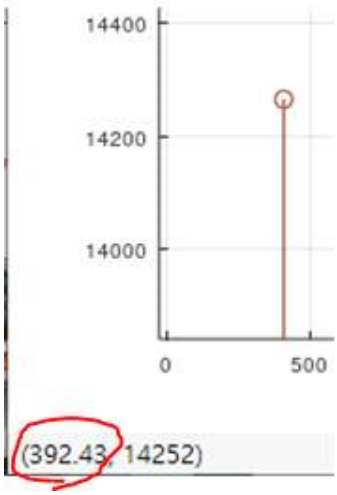

```
>> max(abs(fe))
ans =

   1.3907e+04   1.3907e+04
```

□ 329.6Hz, 약 1.6차이
(Max값 근사)-E4



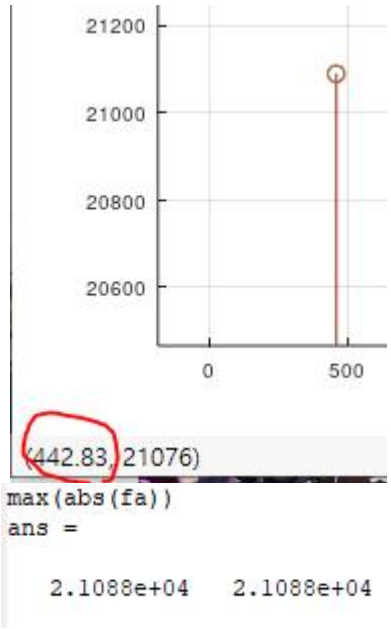

```
>> max(abs(ff))
ans =

   2.0688e+04   2.0688e+04
```

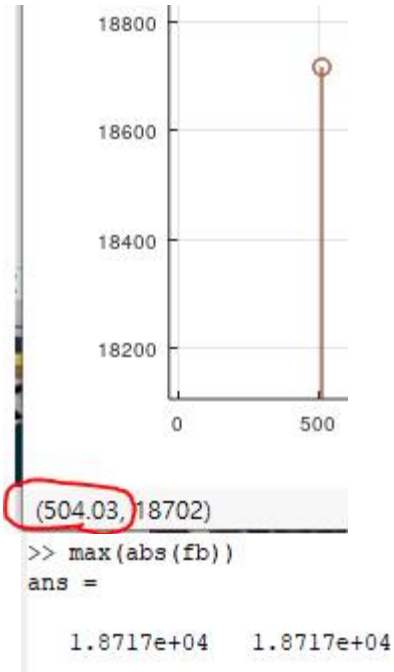
□ 349.23Hz, 약 3.6차이
(Max값 근사)-F4




```
>> max(abs(fg))
ans =

   1.4265e+04   1.4265e+04
```

