EEC 201 Final Project Report

By Christopher Tran, Alec Applegarth

Overview

This document is organized by the given tests, with each individual test answered followed by the given prompt.

Execution Instructions

- Test 2
  - To see part 1 of test 2, run the test2.m file
  - To see part 2 of test 2, run the test2p2.m file
- Test 3
  - To see the mel-spaced filterbank responses and the spectrogram, run the test3.m file
- Test 4
  - Unable to execute independently, but is located in get_mfccs.m
- Test 5 and 6
  - Run the eec201_project_2.m file
- Test 7
  - To switch between the data sets, uncomment/comment out the desired data set. For the given data set, use a num_speakers value of 8, for the students data set, use a num_speakers value of 19
- Test 8
  - Comment in the code under “Apply FIR notch filter” in get_mfccs.m
- Test 9
  - Run speaker_id_test9.m, which uses 2 other folders than the main ones for the combined source signals.
- Test 10
  - In speaker_id, switch path_train & path_test to the section under “For Test 10”

NOTE: Whenever the directory is changed for the dataset, the “num_speakers” variable must be updated.

TEST 1

To begin, use the voice files of "zero" that we provided (not from the class). Play each sound file in the TRAIN folder. Can you distinguish the voices of the given 11 speakers in the database? Next play each sound in the TEST folder in a random order without looking at the ground truth and try to identify the speaker manually. Record what is your (human performance) recognition rate. Use this result as a later benchmark.

Each sound in the train folder was distinguishable. Our human performance ended up being 87.5%, getting 7 out of 8 cases right.

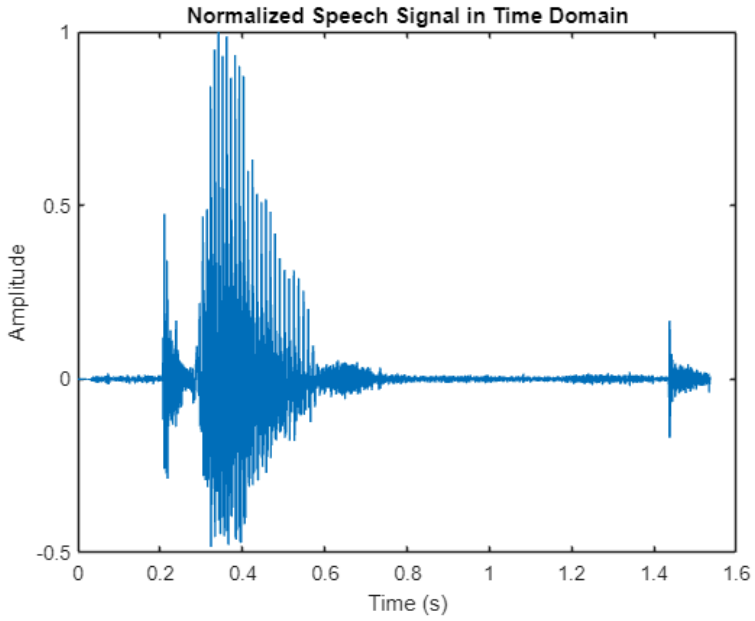
TEST 2

In Matlab one can play the sound file using “sound”. Record the sampling rate and compute how many milliseconds of speech are contained in a block of 256 samples? Now plot the signal to view it in the time domain. It should be obvious that the raw data is long and may need to be normalized because of different strengths.

Using `'Twelve_test1.wav':`
In a block of 256 samples, we calculated the duration of speech to be 5.33 ms. This was done using:

