# Pattern Recognition Assignment 2: Speech Recognition

Hailey Yu Haihong
Levi Leyh

May 10, 2023

# Contents


**1 Plot of female and male speech and the music signal** **1**

**2 Plot of the spectrograms** **3**

**3 Comparison of the spectrogram and the (normalized) cepstrogram** **4**

**4 4. Correlation of the spectral and cepstral coefficient series** **6**

**5 Questions in the text** **7**

**6 Further thoughts** **7**

# 1 Plot of female and male speech and the music signal

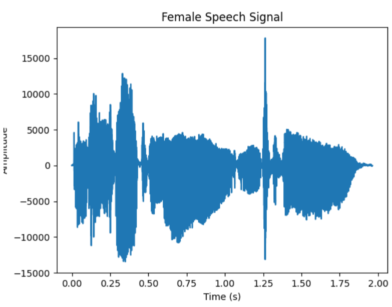
This is the used code:

Listing 1: Plot of female and male speech and the music signal

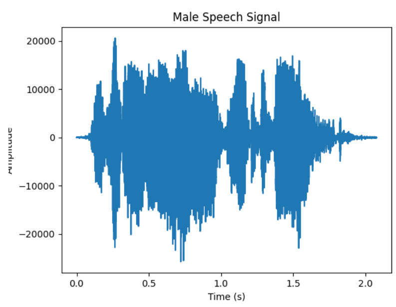
```python
"""
Authors: Hailey Yu Haihong, Levi Leyh

This code loads and visualizes three audio signals:
female speech, male speech, and music.
It then zooms in on a range of 20 ms in different regions of the signals
and plots the zoomed-in signals in separate subplots.
"""
import matplotlib.pyplot as plt
import numpy as np
from scipy.io import wavfile

# Load the female and male speech signal and the music signal
# Get the sampling frequency
female_fs, female_speech = wavfile.read('Sounds/female.wav')
male_fs, male_speech     = wavfile.read('Sounds/male.wav')
music_fs, music          = wavfile.read('Sounds/music.wav')

# Plot the female speech signal
t_female_speech = np.arange(len(female_speech))/female_fs  # Time axis in seconds
plt.figure()
plt.plot(t_female_speech, female_speech)
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('Female Speech Signal')

# Plot the male speech signal
t_male_speech = np.arange(len(male_speech))/male_fs  # Time axis in seconds
plt.figure()
plt.plot(t_male_speech, male_speech)
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('Male Speech Signal')

# Plot the music signal
t_music = np.arange(len(music))/music_fs  # Time axis in seconds
plt.figure()
plt.plot(t_music, music)
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('Music Signal')

# Zoom in on a range of 20 ms in different regions of the signals
start_time = [0.5, 1, 1.5]  # Start time of the zoomed-in range
duration   = 0.02  # Duration of the zoomed-in range in seconds


# Create a figure for female speach signal
fig_f, axes_f = plt.subplots(nrows=len(start_time), figsize=(8, 6))
# Adjust spacing between subplots
plt.subplots_adjust(hspace=0.5)

# Zoom in on the female speech signal
for i, start in enumerate(start_time):
    # Calculate indices and extract zoomed signal
    idx_start = int(start * female_fs)
    idx_end = idx_start + int(duration * female_fs)
    t_zoom = t_female_speech[idx_start:idx_end]
    female_speech_zoom = female_speech[idx_start:idx_end]

    # Plot zoomed signal in corresponding subplot
    ax = axes_f[i]
    ax.plot(t_zoom, female_speech_zoom)
    ax.set_xlabel('Time (s)')
    ax.set_ylabel('Amplitude')
    ax.set_title(f'Female Speech Signal (Zoomed in at {start:.2f} s)')


# Create figure for male speech signal
fig_m, axes_m = plt.subplots(nrows=len(start_time), figsize=(8, 6))
# Adjust spacing between subplots
plt.subplots_adjust(hspace=0.5)

# Zoom in on the male speech signal
for i, start in enumerate(start_time):
    # Calculate indices and extract zoomed signal
    idx_start = int(start * male_fs)
    idx_end = idx_start + int(duration * male_fs)
    t_zoom = t_male_speech[idx_start:idx_end]
    male_speech_zoom = male_speech[idx_start:idx_end]

    # Plot zoomed signal in corresponding subplot
    ax = axes_m[i]
    ax.plot(t_zoom, male_speech_zoom)
    ax.set_xlabel('Time (s)')
    ax.set_ylabel('Amplitude')
    ax.set_title(f'Male Speech Signal (Zoomed in at {start:.2f} s)')
```