□ 392Hz, 일치
(Max값 근사)-G4





▫ 440Hz, 약 2.8차이
(Max값 근사)-A4

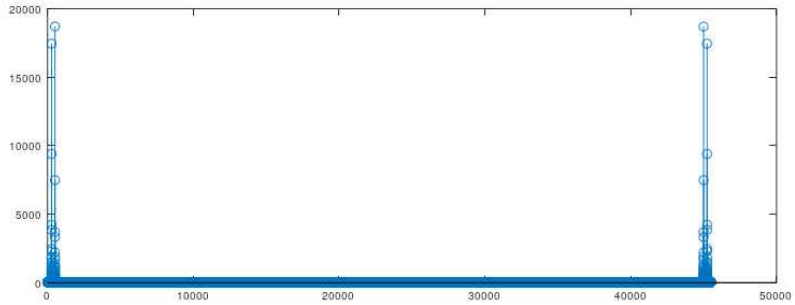
▫ 493Hz, 약 10차이
(Max값 근사)-B4

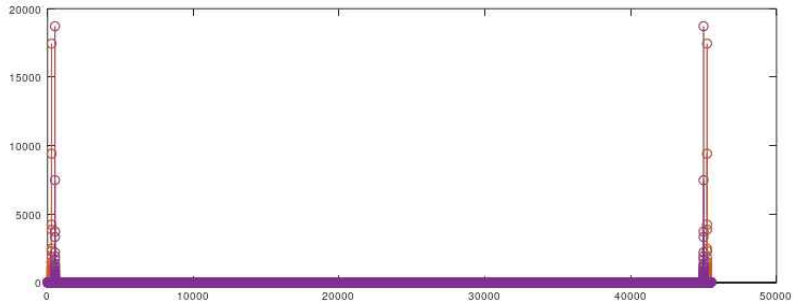
분석 결과 : 피아노 건반 등 스케일에는 여음이 존재한다. 그리고 마우스의 위치가 고정적이기 힘들며 작은 단위의 정확한 포커스를 맞추기 힘들다. 그래서 차이 값이 생긴 것 같다. 튜너라는 어플리케이션을 이용해 여음이 존재함을 알 수 있다.
피아노 건반, 기타, 목소리 등 일정 음을 내면 주파수가 한 지점에 머물지 못하고 왔다갔다 하고 음이 끝날 쯤에는 한옥타브 차이의 음이나 화음(도일 경우 미) 같은 스케일이 주파수에 잡힌다.
이 정도 근사값이면 정상적이게 제대로 분석이 된 것 같다.

3. 2개의 음을 합성 후 주파수 분석

```
%clear
clear all; close all; clc;

%% scale 음원
c = "C.wav"; b = "B.wav";

[scale_C, Cr]=audioread(c); [scale_B, Br]=audioread(b);

%%2번까지 같은 내용
len_c = length(scale_C);
fc = fft(scale_C,len_c/2);
len_b = length(scale_B);
fb = fft(scale_B,len_b/2);

%주파수 기준 데이터가 적은 것을 가져옴
len = min(length(fb),length(fc));
fn = zeros(len,1);
for i = 1:len
   fn(i)=fb(i)+fc(i);
end

%도, 시 음 합성 튜너를 이용해 확인완료
%audiowrite('melody.wav',ifft(fn,len*2),Cr);
%[scale_N,Nr] = audioread("melody.wav");
%sound(scale_N,Nr);

subplot(211); stem(abs(fn));
%% fn을 fc와 fb의 합성과 비교
subplot(212); stem(abs(fc)); hold on; stem(abs(fb));
```

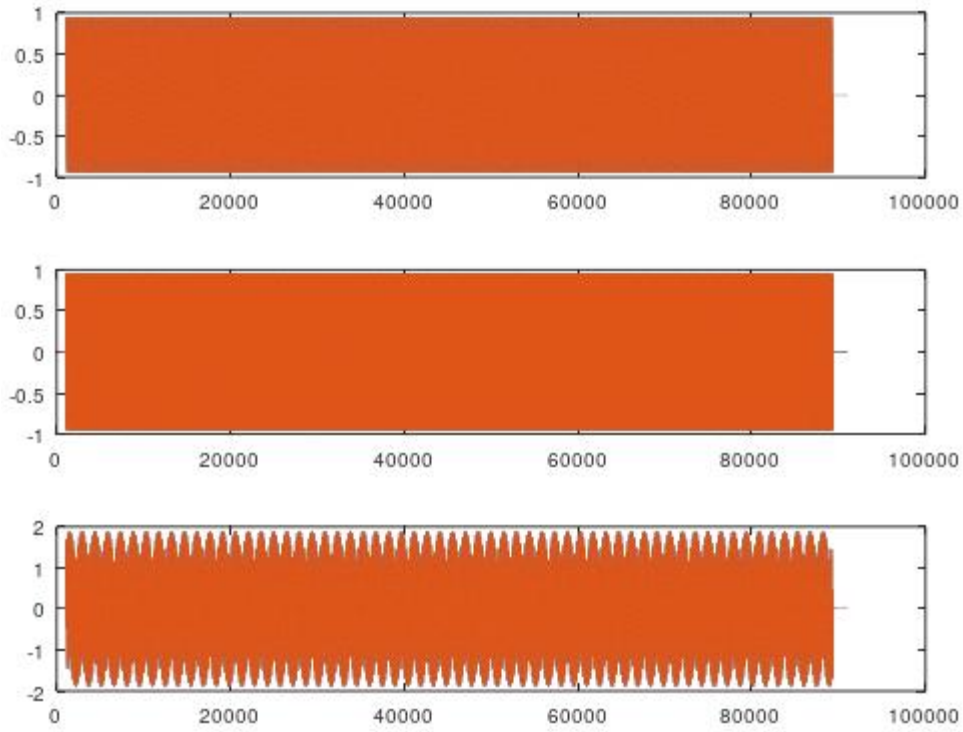
-> 비교하기 쉬운 C4와 B4의 주파수를 기준으로 합성한 후 C4의 주파수와 B4의 주파수를 겹쳐 놓은 것과 비교해보았다.
합동이다.