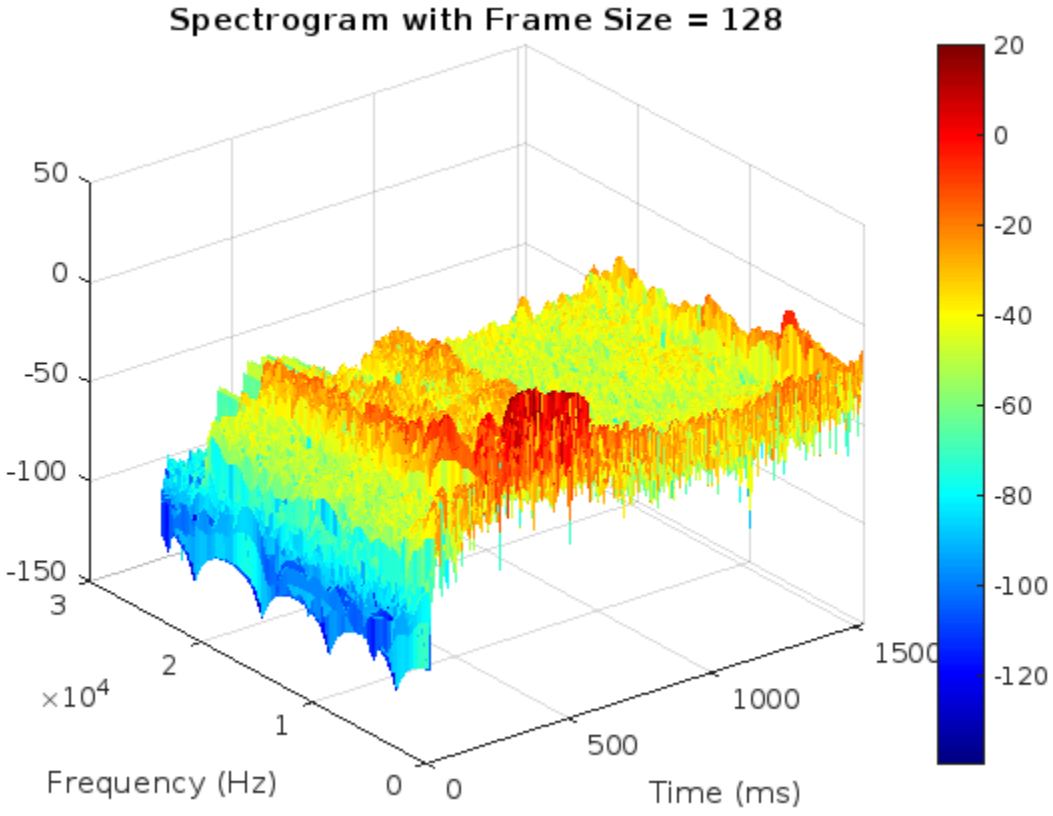
```
duration_ms = (block_size / sampling_rate) * 1000
```

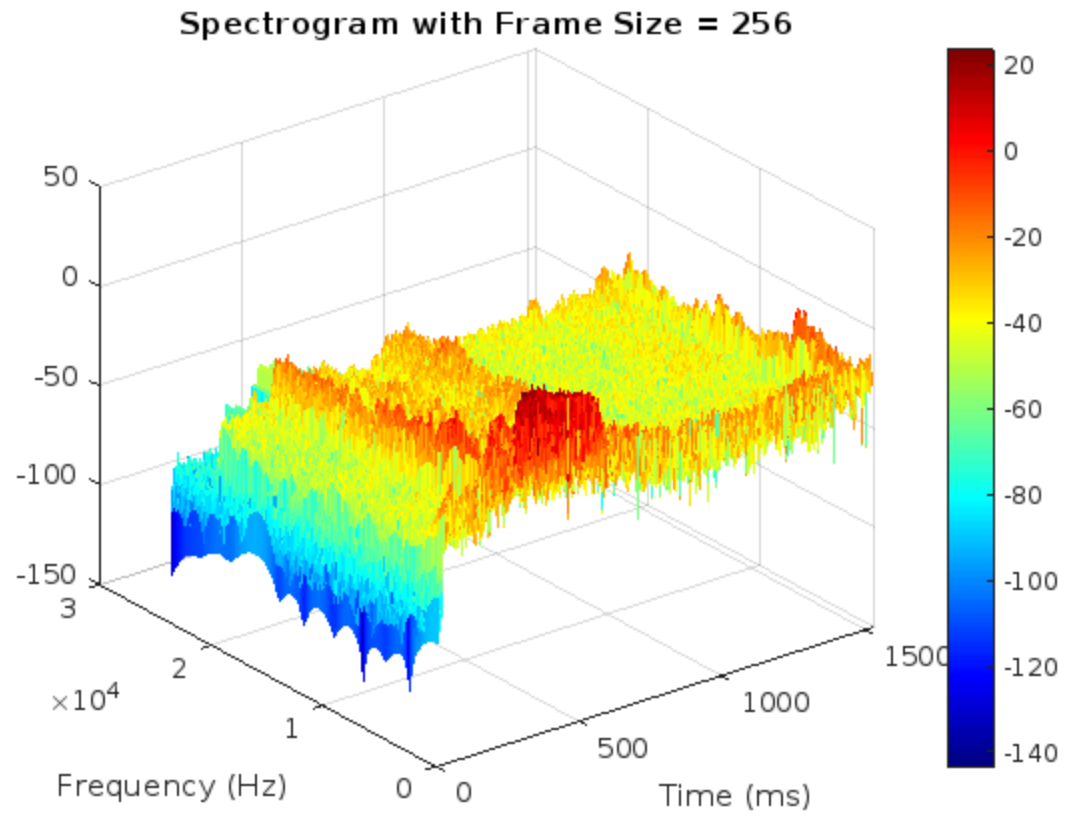
Signal in time domain:

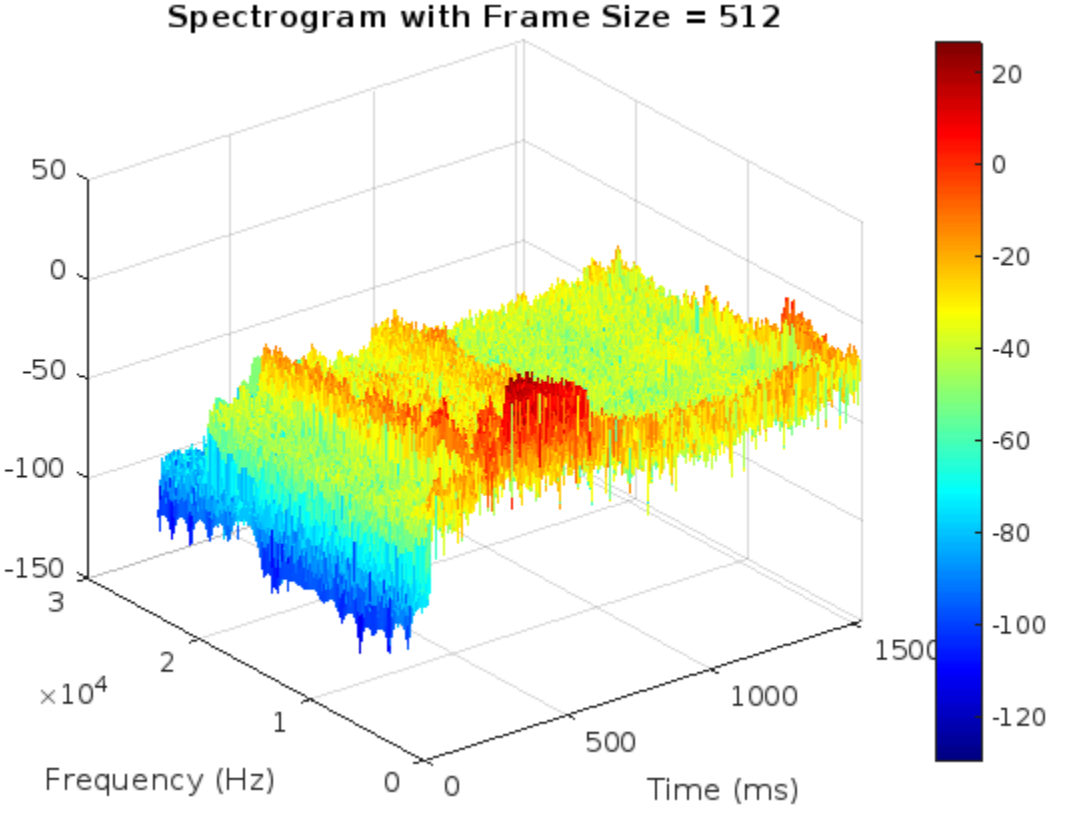


Use STFT to generate a periodogram. Locate the region in the plot that contains most of the energy, in time (msec) and frequency (in Hz) of the input speech signal. Try different frame sizes: for example N = 128, 256 and 512. In each case, set the frame increment M to be about N/3.

The region with the most energy ended up being at time 393.65 ms, and at frequency 656.25 Hz. To verify our results, we also used the spectrogram() function with the same parameters, which ended up giving us the same results. We used a for loop iterating over 128, 256, 512 for frame sizes to determine the max energy.




Spectrogram with Frame Size = 256
50
0
-50
-100
-150
3
2
1
0
×10^4
Frequency (Hz)
0
500
1000
1500
Time (ms)
20
0
-20
-40
-60
-80
-100
-120
-140

