Name: Marco Ravelo

Problem Set 1: Speech Signals
The University of Texas at Austin
CS395T: Spoken Language Technologies / Fall 2022
Instructor: David Harwath

---

*work for problems 1 & 2 were done on paper — scanned notebook pages are at the end of this file

1.1 Vowel 1
Compute the first three formants (sorted by increasing frequency) of the acoustic tube conguration shown in Figure 1. (8 points)

$$F1 = 944.44\,Hz$$
$$F2 = 1214.29\,Hz$$
$$F3 = 2833.33\,Hz$$

Based upon the formant values you computed above, list 1 American English vowel that would be a good match for this vocal tract conguration. (2 points)

$/a/\ as\ in\ \text{“}bott\text{”}$

1.2 Vowel 2
Compute the first three formants (sorted by increasing frequency) of the acoustic tube conguration shown in Figure 2. (8 points)

$$F1 = 268.896\,Hz$$
$$F2 = 1888.889\,Hz$$
$$F3 = 3400.000\,Hz$$

Based upon the formant values you computed above, list 1 American English vowel that would be a good match for this vocal tract conguration. (2 points)

$/i^{y}/\ as\ in\ \text{“}beet\text{”}$

2.1 For each of the following types of phones, identify a segment (start time and end time) that contains an instance of that type of phone: (16 points)

$A)\ Fricative$	$0.20\ to\ 0.32\ and\ 0.62\ to\ 0.71$
$B)\ Stop$	$0.95\ to\ 1.05$
$C)\ Semivowel$	$0.05\ to\ 0.21$
$D)\ Nasal$	$0.05\ to\ 0.10$
$E)\ Front\ Vowel$	$0.50\ to\ 0.55$
$F)\ Back\ Vowel$	$0.10\ to\ 0.20$
$G)\ Alveolar\ Cons.$	$0.60\ to\ 0.70$
$H)\ Retroex$	$1.10\ to\ 1.20$

2.2 What is the fundamental frequency of the voiced excitation (F0) at 0.55 seconds? (4 points)

$$F0\ =\ 140\ Hz$$

3.1 Loading the audio file:

code:

```
# 1. load in audio waveform

from scipy.io import wavfile
samplerate, raw_data = wavfile.read('signal.wav')

print ("samplerate type: ", str(type(samplerate)), ", data type: ",
str(type(raw_data)))
print ("sample rate: ", samplerate)
print ("data: ", raw_data)
print ("size: ", len(raw_data))
```

output:

```
> samplerate type:  <class 'int'> , data type:  <class 'numpy.ndarray'>
sample rate:  16000
> data:  [ 174575  449637  316148 ... -315117 -454616 -419864]
> size:  18091
```

What is the sampling rate of the audio, and how many samples are in the recording? How many seconds worth of audio does this correspond to?

$$sample\ rate:\ 16000\ Hz$$
$$length\ of\ audio:\ 18091\ samples/16000\ Hz\ =\ 1.1307\ sec.$$

3.2 Mean subtraction:

code:

```
# 2. DC subtraction (remove the mean of the signal)

mean = numpy.mean(raw_data)
print ("mean: ", mean)

mean_sub_data = raw_data - mean
print ("data: \n" , mean_sub_data)
```

output:

```
> mean:  11756.525786302582
> data:
 [ 162818.4742137  437880.4742137  304391.4742137 ... -326873.5257863
 -466372.5257863 -431620.5257863]
```

3.3 Pre-emphasis:

code:

```
# 3. pre-emphasis filtering

def filter_data(data, b):
    filt_data = numpy.zeros(len(data))
    for n in range(len(data)):
        if n == 0:
            filt_data[n] = data[n]
        else:
            filt_data[n] = data[n] - (b * data[n - 1])
    return filt_data

b = 0.97
my_filt_data = filter_data(mean_sub_data, b)
print ("computed data: ", my_filt_data)
```

output:

```
> computed data:  [ 162818.4742137   279946.55422641 -120352.58577359 ...
-137258.38577359
 -149305.20577359   20760.82422641]
```