20000
15000
10000
5000
0
0
10000
20000
30000
40000
50000

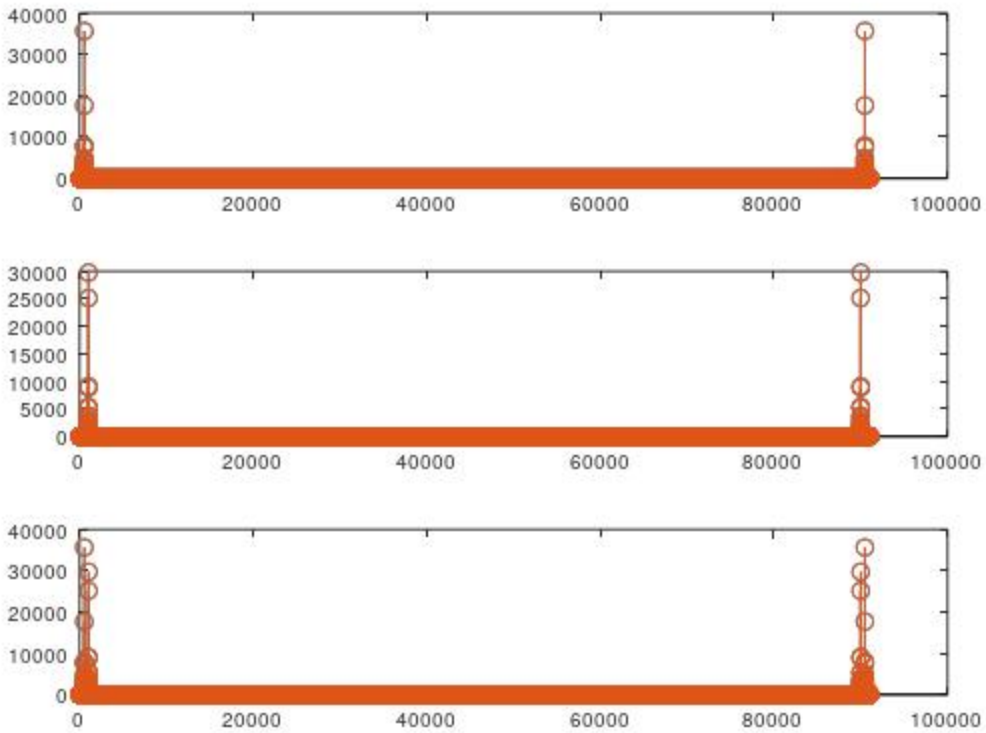

20000
15000
10000
5000
0
0
10000
20000
30000
40000
50000

4. Filter를 사용해 합성된 음을 분리 후 주파수 분석

➜ Filter 전 C4, B4, C4+B4 의 시간 축

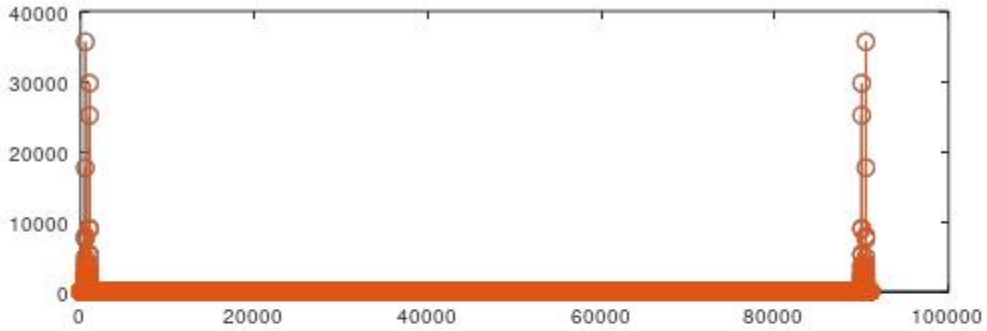


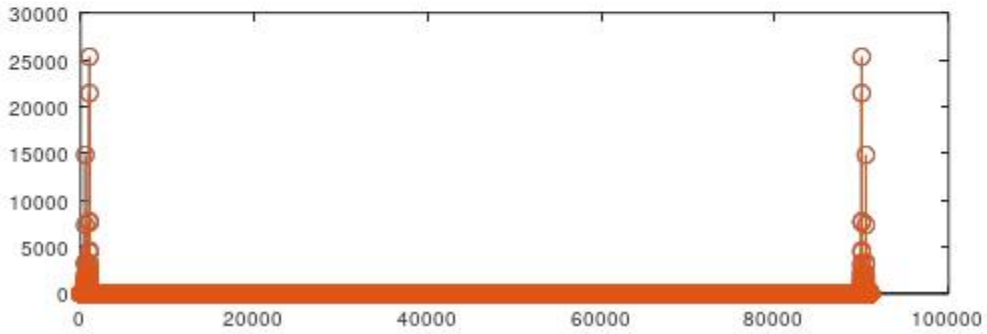
➜ Filter 전 C4, B4, C4+B4 의 주파수 축



```
clear; pkg load signal %butter 사용
C = audioread('C.wav');
B = audioread('B.wav');

length = min(size(C),size(B));
melody_r1 = C(1:length, :); %길이에 맞추어 데이터 나눔
melody_r2 = B(1:length, :);
newMeody = (melody_r1+melody_r2); %합성

audiowrite("newMelody1.wav",newMeody,44100); % 합성한 멜로디 음원생성
[newScale1,Fs1] = audioread("newMelody1.wav");
sound(newScale1,Fs1); %소리 테스트

Fnyq = length(1)/2; % PPT참고, 나이퀴스트 정리에 의해 Sr/2
Fc=542; % 주파수 설정
[b,a]=butter(2,Fc/Fnyq); % butter를 통해 필터 조건 만족, Low
clear("Fnyq","Fc");

[F,FL] = FFT(newMeody(:,1),44100);
[val, idx] = max(abs(FL));
printf("%d, %f \n", idx, val);