Listing 2: Plot of female and male speech and the music signal

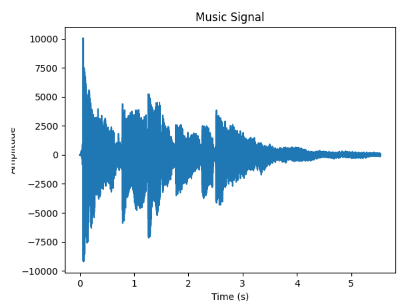
```python
# Create a figure with multiple subplots for the music signal
fig_mu, axes_mu = plt.subplots(nrows=len(start_time), figsize=(8, 6))
# Adjust spacing between subplots
plt.subplots_adjust(hspace=0.5)

# Zoom in on the music signal
for i, start in enumerate(start_time):
    # Calculate indices and extract zoomed signal
    idx_start = int(start * music_fs)
    idx_end = idx_start + int(duration * music_fs)
    t_zoom = t_music[idx_start:idx_end]
    music_zoom = music[idx_start:idx_end]

    # Plot zoomed signal in corresponding subplot
    ax = axes_mu[i]
    ax.plot(t_zoom, music_zoom)
    ax.set_xlabel('Time (s)')
    ax.set_ylabel('Amplitude')
    ax.set_title(f'Music Signal (Zoomed in at {start:.2f} s)')

plt.show()
```




(a) Female speech

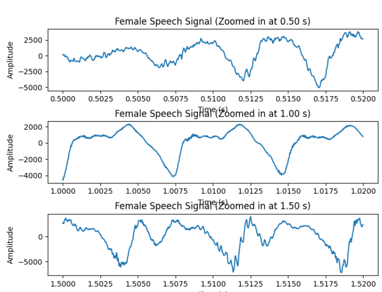



(b) Male speech

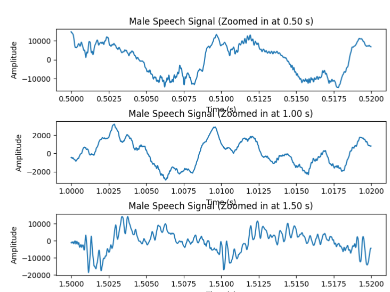



(c) Music signal

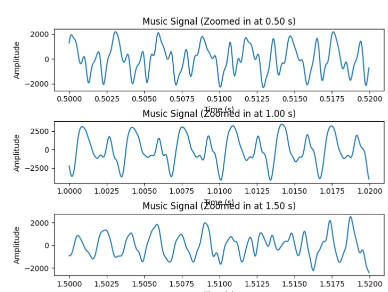
Figure 1: Plot of given audiofiles




(a) Female speech




(b) Male speech




(c) Music signal

Figure 2: Zoom on representative signal patches

## 2 Plot of the spectrograms

This is the used code:

Listing 3: Plot of spectrograms

```python
"""
Authors: Hailey Yu Haihong, Levi Leyh

This code analyzes speech and music signals by computing and plotting spectrograms and Mel-frequency cepstral coefficients (MFCCs.
It also normalizes the cepstrum and generates a cepstrogram displaying the normalized MFCC coefficients for each signal.
Correlation matrices for both the spectral and cepstral coefficient series are computed and plotted,
allowing for direct comparison between the two.
"""
import numpy as np
import matplotlib.pyplot as plt
import scipy.io.wavfile as wavfile
from scipy.signal import hann, spectrogram
from scipy.fftpack import dct
from python_speech_features import mfcc
from python_speech_features import logfbank


# Load the female and male speech signal and the music signal
female_fs, female_speech = wavfile.read('Sounds/female.wav')
male_fs, male_speech     = wavfile.read('Sounds/male.wav')
music_fs, music          = wavfile.read('Sounds/music.wav')

# Set spectrogram parameters
windowSize = 0.03
overlap = 0.015

# Compute spectrogram
female_windowSize = round(windowSize * female_fs )
female_overlap    = round(female_windowSize * 0.5)
female_F, female_T, female_S = spectrogram(female_speech, fs=female_fs, window=hann(female_windowSize),
                                           nperseg=female_windowSize, noverlap=female_overlap)

male_windowSize = round(windowSize * male_fs )
male_overlap    = round(male_windowSize * 0.5)
male_F, male_T, male_S = spectrogram(male_speech, fs=male_fs, window=hann(male_windowSize),
                                     nperseg=male_windowSize, noverlap=male_overlap      )

music_windowSize = round(windowSize * music_fs )
music_overlap    = round(music_windowSize * 0.5)
music_F, music_T, music_S = spectrogram(music, fs=music_fs, window=hann(music_windowSize),
                                        nperseg=music_windowSize, noverlap=music_overlap)


# Plot spectrograms
plt.subplot(3,3,1)
plt.imshow(20*np.log10(abs(female_S)), cmap='jet', aspect='auto', origin='lower',
           extent=[female_T.min(), female_T.max(), female_F.min(), female_F.max()])
plt.colorbar()
plt.xlabel('Time (s)')
plt.ylabel('Frequency (Hz)')
plt.title('spectrogram female speech')

plt.subplot(3,3,2)
plt.imshow(20*np.log10(abs(male_S)), cmap='jet', aspect='auto', origin='lower',
           extent=[male_T.min(), male_T.max(), male_F.min(), male_F.max()])
plt.colorbar()
plt.xlabel('Time (s)')
plt.ylabel('Frequency (Hz)')
plt.title('spectrogram male speech')

plt.subplot(3,3,3)
plt.imshow(20*np.log10(abs(music_S)), cmap='jet', aspect='auto', origin='lower',
           extent=[music_T.min(), music_T.max(), music_F.min(), music_F.max()])
plt.colorbar()
plt.xlabel('Time (s)')
plt.ylabel('Frequency (Hz)')
plt.title('spectrogram music')
```

## 3 Comparison of the spectrogram and the (normalized) cepstrogram

This is the used code:

Listing 4: Plot of female and male speech and the music signal (continued)

```python


# Compute MFCC coefficients
female_coeffs = mfcc(female_speech, female_fs, winlen=windowSize, winstep=overlap, numcep=13)
male_coeffs   = mfcc(male_speech, male_fs, winlen=windowSize, winstep=overlap, numcep=13)
music_coeffs  = mfcc(music, music_fs, winlen=windowSize, winstep=overlap, numcep=13)

print(len(female_coeffs),len(female_coeffs[0]))
print(len(female_S), len(female_S[0]))


# Normalize cepstrum
def normalize(mfcc):
    mean_cep = np.mean(mfcc, axis=0)
    std_cep = np.std(mfcc, axis=0)
    cepstrogram_centered = mfcc - mean_cep
    return cepstrogram_centered / std_cep

female_coeffs_norm = normalize(female_coeffs)
male_coeffs_norm   = normalize(male_coeffs)
music_coeffs_norm  = normalize(music_coeffs)

print(np.min(female_coeffs))


# Plot the MFCC coefficients and the normalized cepstrogtam
plt.subplot(3, 3, 4)
plt.imshow(female_coeffs.T, aspect='auto', cmap='jet', origin='lower',
           extent=[female_T.min(), female_T.max(), np.min(female_coeffs), np.max(female_coeffs)])
plt.xlabel('Time (s)')
plt.ylabel('MFCC coeffiencit')
plt.title('cepstrogtam female speech')
plt.colorbar()

plt.subplot(3, 3, 5)
plt.imshow(male_coeffs.T, aspect='auto', cmap='jet', origin='lower',
           extent=[male_T.min(), male_T.max(), np.min(male_coeffs), np.max(male_coeffs)])
plt.xlabel('Time (s)')
plt.ylabel('MFCC coeffiencit')
plt.title('cepstrogtam male speech')
plt.colorbar()

plt.subplot(3, 3, 6)
plt.imshow(music_coeffs.T, aspect='auto', cmap='jet',origin='lower',
           extent=[music_T.min(), music_T.max(), np.min(music_coeffs), np.max(music_coeffs)])
plt.xlabel('Time (s)')
plt.ylabel('MFCC coeffiencit')
plt.title('cepstrogtam music')
plt.colorbar()

# Plot the normalized mfcc
plt.subplot(3, 3, 7)
plt.imshow(female_coeffs_norm.T, aspect='auto', cmap='jet', origin='lower',
           extent=[female_T.min(), female_T.max(), np.min(female_coeffs_norm), np.max(
               female_coeffs_norm)])
plt.xlabel('Time (s)')
plt.ylabel('MFCC coefficient normalized')
plt.title('normalized cepstrogtam female speech')
plt.colorbar()

plt.subplot(3, 3, 8)
plt.imshow(male_coeffs_norm.T, aspect='auto', cmap='jet', origin='lower',
           extent=[male_T.min(), male_T.max(), np.min(male_coeffs_norm), np.max(male_coeffs_norm)
               ])
plt.xlabel('Time (s)')
plt.ylabel('MFCC coefficient normalized')
plt.title('normalized cepstrogtam male speech')
plt.colorbar()

plt.subplot(3, 3, 9)
plt.imshow(music_coeffs_norm.T, aspect='auto', cmap='jet',origin='lower',
           extent=[music_T.min(), music_T.max(), np.min(music_coeffs_norm), np.max(
               music_coeffs_norm)])
plt.xlabel('Time (s)')
plt.ylabel('MFCC coefficient normalized')
plt.title('cepstrogtam music speech')
plt.colorbar()

plt.show()
```

