LAB 2 - TASK 1 (1 point possible)

In this lab you will observe the frequency analysis for a given speech signal, which is performed by completing the following 3 steps:

1. divide the signal into fixed frames and compute the power of each frame;
2. generate the amplitude spectrum for the extracted frame;
3. approximate the original frame signal using the most significant frequency components.

In this task, we will examine the effect of using different frame lengths.

```
% load a speech signal
[x,Fs] = audioread('test.wav');

% % % % Revise the following code  % % % %
% define the frame length

frame_length = 32;

% % % % Do not change the code below % % % %

% compute power of each frame
frame_power = get_frame_power(x,frame_length);

% plot original waveform and power in each frame
figure(1);
plot_speech_power(x,frame_power,frame_length);
```

Unanswered

Figure 1

tutorials

- MATLAB Sandbox

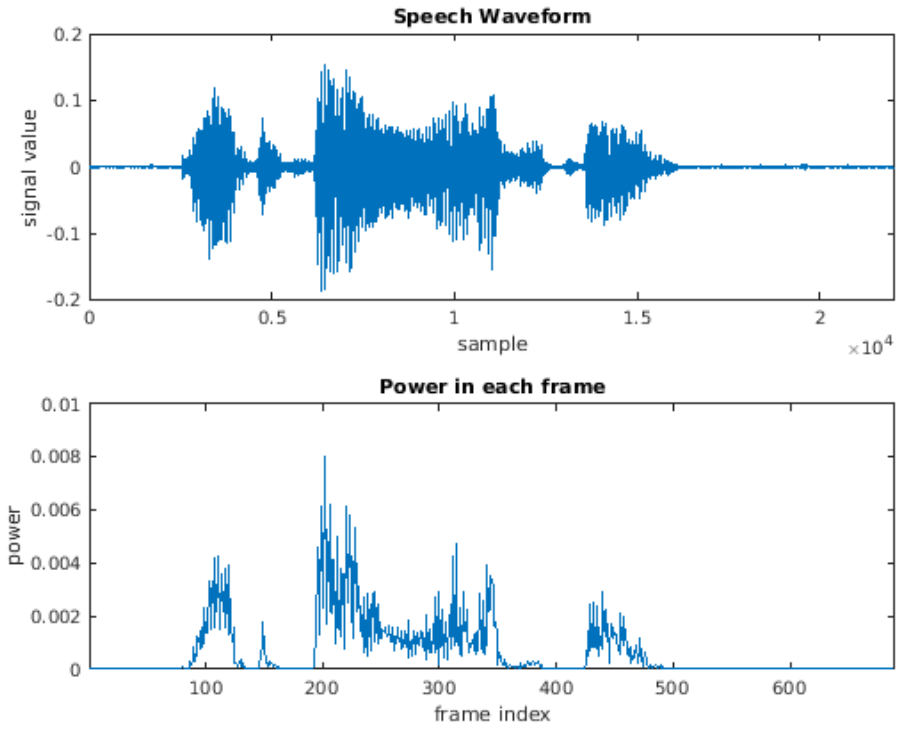
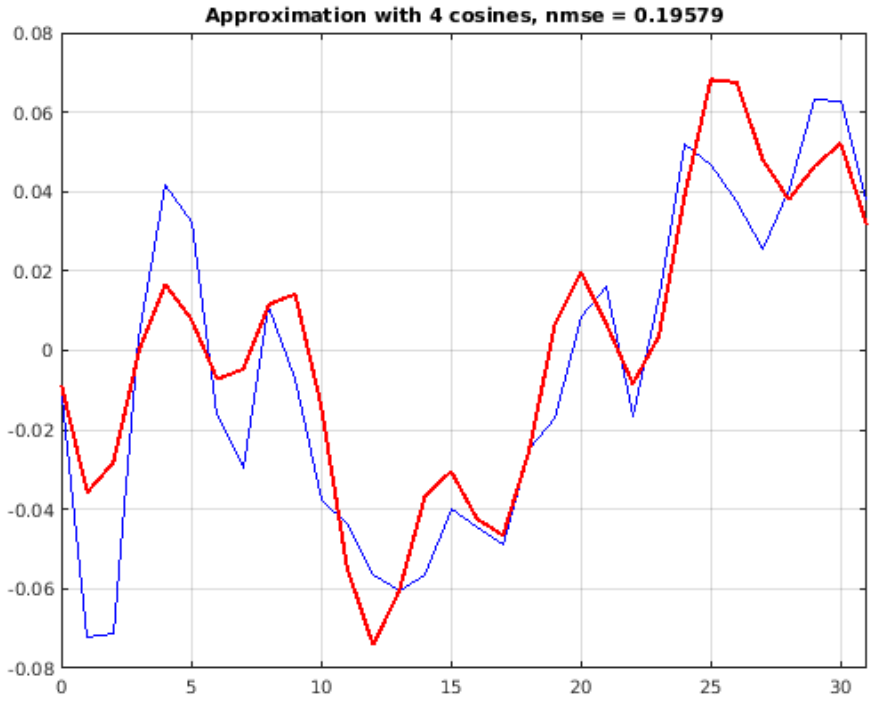


