Assignment #2

REPORT

교과명	Digital Speech Processing
교수명	신종원 교수님
제출일	2023.04.09
제출자	AI 대학원 20231050 채종욱

1. 전체 소스코드

```
import numpy as np
from scipy.io import wavfile
import librosa
import librosa.display
import matplotlib.pyplot as plt

# load the speech signal
fs, speech = wavfile.read('speech1.wav')

print('fs, speech', fs, len(speech))

# LP filter
h = np.ones(11) / 11

# parameter setting.
hop = 80 # Frame shift
window_length = 128 #window length


# get frames
frames = get_frames(speech, hop, window_length)

# window design
window = np.concatenate([np.hanning(48)[:24], np.ones(80), np.hanning(48)[24:]])

# windowing
windowed_frames = frames * window


# Speech FFT
fft_frames = np.fft.fft(windowed_frames, axis=1)

# LPF FFT
fft_filter = np.fft.fft(h, window_length)

# Multiply Speech and LPF
filtered_frames = np.real(np.fft.ifft(fft_frames * fft_filter, axis=1))

# reconstruct
reconstructed_signal = np.zeros(len(speech) + 48) # 원래 시그널의 길이보다 48개의 샘플이 더 발생함.

for i in range(len(frames)):
    reconstructed_signal[i*hop:i*hop+window_length] += filtered_frames[i, :]



# 파일 저장.
wavfile.write('reconstructed_signal_without_lpf.wav', fs, reconstructed_signal.astype(np.int16))
```

2. get_frames

```
def get_frames(speech, hop, window_length):

    frames = np.zeros((len(speech) // hop, window_length))

    for i in range(len(frames)):
        # 0~23까지 이전 데이터, 23~ 103 까지는 현재 데이터, 104~123까지는 zero-padding.
        frames[i, :24] = frames[i-1, -48:-24] # * np.hanning(48) # 처음 샘플 24개는 이전 프레임의 24개로부터 얻어짐.
        frames[i, 24:104] = speech[i*hop:i*hop+80] # 80개 샘플은 rectangular window
        frames[i, 104:] = 0

    return frames
```

Frame의 마지막 24개는 zero-padding이기 때문에, Frame의 처음 24개는 이전 frame의 80~104까지의 데이터를 가져오도록 하였다.

그리고 첫번째 frame일 경우, 앞의 24개의 샘플은 이전 샘플로부터 가져올 데이터가 없기 때문에 현재 샘플로 채워야할지 아니면 0으로 채워야할지 고민하였는데, 우선 0으로 채우는 방향으로 진행하였다.

3. LPF

The LPF is simply moving-average filter which is $h[n] = 1,\ 0 \leq n \leq 10\ (P = 11)$.

```
# LP filter
h = np.ones(11) / 11
```

4. Window

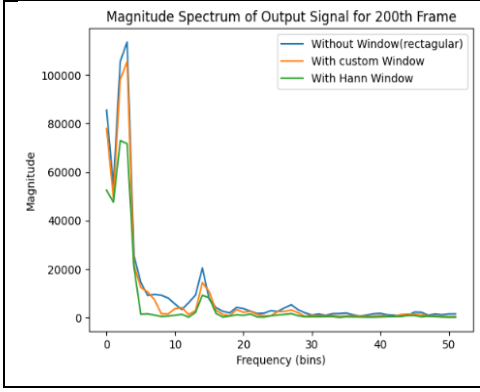
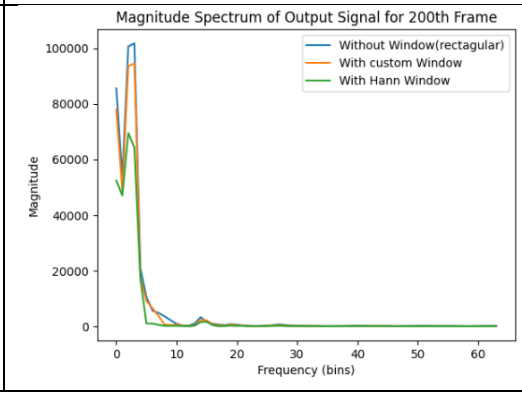
```
# window design
window = np.concatenate([np.hanning(48)[:24], np.ones(80), np.hanning(48)[24:]])
```

설계 조건에 맞게, 앞의 24개와 뒤의 24개는 hanning window, 그리고 현재 샘플인 80개는 rectangular 윈도우로 설계하였다.