t=0:(1.5)/length(1):(1.5)-(1.5/length(1));

%butter에서 나온 값으로 필터링, 로우 필터
output = filter(b,a,newMeody);

figure(1)
subplot(211); plot(t, newMeody);
subplot(212); plot(t, output);

figure(2)
subplot(211); stem(abs(fft(newMeody)));
subplot(212); stem(abs(fft(output)));

audiowrite("newMelody.wav",output,44100);
[newScale,Fs] = audioread("newMelody.wav");
sound(newScale,Fs);

[F,FL] = FFT(output(:,1), 44100);
[val, idx] = max(abs(FL));
printf("%d, %f \n", idx, val);
```

-> Fc를 542Hz (C4의 나이퀴스트 정리)로 설정 후 로우패스 Filter 실행
∴ 노이즈 제거 및 (도,시)화음에서 깨끗한 도만 들림.

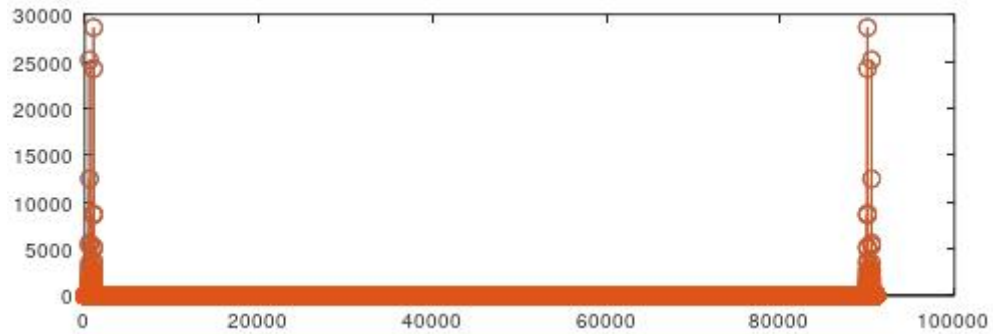




- 필터에 따른 주파수 분석결과를 보인다.

Filter를 high로 실행 해보았다.

결과) 노이즈가 남아있는 C4가 들리며 그래프는 변경되었다.



Low 필터

```
>> new_piano
542, 0.782922
542, 0.554631
```

High 필터

```
>> new_piano
542, 0.782922
1020, 0.628529
```

5. 기타 작업을 수행한다.

➜ Filter가 진행된 Data를 음원으로 바꿔서 들어보았다. 소리의 변화가 확실히 생겼다. 그리고 주파수의 변화를 코드를 통해 직접 확인하였다. 변화된 주파수를 ifft를 이용해 복구 시켜보았지만 큰 변화가 없었다.