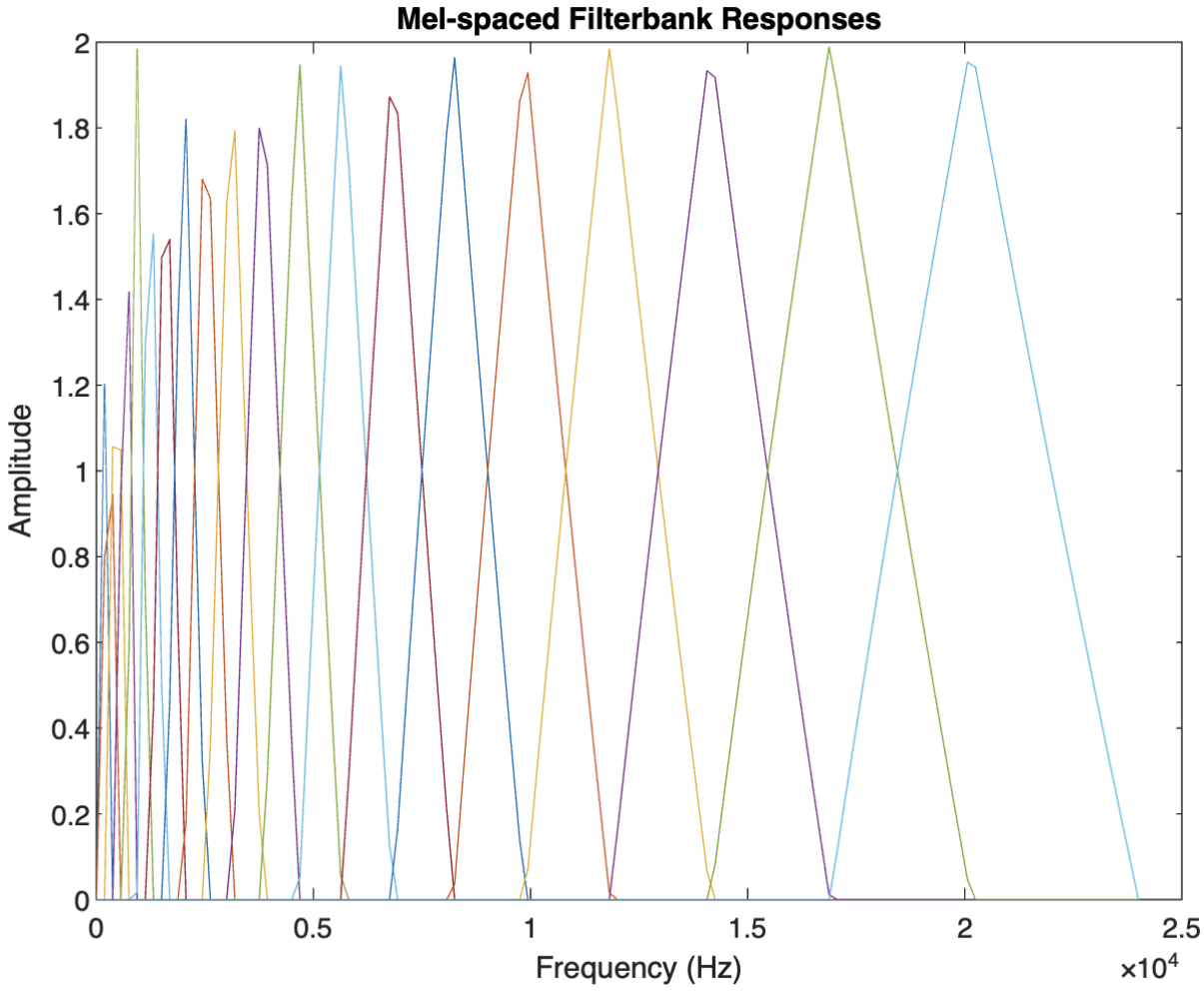

Spectrogram with Frame Size = 512
50
0
-50
-100
-150
3
2
1
0
×10^4
Frequency (Hz)
0
500
1000
1500
Time (ms)
20
0
-20
-40
-60
-80
-100
-120