3.4 Computing frames:

code:

```
# 4. transforming the audio into a sequence of frames

from random import sample

def compute_frames(x, L, S): # signal x, window length L, window shift S
    signal_size = len(x)
    frames_size = int(numpy.ceil(signal_size / S) - 2)
    frames = numpy.zeros([frames_size, L], dtype = float)

    for k in range(frames_size):
        for n in range(L):
            index = (k * S) + n
            # set frame value
            if (index < signal_size):
                frames[k, n] = x[index]
            # pad with zeros
            else:
                frames[k, n] = 0.000

    return frames

length = int(0.025 * samplerate)
shift = int(0.01 * samplerate)
print ("length: ", length, " shift: ", shift)

my_frames = compute_frames(my_filt_data, length, shift)
print ("computed frames: \n", my_frames)
```

output:

```
> length:  400  shift:  160
> computed frames:
 [[ 162818.4742137   279946.55422641 -120352.58577359 ...   74655.24422641
   -17433.40577359   55738.71422641] ...
 [ -16179.16577359 -229856.06577359   75007.57422641 ...       0.
        0.              0.        ]]
```

3.5 For each frame:

code:

```
# 5. for each frame ...

from scipy import signal
from scipy import fft
import matplotlib.pyplot as plot

mel_filters = numpy.load("mel_filters.npy")

def compute_C_array(frame, N, C_array_size):
    # apply the window function
    window = signal.hamming(len(frame))
    windowed_frame = frame * window
    # compute the fourier transform
    X_m = fft.fft(windowed_frame, N)
    # compute magnitude
    X_m = numpy.abs(X_m)
    # square each element to get the power spectra
    X_m = X_m * X_m
    # apply mel-filter
    X_m_half = X_m[0:mel_filters.shape[1]]
    X_mel = numpy.matmul(X_m_half, mel_filters.T)
    # take the log
    X_log_mel = numpy.zeros(X_mel.shape)
    for k in range(len(X_log_mel)):
        X_log_mel[k] = numpy.maximum(-50.0, numpy.log(X_mel[k]))
    # compute the DCT and "liftering"
    C_array = numpy.zeros(C_array_size, dtype = float)
    for i in range(C_array_size):
        sum = 0.0
        for k in range(23):
            sum += X_log_mel[k] * numpy.cos(((numpy.pi * i) / 23) * (k + 0.5))
        C_array[i] = sum
    return C_array

N = 512
C = 13
MFCC_REP = numpy.zeros([len(my_frames), C], dtype = float)
for n in range(MFCC_REP.shape[0]):
    MFCC_REP[n] = compute_C_array(my_frames[n], N, C)

print ("MFCC_REP shape: ", MFCC_REP.shape)
print ("MFCC_REP: ", MFCC_REP)
```

output:

```
MFCC_REP shape:  (112, 13)
MFCC_REP:  [[ 5.39121421e+02 -1.88310069e+01  1.10734460e+00 ...
3.42458454e+00 3.04125377e+00 -9.07008924e-01] ...
 [ 6.41720883e+02 -3.18541353e+01 -4.99451925e+00 ...  8.15009283e-01
  -5.71254995e-01  2.26657194e+00]]
```

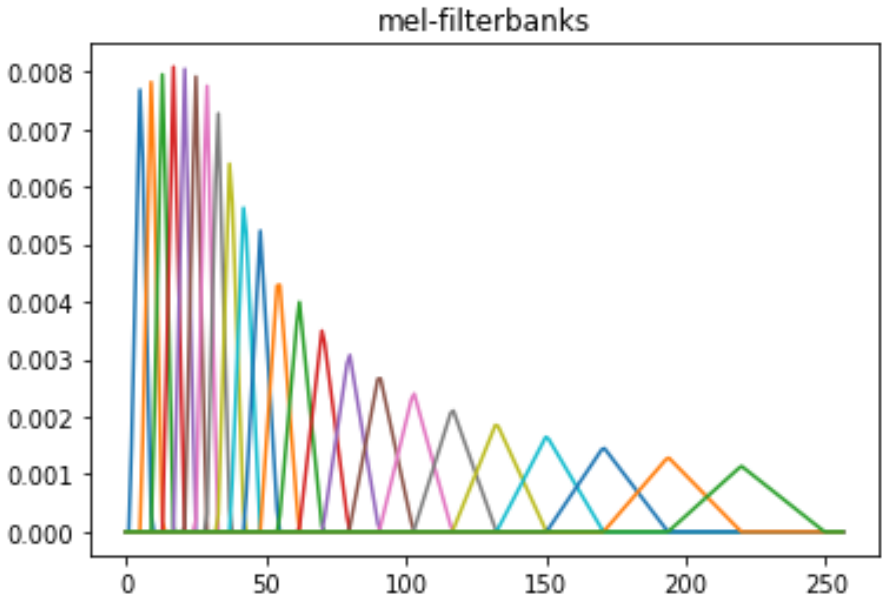
3.6 Mel-lterbank application:

code:

```
# mel filter plot
import matplotlib.pyplot as plot

mel_filters = numpy.load("mel_filters.npy")
print ("mel_filters.shape: ", mel_filters.shape)

plot.plot(mel_filters.T)
plot.title("mel-filterbanks")
plot.show()
```

output:

```
> mel_filters.shape:  (23, 257)
```



3.7 Putting it all together:

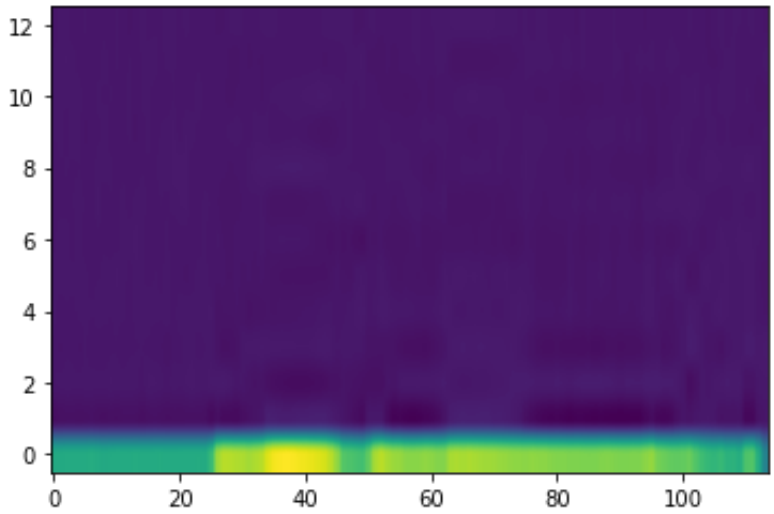
code:

```
# all together now!

def compute_MFCC(wav, b, win_length, win_shift, N, C):
    # 1. load in .wav file
    samplerate, raw_data = wavfile.read(wav)
    # 2. DC subtraction (remove the mean of the signal)
    mean = numpy.mean(raw_data)
    mean_sub_data = raw_data - mean
    # 3. pre-emphasis filtering
    filt_data = filter_data(mean_sub_data, b)
    # 4. transform audio into a sequence of frames
    length = int(win_length * samplerate)
    shift = int(win_shift * samplerate)
    my_frames = compute_frames(filt_data, length, shift)
    # 5. calculate C array for each frame
    MFCC_REP = numpy.zeros([len(my_frames), C])
    for n in range(MFCC_REP.shape[0]):
        MFCC_REP[n] = compute_C_array(my_frames[n], N, C)
    return MFCC_REP


MFCC_REP = compute_MFCC('signal.wav', 0.97, 0.025, 0.01, 512, 13)
print ("MFCC_REP shape: ", MFCC_REP.shape)
print ("MFCC_REP: ", MFCC_REP)

plot.imshow(MFCC_REP.T, origin = 'lower', aspect = 'auto')
```

output:

```
> MFCC_REP shape:  (112, 13)
> MFCC_REP:  [[ 5.39121421e+02 -1.88310069e+01  1.10734460e+00 ...
3.42458454e+00 3.04125377e+00 -9.07008924e-01] ...
 [ 6.41720883e+02 -3.18541353e+01 -4.99451925e+00 ...  8.15009283e-01
  -5.71254995e-01  2.26657194e+00]]
```

Visualize your MFCC features for only C1 through C12, because the dynamic range of C0 is much larger than the rest of the coecients using imshow() and include the plot in your writeup.