Project - 2

Pjt2

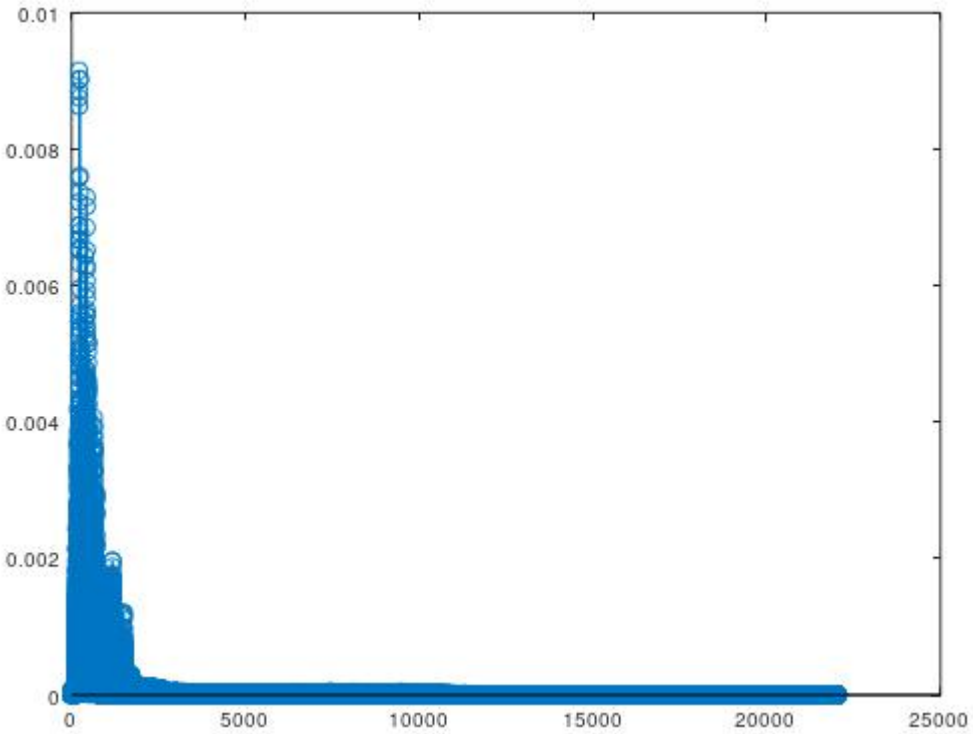
1. 휴대폰을 사용하여 노래 한 소절을 녹음한다.
2. 녹음한 음성 데이터를 주파수 분석 한다.
3. 배경음악을 준비하여 두 신호를 합성한 후 주파수 분석을 한다.
4. 배경음악을 필터를 사용하여 분리할 수 있는지 실험한다.

---

1. 휴대폰을 사용하여 노래 한 소절을 녹음한다.

녹음.m4a -> sing.wav

2. 녹음한 음성 데이터를 주파수 분석한다.



녹음한 음성 데이터를 주파수 차트로 변경하였다.

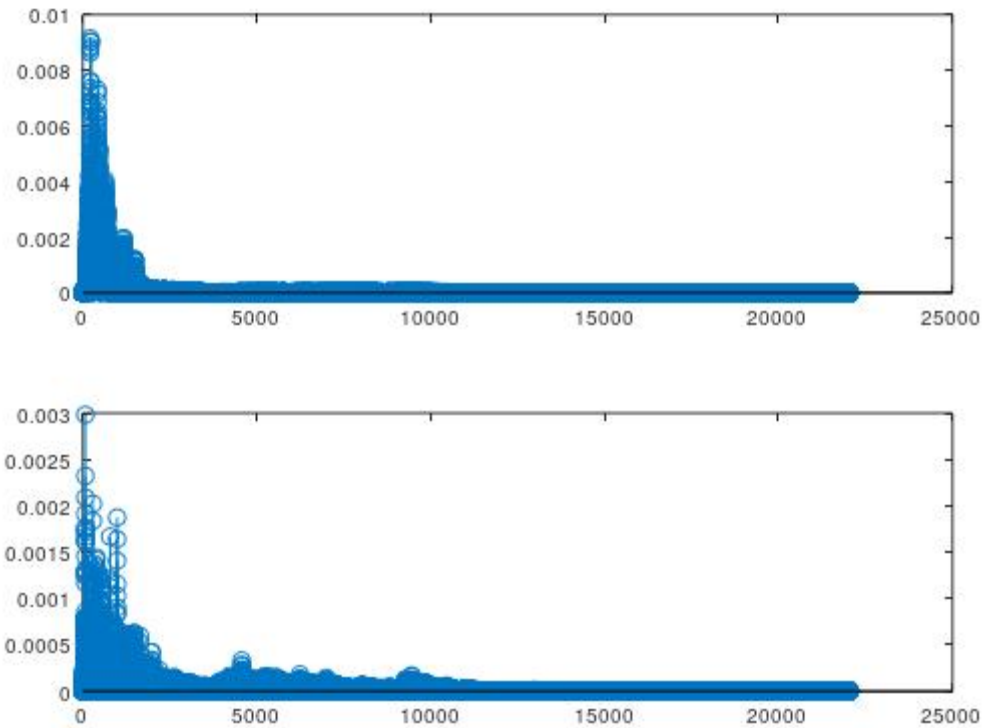
```
>> song
max : 2683 Hz => 0.009150
```

그래프만 보아도 최고음은 200Hz 정도 된다. 그리고 낮은 노래를 불렀는데 최고 주파수가 2683Hz가 나왔는데 이유를 모르겠다.