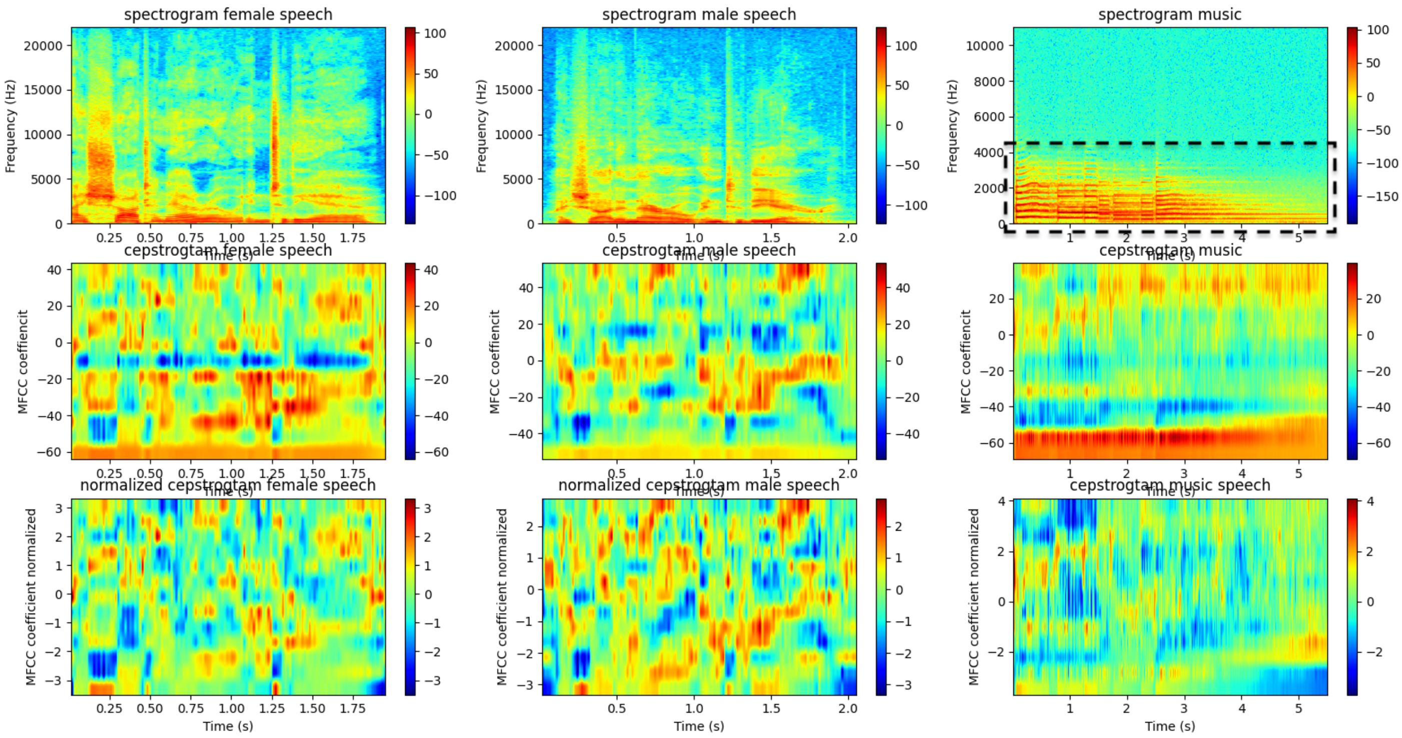



Figure 3: Plotted spectrograms and cepstrograms

In the music sample, harmonics are identified and marked by a dotted square. It's easy to see there are many horizontal lines in that area, which corresponds to harmonic components.