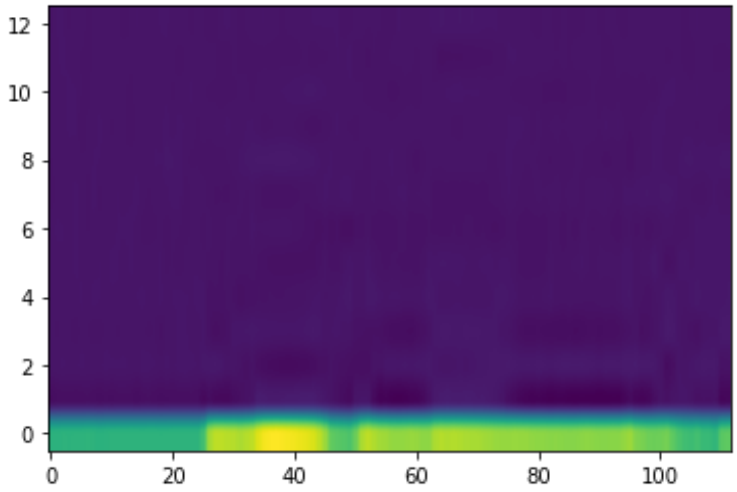
Also compute the difference between your MFCCs and the reference MFCCs (all 13 coecients). What is the Mean Squared Error (MSE) between the two matrices?

code:

```
# get reference mfcc
ref_MFCC = numpy.load("reference_mfcc.npy")
print ("ref_MFCC.shape: ", ref_MFCC.shape)

plot.imshow(ref_MFCC.T, origin = 'lower', aspect = 'auto')
```

output:

```
> ref_MFCC.shape:  (112, 13)
```



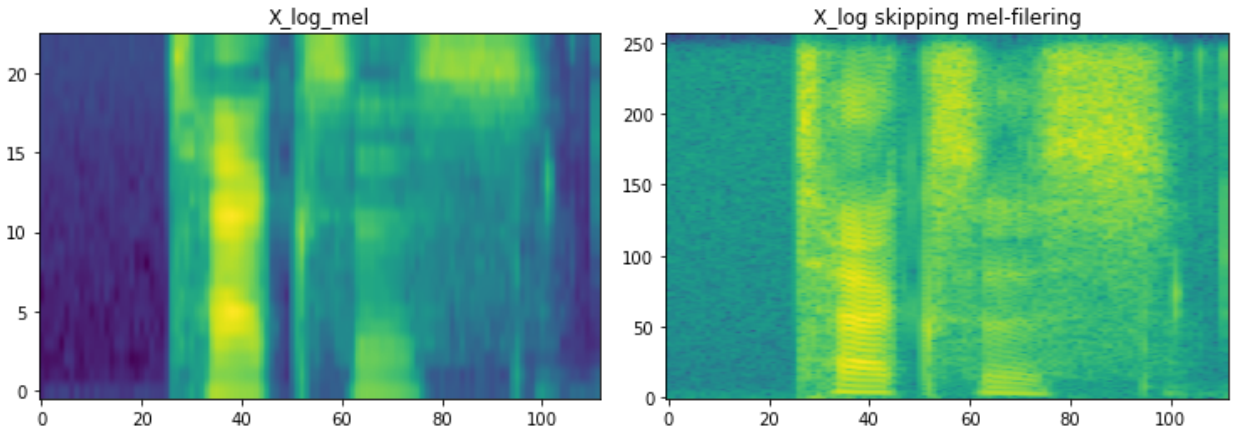
code:

```
# Mean Squared Error (MSE) between the two matrices?

A = ref_MFCC.T
B = MFCC_REP.T

mse = (numpy.square(A - B)).mean(axis=None)
print ("Mean Square Error: ", mse)
```

output:

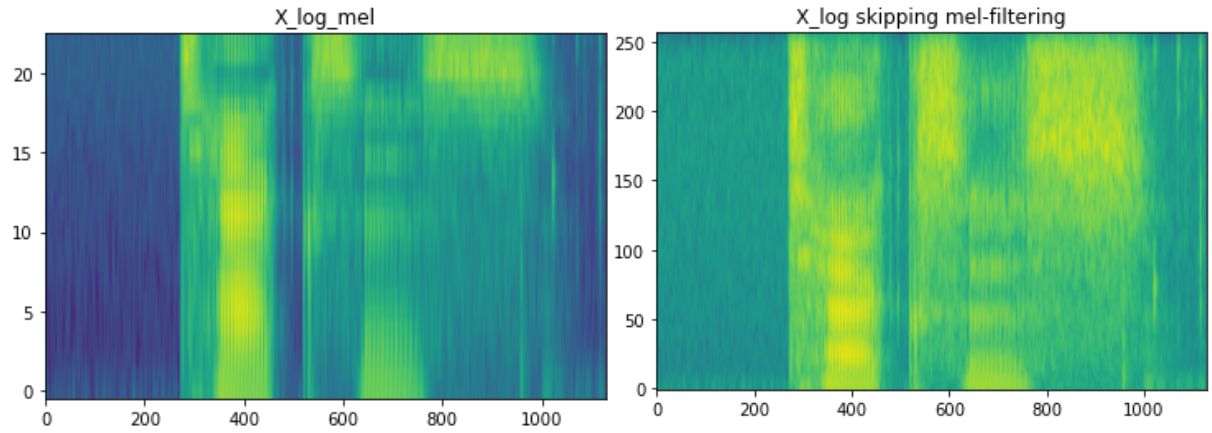
```
> Mean Square Error:  2.0893358167628796e-26
```

Next, plot the MFSCs X_logmel for the entire utterance (i.e. omit the DCT and liftering steps) again using imshow(). Alongside it, plot the log power spectrum which corresponds to skipping the application of the Mel-filterbanks. Comment on how these two spectra look dierent, and include both plots in your writeup.



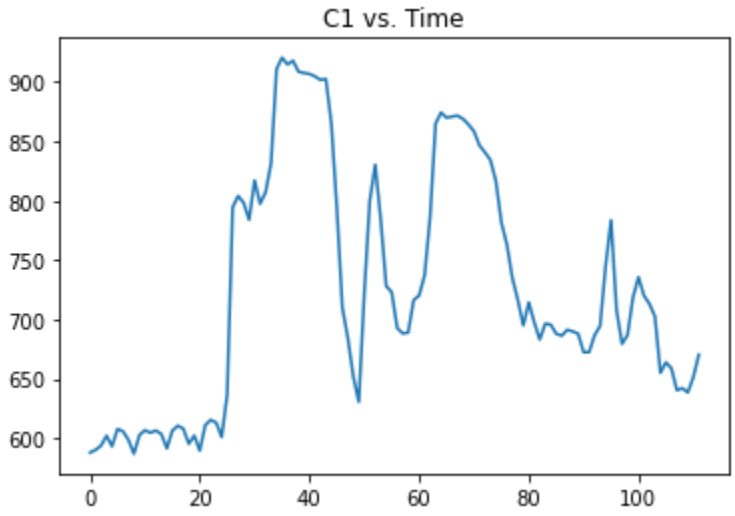
One can clearly see than when we skip the mel-filtering step, the spectrogram becomes significantly noisier and more difficult to read.

Next, change your frame extraction conguration to use a window length L of 4 milliseconds, and a window shift S of 1 millisecond. Using this conguration, plot X_logmel and max(−50, log(X_pow[m])) alongside one another again. How does the spectrum look dierent when using 25 millisecond windows as opposed to 4 millisecond windows? Why does this happen? Include both of these plots in your writeup.



When we change the length and shift of the frame extraction to be smaller, the spectrogram's time resolution (x-axis) become clearer while the frequency resolution (y-axis) becomes blurier. This happens because the shorter window blurs together the harmonics (blurs the y-axis) while giving the time axis more points to plot (visualy looks sharper along the x-axis).

