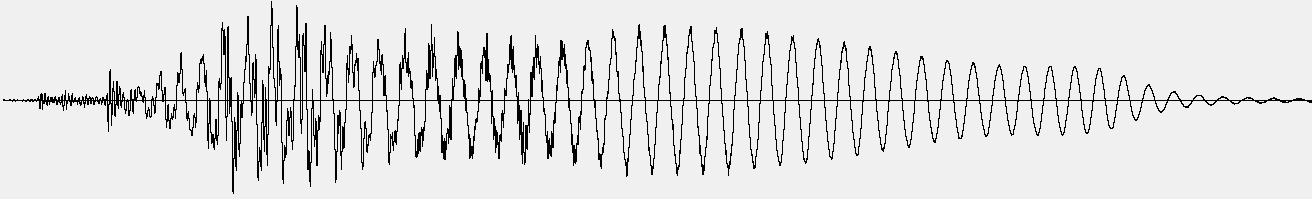
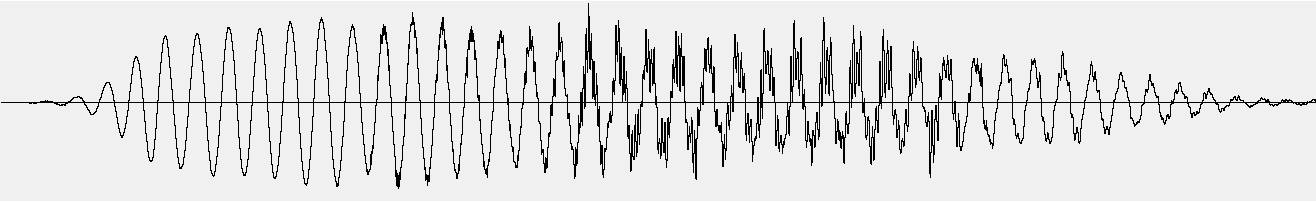
**Project-4 Spectrogram**

1. **Objectives**
2. Be familiar with Discrete Fourier Transform (DFT).
3. Understand how a spectrogram is generated.
4. Mastery over the C language.
5. Start to learn “shell script.”
6. Start to use popular python library “Matplotlib.”
7. **Background**

Fig.1 shows waveforms of a compound vowel ‘ai’ and a syllable ‘ya’. The compound vowel ‘ai’ and the syllable ‘ya’ have very similar sound components ‘a’ and ‘i’, but are in different order over time. It can be obviously observed that the waveform in the beginning of ‘ai’ and the one in the end of ‘ya’ are very similar, that is, they may have similar harmonic component, since they all are the sound ‘i’ (vice versa for the case of the end of ‘ai’ vs. the beginning of ‘ya’. These two waveforms are different speech segment. But, if you take Fourier transform for ***whole segments of ‘ai’ and ‘ya’***, they may have very similar frequency responses in Discrete-Time Fourier Transforms (DTFTs), indicating that we cannot distinguish the two sound ‘ai’ and ‘ya’ if the analysis window is too long so as to make Fourier analysis lose ***time resolution*** due to the fact that speech signals change over time.



(a)



(b)

Fig.1: (a) waveform of a compound vowel ‘ai’ and (b) waveform of a syllable ‘ya’. Note that ‘ai’ consists of two sounds ‘a’ and ‘i’ in sequence and ‘ya’ consists of two sounds ‘i’ and ‘a’ in sequence.

To catch the change in frequency components over time, as shown in Fig. 2, ***Short-Time Fourier Analysis*** or ***Short-Time Fourier Transform (STFT)*** is adopted to analyze a speech signal by decomposing the speech signal into a series of short segments, referred to as ***analysis frames***, and analyze each one independently. Apparently, the waveform in each short segment (analysis frame) could be more stationary since waveforms in a shorter window look more periodic and have more similar harmonic components. Also, each analysis frame could catch different waveform so as to observe signal changing over time. We, therefore, can analyze phonemes and their transition by spectrogram.

A spectrogram is a visual representation of a ***spectrum sequence***, capturing spectrum changes with time. Spectrograms are sometimes called sonogram, spectral waterfalls, voiceprints, or voicegrams. In Digital Signal Processing (DSP), we can regard spectrogram, , as a function of time (frame index) and frequency bin, i.e.

 (1)

where *m* represents frame index; *k* represents frequency bin index. Note that  in Eq. 1 is in the same form with DFT.  is a short-time signal with a length of *N* samples, indicating taking DFT of a discrete time signal of length *N*. We usually call the parameter *N* ‘*FFT window length*’. Note that FFT stands for Fast Fourier Transform which is an efficient implantation of DFT. In mathematics,  is expressed by

 (2)

where  denotes a time-shifted signal of original input speech, ; *M* represents *frame interval* in sample;  is a window function, a mathematical function with zero-valued outside of some chosen interval. Therefore, we can regard  as *m*-th frame signal of length *N*. There are many types of window functions, such as rectangular window (Eq. 3), Hamming window (Eq. 4), and Hanning window (Eq. 5).