# 4 4. Correlation of the spectral and cepstral coefficient series

This is the used code:

Listing 5: Correlation of spectrograms and cepstrogram (continued)

```


# Compute the correlations
female_corr_spec = np.corrcoef(20*np.log10(abs(female_S)), rowvar=True)
male_corr_spec   = np.corrcoef(20*np.log10(abs(male_S)),   rowvar=True)
music_corr_spec  = np.corrcoef(20*np.log10(abs(music_S)),  rowvar=True)

female_corr_ceps = np.corrcoef(female_coeffs.T, rowvar=True)
male_corr_ceps   = np.corrcoef(male_coeffs.T,   rowvar=True)
music_corr_ceps  = np.corrcoef( music_coeffs.T, rowvar=True)

female_corr_ceps_norm = np.corrcoef(female_coeffs_norm.T, rowvar=True)
male_corr_ceps_norm   = np.corrcoef(male_coeffs_norm.T,   rowvar=True)
music_corr_ceps_norm  = np.corrcoef( music_coeffs_norm.T, rowvar=True)

# Plot the correlations
plt.subplot(3,3,1)
plt.imshow(np.abs(female_corr_spec), aspect='auto', origin='lower', cmap='gray')
plt.title('spectrum correltion female')
plt.colorbar()

plt.subplot(3,3,2)
plt.imshow(np.abs(male_corr_spec), aspect='auto', origin='lower', cmap='gray')
plt.title('spectrum correltion male')
plt.colorbar()

plt.subplot(3,3,3)
plt.imshow(np.abs(music_corr_spec), aspect='auto', origin='lower', cmap='gray')
plt.title('spectrum correltion music')
plt.colorbar()

plt.subplot(3,3,4)
plt.imshow(np.abs(female_corr_ceps), aspect='auto', origin='lower', cmap='gray')
plt.title('cepstrum correltion female')
plt.colorbar()

plt.subplot(3,3,5)
plt.imshow(np.abs(male_corr_ceps), aspect='auto', origin='lower', cmap='gray')
plt.title('cepstrum correltion male')
plt.colorbar()

plt.subplot(3,3,6)
plt.imshow(np.abs(music_corr_ceps), aspect='auto', origin='lower', cmap='gray')
plt.title('cepstrum correltion music')
plt.colorbar()

plt.subplot(3,3,7)
plt.imshow(np.abs(female_corr_ceps_norm), aspect='auto', origin='lower', cmap='gray')
plt.title('normalized cepstrum correltion female')
plt.colorbar()

plt.subplot(3,3,8)
plt.imshow(np.abs(male_corr_ceps_norm), aspect='auto', origin='lower', cmap='gray')
plt.title('normalized cepstrum correltion male')
plt.colorbar()

plt.subplot(3,3,9)
plt.imshow(np.abs(music_corr_ceps_norm), aspect='auto', origin='lower', cmap='gray')
plt.title('norlamized cepstrum correltion music')
plt.colorbar()

plt.show()
```