녹음한 음성 데이터의 주파수 분석 코드

```
clear;

[d, f] = audioread('sing.wav');
[d1, f1] = audioread('back.wav');
player = audioplayer(d,f)
player = audioplayer(d1,f1)
%play(player);

[F, FL] = FFT(d(:,1),f);
[val, idx] = max(abs(FL));
printf("max : %d Hz => %f\n", idx, val);
subplot(211); stem(F,abs(FL));

[F, FL] = FFT(d1(:,1),f1);
[val, idx] = max(abs(FL));
printf("max : %d Hz => %f\n", idx, val);
subplot(212); stem(F,abs(FL));
```

3. 배경음악과 노래 두 신호를 합성 후 주파수 분석



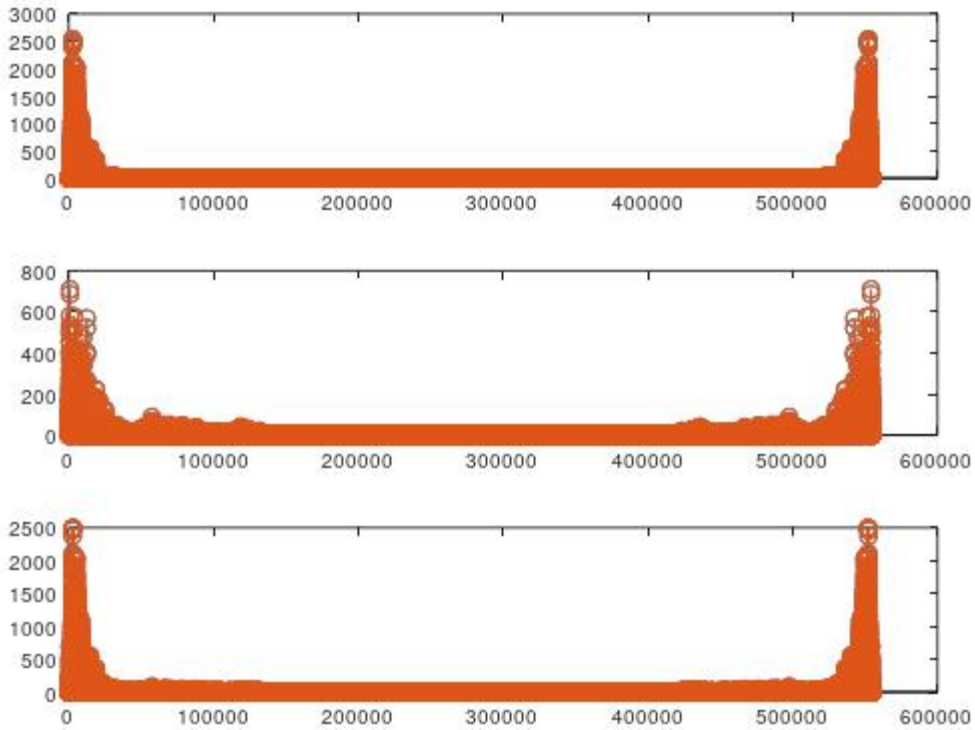
두 신호의 합성 전 주파수 이다. (위-녹음, 아래-인트로)

```
max : 2683 Hz => 0.009150
max : 995 Hz => 0.002981
```