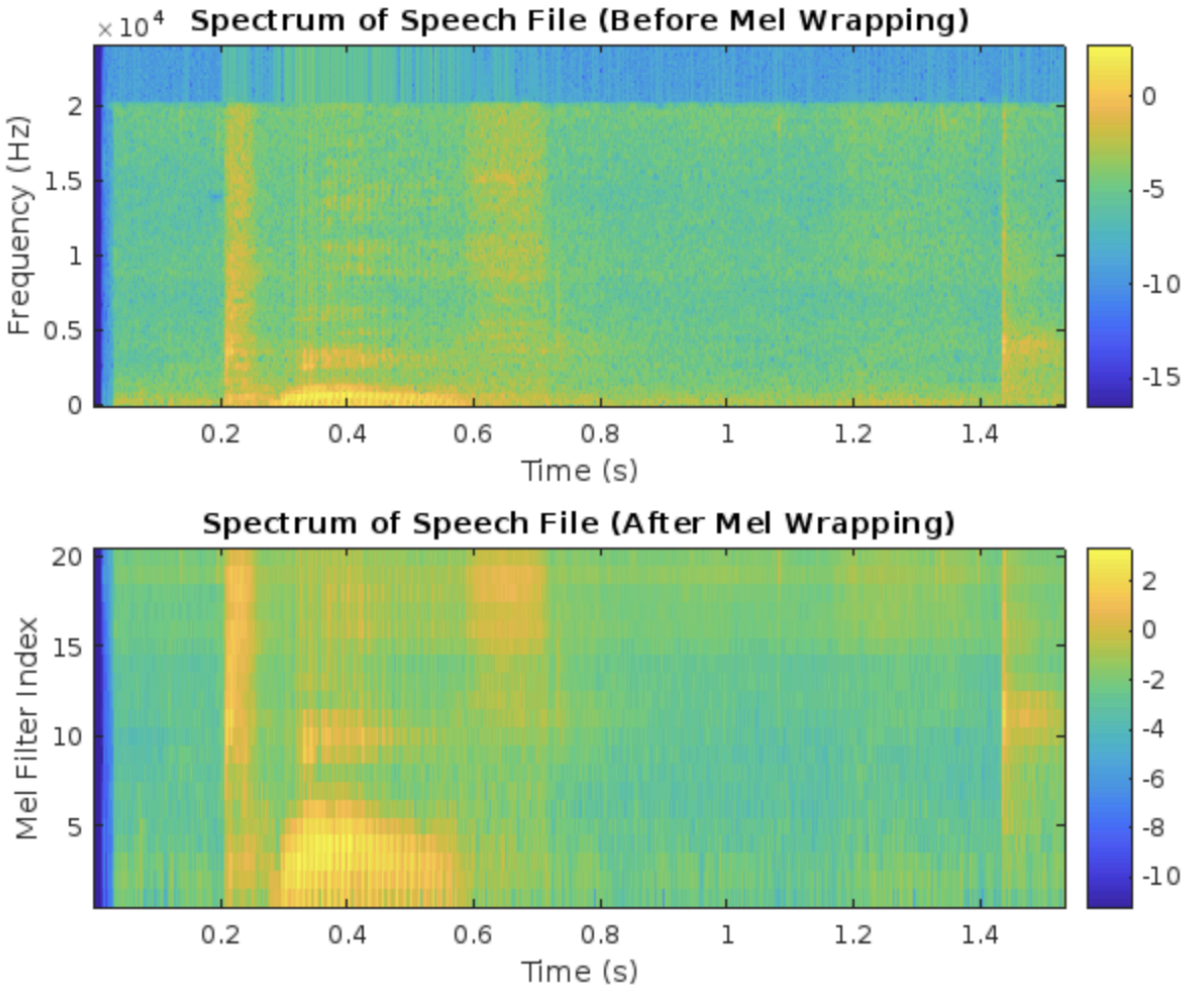
TEST 3

Plot the mel-spaced filterbank responses. Compute and plot the spectrum of a speech file before and after the mel frequency wrapping step. Describe and explain the impact of the melfb.m or melfbown.m program

Using `'Twelve_test1.wav'`:

To generate the mel-spaced filterbank responses, we used the given melfb example.





After the melfb.m program, it helps us view the lower frequencies more in line with human perception of frequency. This is why the resolution of lower frequencies is increased after Mel wrapping.

TEST 4

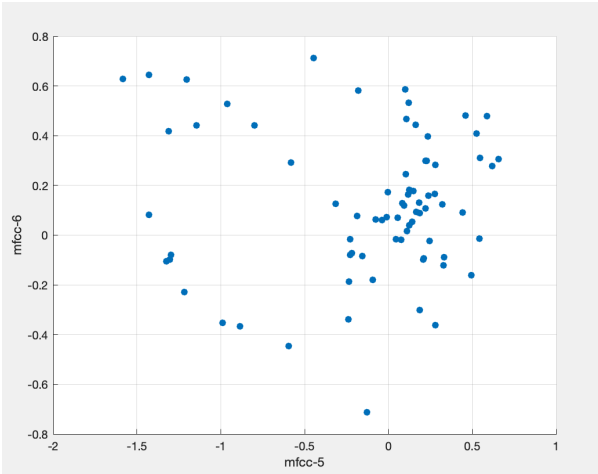
Complete the "Cepstrum" step and put all pieces together into a single Matlab function, e.g., mfcc.m

To complete this portion, we used the recommended DCT function to convert to the time domain.