Fig.2: Overview of obtaining spectrogram. The process in sequence, includes framing, taking window, zero padding, and DFT or FFT.

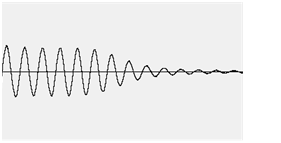
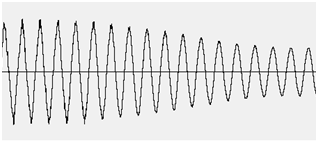
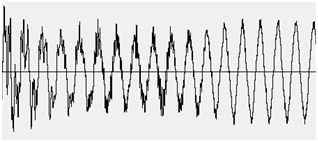
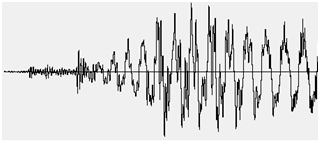
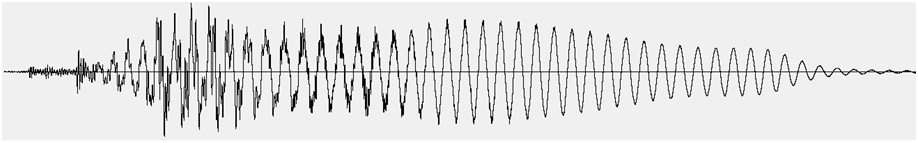




0

P-1

*n*





0

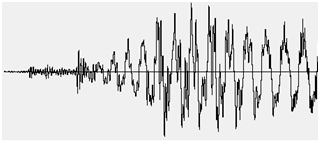
P-1

0

M

2M

3M





0

P-1

N-1









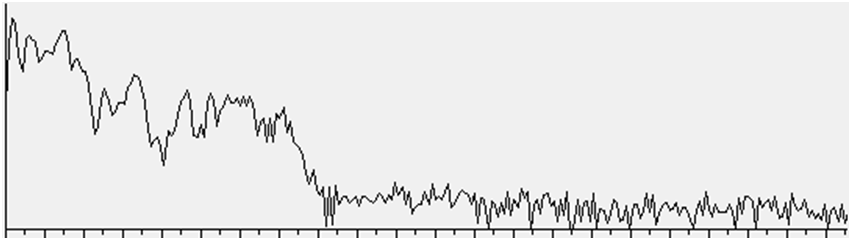


**framing**

**Taking window**

**Zeros padding**

**DFT or FFT**



0

N/2-1





*k*



frame 0

frame 1

frame 2

frame 3

Rectangular:  (3)

Hamming:  (4)

Hanning:  (5)

For analysis of speech, we usually choose Hamming window as the window function in Eq. 2. Note that in Eqs 3-5, a window function has zero-value outside of an interval . We therefore define *P* as ***analysis window size***, regarding how many sample points are excerpted from the analyzed signal  for an analysis window. Thus, the value *P* must be equal or smaller than the FFT window length *N*. An analyzed signal is said to be zero-padded as *P*<*N* because  is zero for . Since we only have non-zero samples for , only *P* points excerpted from  are informative for frequency analysis. As FFT window length *N* is set to be greater than analysis window size *P*, the *N*-*P* zero-padded samples in  just let the spectrum of *m*-th frame  has a smoother outline extrapolating more frequency bins.

1. **Parameters of spectrogram by DFT**

* Analysis Window Types

***Why not using rectangular window for spectrogram?*** Assume we have a pure sinusoidal signal to be analyzed by spectrogram, e.g.

 (6)

and the corresponding Fourier transform is

 (7)

But that spectrogram analyzes the windowed signal  and this windowed signal is a distorted version of original pure sinusoidal signal . Also, let’s recall the convolution theorem, i.e.

 (8)

From Eq. (9), we know that the DTFT  depends on the DTFT  (DTFT of the window function ). So, let’s check the  if  is a rectangular function of length *P*:

 (9)

Apparently,  is in a form of a sinc function. **So….Think about the rest by yourself. I talked too much!**

* Window Size *N*, frequency bin index *k*, and analysis bandwidth

The parameter *N* is the number of points for DFT analysis (sometime called ‘***window size***’). Since each frequency bin index *k* in  corresponds to a frequency component of Hz of continuous-time signal (: sample rate), larger *N* means more points analyzed and more sophisticated frequency resolution. We usually define the term (Hz) as ***analysis bandwidth***.