이것도 최고 주파수가 이상하게 나온 것 같다.

```
TotalSamples = 555008

TotalSamples = 595968
```

이번엔 두 샘플의 길이가 다르다. 샘플의 길이부터 맞추며 시작



-> 합성 결과: 직접 부른 곡과 비슷해졌고 배경을 포함한 음악이 완성

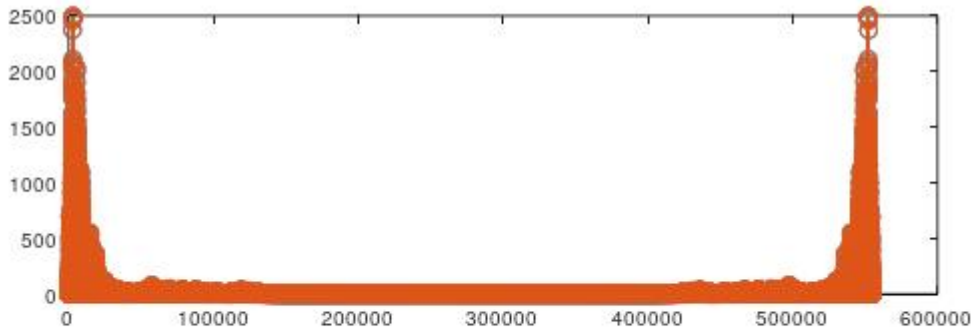
두 신호를 합성 후 주파수 분석한 코드

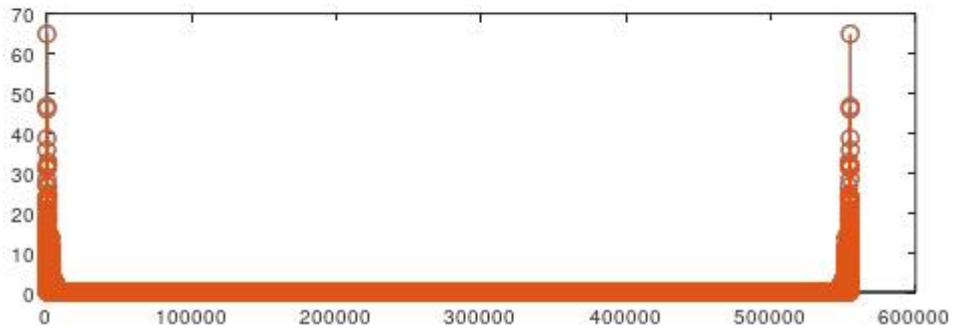
```
clear;

sing = audioread('sing.wav');
mr = audioread('back.wav');

length = min(size(sing),size(mr));
sing_r = sing(1:length,:);
mr_r = mr(1:length,:);
song = sing_r.+mr_r;

subplot(311); stem(abs(fft(sing_r)));
subplot(312); stem(abs(fft(mr_r)));
subplot(313); stem(abs(fft(song)));

audiowrite("song.wav",song,44100);
[newSong,Fs] = audioread("song.wav");
sound(newSong,Fs);
```

4. 배경음악을 필터를 사용해 분리

코드

```
clear; pkg load signal

sing = audioread('sing.wav');
mr = audioread('back.wav');

length = min(size(sing),size(mr));
sing_r = sing(1:length,:);
mr_r = mr(1:length,:);
song = sing_r.+mr_r;

Fnyq = length(1)/2;
Fc = 200; %최고음이 200Hz 내외
[b,a] = butter(2, Fc/Fnyq);

subplot(211); stem(abs(fft(song)));
output = filter(b,a,song);
subplot(212); stem(abs(fft(output)));

audiowrite("filterSong.wav",output,44100);
[newSong,Fs] = audioread("filterSong.wav");
sound(newSong,Fs);
```





-> Fc를 200으로 주며 최고음역대 부분만 걷어낼 줄 알았는데 마음처럼 되지 않았다. 그래서 Fc를 늘려봤는데 3500까지 올려야 배경음이 조금 자리가 잡힌다. 정확한 이유를 모르겠어서 더 공부해야겠다.

Project - 3

---

fft 를 활용하여 관심 있는 주제를 선정하여 수행한다.

->

동물 중 호랑이, 악어 중 Alligator는 저주파로 유명하다.

호랑이의 경우 저주파로 먹잇감을 정지 시킨다는 말도 있고 악어는 저주파 관련 상품도 굉장히 많다.

그리고 돌고래와 박쥐는 고주파로 유명하다.

돌고래와 박쥐가 높게 내는 음역대는 사람의 가청음역을 벗어나고, 도대체 얼마나 되는지 그것이 궁금해서 이번에 배운 fft로 직접 실험해보게 되었다.