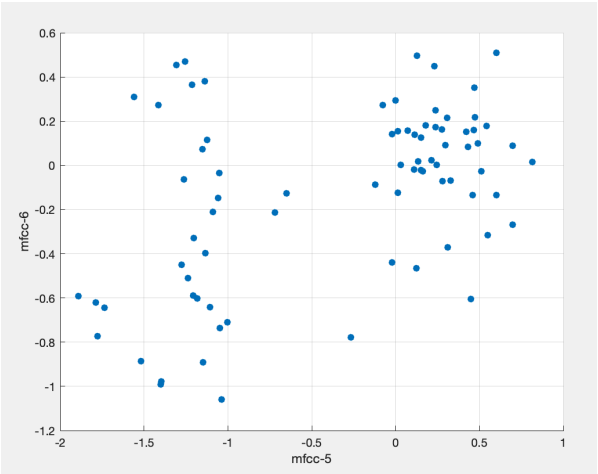
TEST 5

Now apply a VQ-based pattern recognition technique to build speaker reference models from those vectors in the training set before identifying any sequences of acoustic vectors from unmarked speakers in the test set.

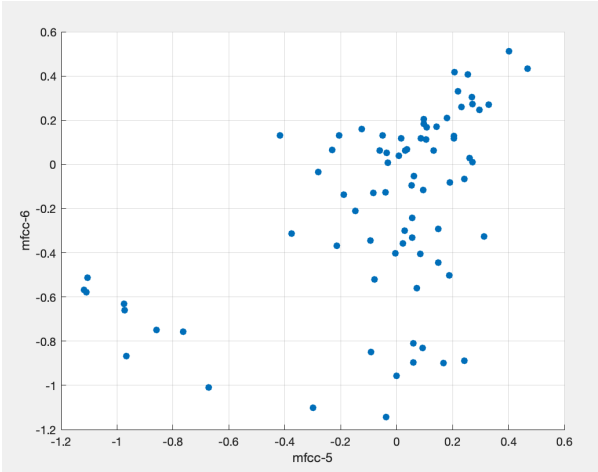
To check whether the program is working, inspect the acoustic space (MFCC vectors) in any two dimensions in a 2D plane to observe the results from different speakers. Are they in clusters?



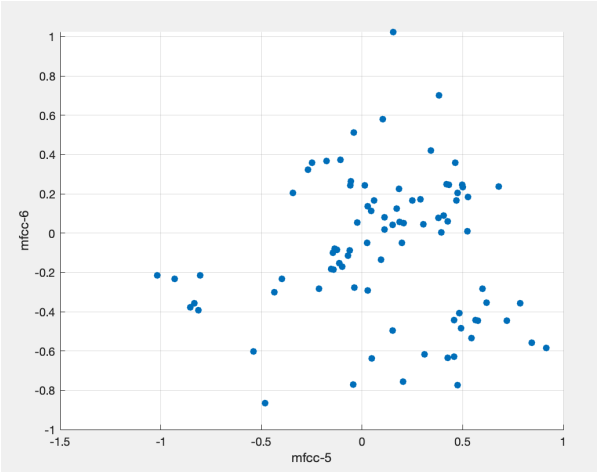
Given speaker 1



Given speaker 2



Given speaker 3



Given speaker 4

From the four given speaker plots, we can see that they are in clusters.