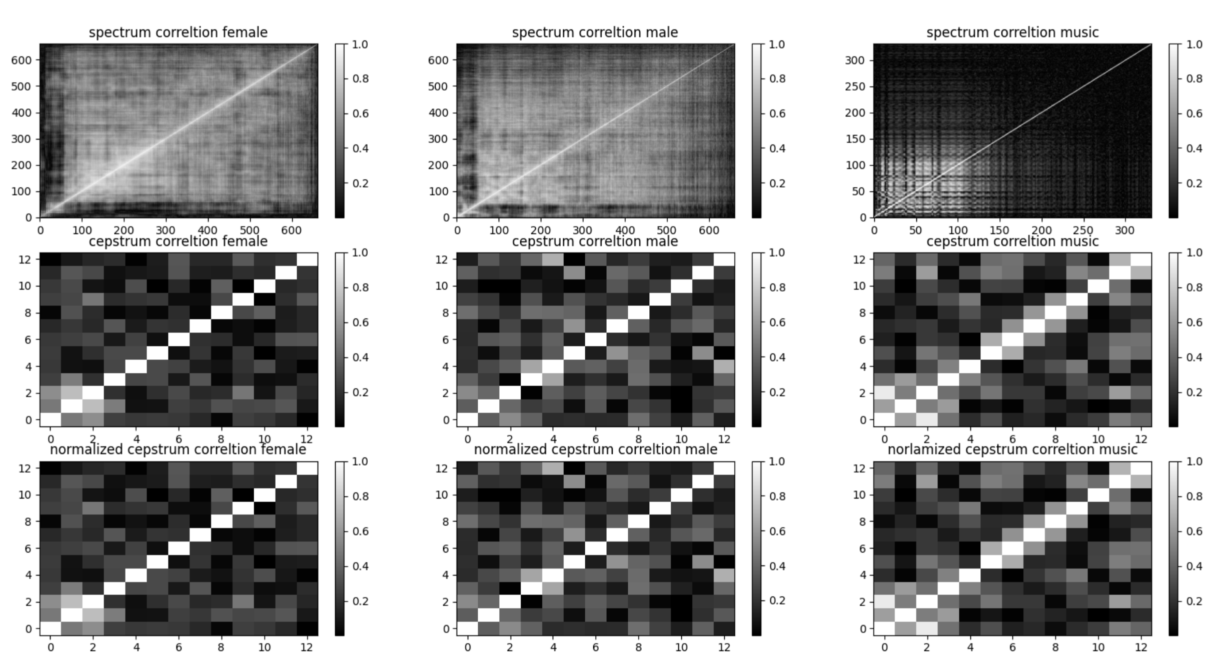



Figure 4: Plotted correlation matrices

## 5 Questions in the text

Which representation do you think is the easiest for you, as a human, to interpret, and why?

The easiest representation for a human to interpret is the spectrogram because it provides a visual representation of the frequency content of a signal over time. This allows for easy identification of patterns and variations within the signal. For example, it's very easy to find the harmonics in spectrogram by searching for horizontal lines. On the other hand, the cepstrogram shows the spectral envelope of a signal, which is a less intuitive concept for most people.

Can you see that the male and female spectrograms represent the same phrase? Could a computer discover this? Why/why not?

Yes, you can see that the male and female spectrograms represent the same phrase from the figures and and can be easily distinguished from the music signal. A computer can discover this as well depending on the quality of the samples and the algorithms used for analysis. Humans are generally better at recognizing patterns and making judgments based on context, while computers are better at analyzing large amounts of data quickly and accurately. When comparing spectrograms and checking for similar phases in two audio files, humans can interpret the content of the audio and identify patterns in the spectrograms based on their knowledge of language and speech and their perception of sound, while computers are better suited for tasks like identifying specific patterns or comparing large datasets but may struggle to interpret the context or meaning behind the audio and may not recognize subtle variations or nuances.

Can you see that the male and female cepstrograms represent the same phrase now? What about a computer?

The cepstrogram is more difficult for human to interpret. Even looking at the normalized cepstrogram, it's hard for human-eye to capture similarities in this complex pattern. However, it would be easier for computers, which can convert visual data into numbers and calculations. We can compute the distance between two cepstral vectors using techniques such as Euclidean distance or cosine similarity. If the distance between two cepstrograms is small, it indicates that they are similar. Additionally, we can use machine learning algorithms such as k-NN, SVM, or neural networks to train a model to classify cepstral vectors as similar or dissimilar.

Which matrix, spectral or cepstral, looks the most diagonal to you?

For speech signals, the cepstral matrix looks more diagonal. There's a distinct contrast between the diagonal line (bright) and the rest(dark), which means that the correlation between the coefficients are very small. In contrast, the spectral matrix is quite bright not only in the diagonal line, implying that its coeffieicnts are stongly correlated. As for the music signal, there's not much difference between spectral and cspstral. Maybe this id becasue MFCC is more for speech signals rather than music.

## 6 Further thoughts

Some thoughts on the possibility of confusing the MFCC representation in a speech recognizer. Can you think of a case where two utterances have noticeable differences to a human listener, and may come with different interpretations or connotations, but still have very similar MFCCs? (Hint: Think about what information the MFCCs remove.) What about the opposite situation—are there two signals that sound very similar to humans, but have substantially different MFCCs?

Yes, two utterances can have noticeable differences to humans but still have very similar MFCCs. This is because MFCCs remove information that is important for human perception but not necessarily for speech recognition. Conversely, two signals that sound similar to humans may have substantially different MFCCs, due to differences in acoustic characteristics that are not perceptually salient to humans but are captured by the MFCCs.