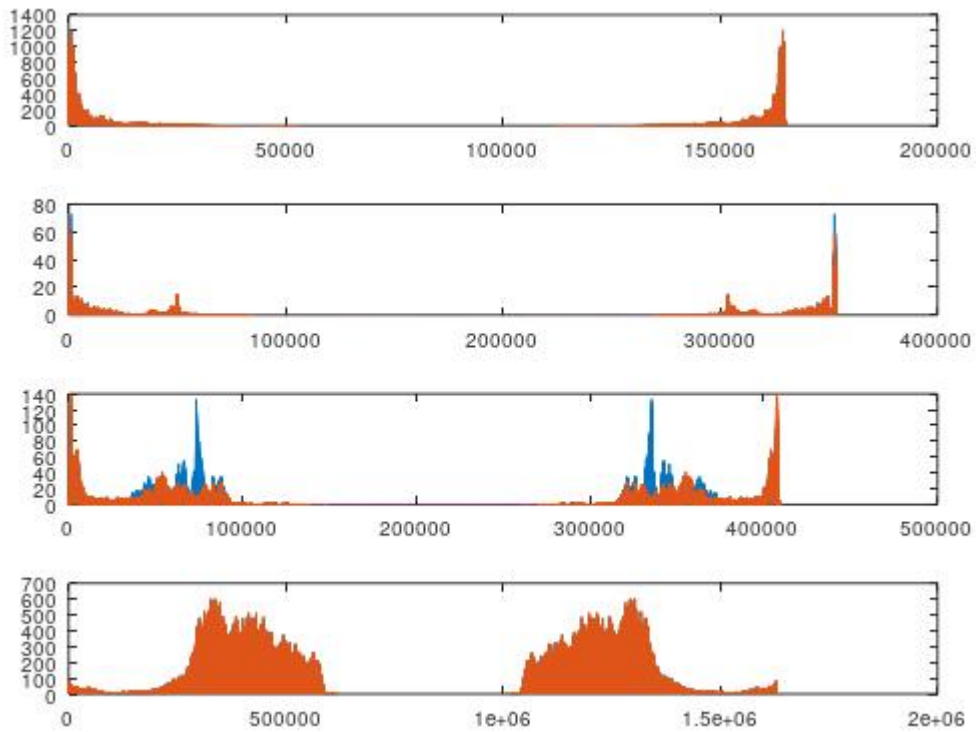
먼저 최고 주파수를 찾아보았다.

```
>> animal
Tiger, max : 532 Hz => 0.014569
Alligator, max : 1081 Hz => 0.000413
Dolphin, max : 73401 Hz => 0.000652
Bat, max : 326754 Hz => 0.000731
```

뭔가 결과 값이 이상하다.

호랑이의 경우 10~400으로 검색되고 돌고래와 박쥐의 경우는 옳은 것 같다.

들린 부분은 낮게 내고 안들린 부분도 분명 있을 것이다.



해당 주파수들을 그래프로 변환 해보았다. 박쥐와 돌고래는 고음역대가 기본이며 호랑이와 악어는 저 음역대가 기본임이 보인다.

저음역대도 알아보고 싶지만 아직 공부가 부족해서 방법을 잘 모르겠다.

프로젝트를 마무리하며

● 문제점 발견
➜ 시험기간이 겹쳐 있고 점수는 잘나오고 싶은 마음에 프로젝트를 제대로 진행하지 못하였다.
➜ 강의 영상을 보며 응용을 하는 과목인데 이해도가 부족했다.
➜ 위 두 내용을 합쳐 프로젝트 전 ZOOM 설명에서 독창성을 본다는 말에 너무 혼자 진행했다. 그래서 Filter에서 막혔었다.

● 문제점 해결
➜ 다른 내용들은 어느정도 이해를 했는데 FILTER에 대한 이해도가 부족해서 강의 영상을 무한 반복 시청했다. 그 결과 근접하지도 않던 답이 도출 되었다. 정답인지는 모르겠지만 FILTER를 공부하다가 FILTER의 기능을 알게되고 문제를 해결 할 수 있었던 것 같다.

● 프로젝트를 진행하며 배운 점
➜ 노이즈 제거 등 소리의 안정성과 원하는 값만 도출 할 수 있다는 것을 깨닫게 되었다.
➜ 음성이 보이지도 않고 진동도 안느껴지는데 어떻게 미디어로 만드는지 궁금했다. 궁금증이 해결되었다.

● 독창성에 대해
➜ 저는 교수님께서 공유하신 자료의 주어진 틀에 의존하지 않고 1번 문제를 스스로 방법을 찾아 해결하고 있었습니다.
➜ 고주파, 저주파 동물에 대한 조사는 좋은 의견이라고 생각합니다.

● 마무리
➜ 결과적으로 교수님의 자료에서 이외에 FILTER에 대한 답을 도출하기 힘들었습니다. 이번 프로젝트, 수업을 통해 Octave와 어려운 난이도의 수학 쓰임새를 제대로 배우게 되었습니다.