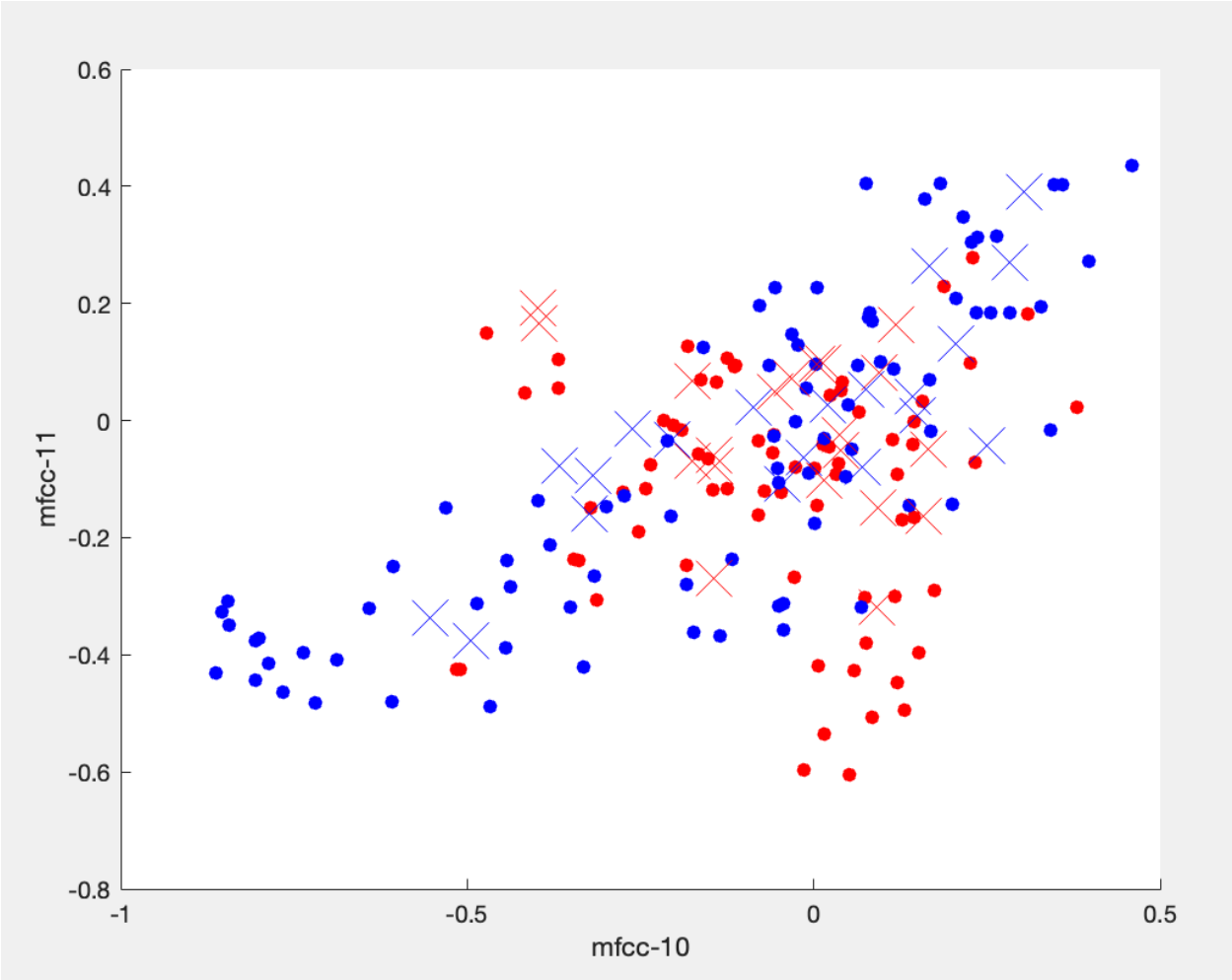
Now write a function that trains a VQ codebook using the LBG algorithm.

This function can be found on our repo under ‘codebook_generate.m’

TEST 6

Plot the resulting VQ codewords using the same two dimensions over the plot of in TEST 5. You should get a figure like Figure 4.

To get a figure similar to figure 4, we graphed the MFCC vectors of two speakers (represented by blue and red) and their respective centroids, marked by the X’s. We removed the first MFCC, as it is the DC component and was contributing heavy outliers.



Speakers 1 (red) and 7 (blue) graphed on MFCCs 10 and 11 with their respective code vectors

TEST 7

Using the programs to train and test speaker recognition on the data sets.

Record the results. What recognition rate can your system achieve? Compare this with human accuracy. Experiment and find the reason if high error rate persists. Record more voices of yourself and your teammates/friends. Each new speaker can provide one speech file for training and one for testing.

With default values:

```
num_clusters = 20;
num_mel_coeffs = 20;
frame_size = 256;
overlap_size = 100;
epsilon = 0.01;
window = hanning(frame_size);
```