Figure 2



Run Code

You have used 0 of 10 submissions

INSTRUCTIONS

The MATLAB code in the above window is used to study the input signal in both the time and frequency domains. The code first loads a speech signal, corresponding to the word “HKUST”, from the file "test.wav" into a variable **x**. To hear the waveform, click below.

The input signal is composed of 22050 samples acquired with a sampling frequency (**Fs**) of 11025 Hz, which is one quarter the sampling frequency used in audio CDs. A speech signal is normally highly variable over a long time, however it is almost periodic over a short period, especially for voiced speech. Thus, to analyze speech signal, we break it into a set of short segments of fixed length, called frames. As a result, we can calculate the Fourier series for one frame, extract the most important frequency components and approximate the frame as a sum of sinusoidal functions.

In the initial code, we defined the frame length (variable **frame_length**) to be about 186 ms (2048 samples). The first frame consists of the first 2048 samples. The second frame consists of the next 2048 samples. Thus, for **frame_length** = 2048, the entire waveform contains floor(22050/2048) = 10 frames (the samples at the end that do not fill an entire frame are discarded). The floor function returns the largest integer less than the argument.

Then we use the function **get_frame_power** to get the power of each frame and we visualize the speech with the function **plot_speech_power**. This function generates two subplots. The first one plots the entire speech waveform, **x**. You can see a large amplitude region near the center of the signal, corresponding to the two letters “US” in "HKUST". The second subplot shows the signal power in each frame.

Then, we select the frame with the highest power using the function **max** and we extract the corresponding samples from the input signal using the function **extract_frame**. At this point, we approximate the waveform of the selected frame using the 4 most important (largest amplitude) frequency components of the frame. It means the we approximate the frame using a sum of 4 sinusoidal functions whose amplitude, frequency and phase are given by the frequency analysis of the

frame. Finally, we compare the approximated frame with the original one using the function **plot_approx**. In Figure 2, the original waveform of frame is plotted in blue. The approximate waveform is plotted in red. The title of the figure shows the **nmse** value, which stands for "Normalized Mean Squared Error" and measures the difference between the original and the approximate waveform.

The functions **get_frame_power**, **plot_speech_power**, **extract_frame** and **plot_approx** were written for this lab.

In the initial code the frame length is too long. Thus, the signal in the frame does not appear very periodic. Try changing the frame length to successively smaller powers of 2 (e.g. 1024, 512, etc.) and observe the shape of the waveform and how well the approximation matches it. As the frame length gets smaller, the approximation improves.

Your final task is to find the frame length that contains 2^n samples, where n is an integer, and for which the corresponding time interval is the closest to 25ms. This interval is short enough that the signal is fairly periodic, but also long enough that the important frequency structure can be extracted by the Fourier analysis. You will need to use the sampling frequency to compute the actual length of the frame in seconds. Once you have changed the variable **frame_length** to this desired value submit your code to get credit.


Download on the
App Store


ANDROID APP ON
Google play