* Frame Interval *M*

The parameter *M* is the ***frame interval*** regarding rate of DFT analysis on time axis. Smaller *M* means more detailed time resolution for spectrogram. Conventionally, *M* is set to be equal or smaller than *N*, indicating each analysis frame  could have some overlapping samples to the neighboring analysis frames (e.g. ..., ,…). Typically, ***window size*** is set to be 20ms~30ms, that is *N*=160 points~240points for an 8kHZ sample-rate speech signal, or *N*=320 points~480points for a 16kHz sample rate signal. ***Frame interval*** is usually set to be 5ms~20ms, that is M=40points~160points for an 8kHZ sample-rate speech signal, or *M*=80 points~320points for a 16kHz sample rate signal.

1. **Implementations**
   1. Spectrogram by ***DFT***

Steps:

* + 1. Generate two waveforms (\*.wav) that satisfy the following specifications:
       1. Sample rates: 16,000 Hz or 8,000 Hz
       2. Bit depth: 16 bits
       3. Digital signal that represents

; where is the unit step function; are amplitudes; are frequencies in Hz; are the waveforms generated by the function generator:

* + - 1. Please name the two wave files s-{$smp\_rate}.wav for {$smp\_rate}={“8k”, “16k”}.
    1. We also provide 2 wave files that contain 5 basic vowels with 4 tones in 8kHz and 16kHz sample rates. The two files are named aeueo-16kHz.wav and aeueo-8kHz.wav
    2. Save the spectrograms  for all the WAVE files with the following settings, to files in ascii:

Setting 1:

Analysis window size = 32ms

Analysis window type = rectangular

DFT/FFT window size = 32ms

Frame interval = 10ms

Setting 2:

Analysis window size = 32ms

Analysis window type = hamming

DFT/FFT window size = 32ms

Frame interval = 10ms

Setting 3:

Analysis window size = 30ms

Analysis window type = rectangular

DFT/FFT window size = 32ms

Frame interval = 10ms

Setting 4:

Analysis window size = 30ms

Analysis window type = hamming

DFT/FFT window size = 32ms

Frame interval = 10ms

Therefore, 16 ascii files are saved: s-16k.{Set1~Set4}.txt, s-8k.{Set1~Set4}.txt, aeueo-16kHz.{Set1~Set4}.txt, and aeueo-8kHz.{Set1~Set4}.txt,

* + 1. Compare the results by Settings 1-4 and discuss the differences and their significance.
    2. Calculate how many multiplications and additions are executed for Settings 1-4.
  1. Requirements

Submit the following code files:

1) mini-project-4.sh

2) sinegen.c

3) cascade.c

4) spectrogram.c

5) spectshow.py.

These requirements for the code files are described as follows:

* + 1. Please write a shell script named “**mini-project-4.sh**” to compile and build your C programs, generate the 0.1-second WAVE files, cascade the generated WAVE files, produce the 16 ascii files that store the spectrogram datum, and 16 pdf files that contain figures for the waveforms and spectrograms.
    2. The waveform of each 0.1 seconds must be generated by the program “**sinegen.c**” which you wrote for “Mini Project 3 - Generating Waves.”
    3. To cascade the generated WAVE files, please write a C program named “**cascade.c**” with the following usage:

cascade scp output

scp: the name of a list file that contains the names of the WAVE files to be cascaded in the order

output: the name of the WAVE file that cascades the WAVE files listed in the scp file.

* + 1. To generate the 16 spectrogram, please write a C program “**spectrogram.c**” with the following usage:

spectrogram w\_size w\_type dft\_size f\_itv wav\_in spec\_out

w\_size: analysis window size (unit: millisecond)

w\_type: a string to be “hamming” or “rectangular”

dft\_size: DFT/FFT window size (unit: millisecond)

f\_itv: frame interval (unit: millisecond)

wav\_in: input WAVE file

spec\_out: output spectrigram data (ascii)

* + 1. Please write a program named “**spectshow.py**” to save pdf files that display waveforms and spectrograms according to the files: s-16k.{Set1~Set4}.txt, s-8k.{Set1~Set4}.txt, aeueo-16kHz.{Set1~Set4}.txt, and aeueo-8kHz.{Set1~Set4}.txt. You can use Matplotlib (<https://matplotlib.org/>) library to generate the figures. The usage of “spectshow.py” is

python3 spectshow.py in\_wav in\_txt out\_pdf

in\_wav: input WAVE file

in\_txt: input text file that contains spectrogram data in ascii

out\_pdf: a pdf file that shows the figures of a waveform and the corresponding spectrogram.

The pdf files opened should look like Figs. 3 and 4.

一張含有 螢幕擷取畫面, Rectangle, 樣式 的圖片

自動產生的描述

Fig. 3: The waveform and spectrogram for aeueo-8kHz.wav.

一張含有 螢幕擷取畫面, 電路 的圖片

自動產生的描述

Fig. 4: The waveform and spectrogram for aeueo-16kHz.wav.

References:

1. Xuedong Huang, Alex Acero, Hsiao-Wuen Hon (2001). *Spoken Language Processing: a guide to theory, algorithm, and system development*, page 274-281. Prentice Hall
2. https://www.speech.kth.se/wavesurfer/