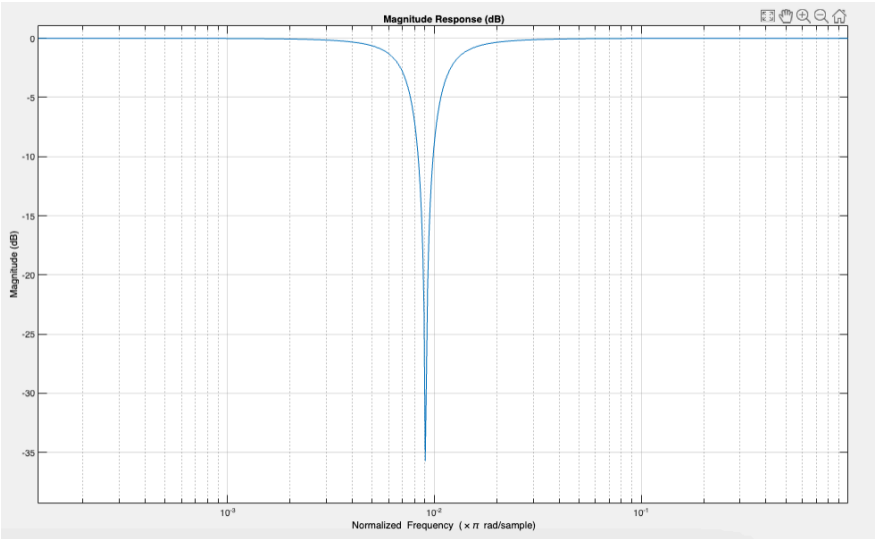
We recorded the following results on the labeled data sets:

Data Set	Recognition Rate (Default)	Recognition Rate (Optimized)
GivenSpeechData	87.50% (7/8)	87.50% (7/8)
StudentAudioRecording (Twelve)	77.77% (14/18)	88.88% (16/18)
StudentAudioRecording (Zero)	77.77% (14/18)	88.88% (16/18)

The error rate seems to be partially determined by the set of random vectors chosen for the initial codebook. This can result in slight variability between different executions of the program. The percentages listed above are the most consistent, but have seen some variation between executions. Another plausible reason for error is that the voices are similar enough that they may be characterized similarly. Outdoor ambience could also be a factor.

TEST 8

Use notch filters on the voice signals to generate another test set. Test your system on the accuracy after voice signals have passed different notch filters that may have suppressed distinct features of the original voice signal. Report the robustness of your system.



A notch filter at a frequency of 200 Hz and a bandwidth of 100Hz was applied to the input audio signal. The lower frequencies of the human voice are more impactful in characterizing a voice, and 200 Hz is an essential frequency band, especially for lower voices.

Data Set (after notch filtering)	Recognition Rate (Default)
GivenSpeechData	87.50% (7/8)
StudentAudioRecording (Zero)	88.88% (16/18)
StudentAudioRecording (Twelve)	83.33% (15/18)

A change was only observed in the StudentAudioRecording dataset for the recordings of ‘Twelve’, where a single speaker was dropped as a result of the notch filtering. Even if the MFCCs of some of the lower frequencies change, the system is still able to characterize the voice from the other MFCCs computed.

TEST 9

Assuming that your system has been trained with the non-students' "zero" speeches, we now augment the problem by adding 10 students' voice as follows: (a) Randomly select speech “zero” of 10 students from EEC 201 we recorded twice: one for training and one for recognition test. (b) Next, retrain your system by adding our own recorded speech to our existing speakers’ samples. (c) Test the accuracy of your system and compare the accuracy obtained previously. NOW this system can identify more speakers that include the original set of speakers + the 10 students.

Data Set	Recognition Rate (Default)	Recognition Rate (Optimized)
GivenSpeechData + Zero	83.33% (15/18)	83.33% (15/18)

The accuracy of the system with the randomly selected students combined with the non-students lead to a similar result as test 7. This shows that our system is able to handle a variety of potential sources in terms of recognition, while still being able to maintain a consistent rate of accuracy.

TEST 10

Use all samples of EEC 201 students saying “zero” and “twelve”. Retrain the speech recognition system and test the accuracy in terms of the accuracy to identify the different "zero" and "twelve" sounds. Question 1: If we use "twelve" to identify speakers, what is the accuracy versus the system that uses "zero"? Question 2: If we train a whole system that tries to identify a) which speaker, and b) whether the speech is "zero" or "twelve", how accurate is your system?

Question 1):
Trained completely using “Twelve”:

Data Set	Recognition Rate (Default)
StudentAudioRecording (Twelve)	88.88% (16/18)
StudentAudioRecording (Zero)	38.88% (7/18)

Trained completely using “Zero”:

Data Set	Recognition Rate (Default)
StudentAudioRecording (Twelve)	88.88% (16/18)
StudentAudioRecording (Zero)	33.33% (6/18)

The accuracy was the same in identifying the same word (i.e. Training “Twelve” → Testing “Twelve” but the accuracy decreased significantly when mismatching words were used).

Question 2):
This question treats each speaker / word combination as their own codebook / object of identification. So, all samples can be pulled into larger test/train folders to use the same, unmodified code to run this test.

Trained with both models to identify both twelves and zeros: 88.88% accuracy (32/36).