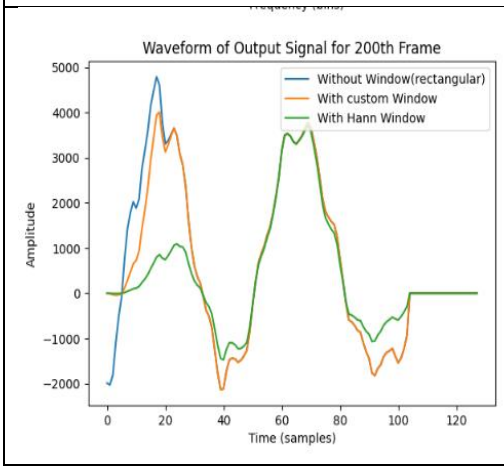
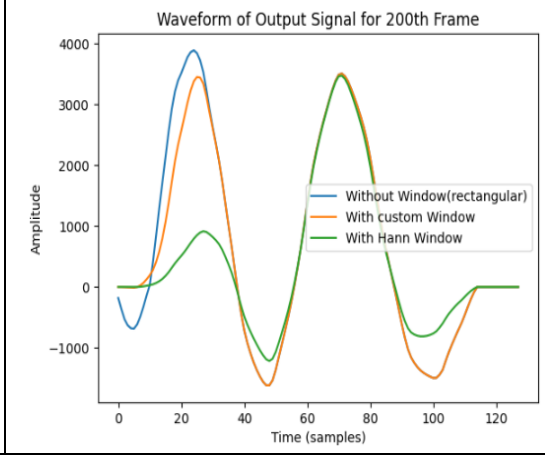
5. Reconstructed signal

IFFT한 신호에 Overlap-add를 해주었습니다. 이때, 복원된 신호의 길이가 원본 신호의 길이보다 48개 더 많은 것을 확인하였습니다. 첫번째 frame의 앞의 24개 샘플과 마지막 frame의 뒤의 24개 샘플로 인해서 원본신호보다 48개 더 많아진 것으로 생각이 됩니다. 우선 48개의 샘플도 값을 가지고 있어서 전부 복원하였습니다. 이러한 경우 앞의 24개와 뒤의 24개의 샘플을 버리고 복원을 해서 원본 신호의 길이와 맞춰야 하는지 아니면 크게 상관이 없는지 궁금합니다.

6. Spectrum 비교

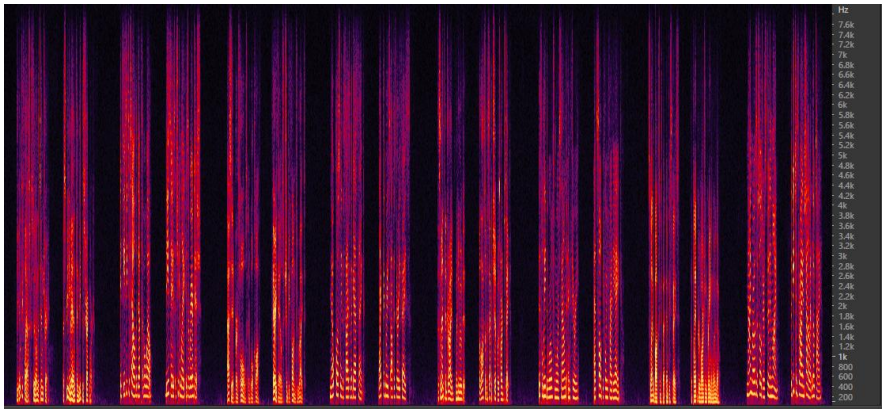
LPF 적용 X	LPF 적용 O
	

Rectangular, custom, hanning window 순으로 magnitude가 높은 것을 확인할 수 있다. 이는 Rectangular window의 frequency resolution이 가장 좋기 때문으로 생각된다. 반면, hanning winodw는 magnitude가 낮은 대신에, sidelobe의 간섭이 가장 덜한 것을 확인할 수 있다. 그리고 마지막으로 custom window는 현재 샘플에 대해서는 rectangular, 과거 데이터에 대해서는 hanning window를 사용했기 때문에, 두 window의 장점이 적절하게 반영되어 있는 것을 확인할 수 있다.

LPF 적용 X	LPF 적용 O
	

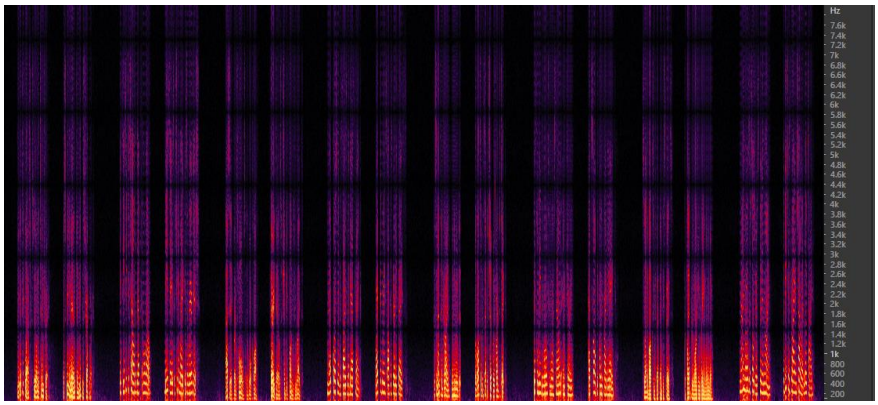
위의 그래프는 window를 적용한 waveform이다. Hanning window와 양끝을 hanning으로 설계한 custom window는 양 끝쪽이 0값에 가까워지는 weighted window인 것을 볼 수 있다. 그리고 전체적으로 LPF를 적용한 그래프는 전체적으로 완만한 형태의 그래프인 것을 확인할 수 있다.

Original



Reconstruced(LPF O)

MA필터로 인해서 특정 주파수 부분이 사라진 것을 볼 수 있다.



Reconstructed(LPF X)