Finally, make a plot of C1 as a function of time and include this plot in your writeup. What do you notice about the value that C1 takes on during a vowel vs. during a fricative? Why is this the case?



We can see from the graph that C1 changes dramatically depending on the type of phone that is being spoken. During vowels, C1 tends to spike:

- between t=25 and t=45 is the "e" in "texas"
- between t=60 and t=75 is the "a" in "texas"

During fricatives, C1 dips down much lower:

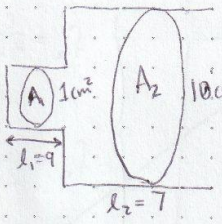
- Between t=50 and t=60 is the "x" in "texas"
- Between t=80 and t=120 is the "s" in "texas"

This happens because vowels have a much more discernible pitch while fricatives are mostly just noise.

*scanned notebook pages

PROBLEM SET 1 : SPEECH SIGNALS

1.1



$= \square + \square$ (1/4 WL, 1/4 WL)

$c = 34{,}000$ cm/sec

$f_n = \frac{c}{4l_1}(2n-1)$ $\quad f_n = \frac{c}{4l_2}(2n-1)$ $\quad$ for $n = 1, 2, 3$

$f_n = \frac{34{,}000}{4(9)}(2n-1)$

$= 944.44(2n-1)$

$n=1 \rightarrow 944.44\ s^{-1}$
$n=2 \rightarrow$ ~~1888.889~~ 2833.332
$n=3 \rightarrow$ ~~2833.333~~ 4722.22

$f_n = \frac{34{,}000}{4(7)}(2n-1)$

$= 1214.286(2n-1)$

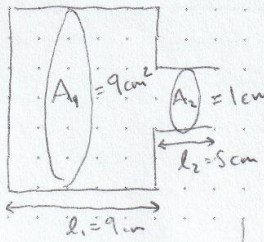
$n=1 \rightarrow 1214.286$
$n=2 \rightarrow 3642.858$
$n=3 \rightarrow 6071.43$

$\therefore$

$F_1 = 944.44$ Hz
$F_2 = 1214.286$ Hz
$F_3 = 2833.33$ Hz

American English vowel:
/a/ as in "bott"

1.2



$=$ Helmholtz + 1/2 WL + 1/2 WL

$f = \frac{c}{2\pi}\sqrt{\frac{A_2}{A_1 l_1 l_2}}$ $\quad f_n = \frac{c}{2l_1}n$ $\quad f_n = \frac{c}{2l_2}n$

$f_n = \frac{34{,}000}{2\pi}\sqrt{\frac{1}{9(9)(5)}}$

$= (5411.428)(0.04961)$

$= 268.896$ ★

$f_n = \frac{34{,}000}{2(9)}n$

$= 1888.889n$

$n=1 \rightarrow 1888.889$ ★
$n=2 \rightarrow 3777.778$
$n=3 \rightarrow 5666.667$

$f_n = \frac{34{,}000}{2(5)}$

$= 3400$

$n=1 \rightarrow 3400$ ★
$n=2 \rightarrow 6{,}400$
$n=3 \rightarrow 10{,}200$

$\therefore$

$F_1 = 268.896$ Hz
$F_2 = 1888.889$ Hz
$F_3 = 3400$ Hz

American English vowel:
/iy/ as in "beet"

2.1 A) Fricative 0.2 → 0.32 and 0.62 → 0.71

B) Stop 0.95 → 1.0

C) Semivowel 0.05 → 0.2

D) Nasal 0.05 → 0.1

E) Front Vowel 0.5 → 0.55

F) Back Vowel 0.1 → 0.2

G) Alveolar Consonant 0.6 → 0.7

H) Retroflex 1.1 → 1.2

2.2 What is the fundamental frequency @ 0.55 sec?

pitch periods between 0.5 and 0.6 = 14

$\frac{14 \text{ pp}}{0.1 \text{ sec}} = 140 \text{ /sec}$ ∴ $F_0 = 140 \text{ Hz}$