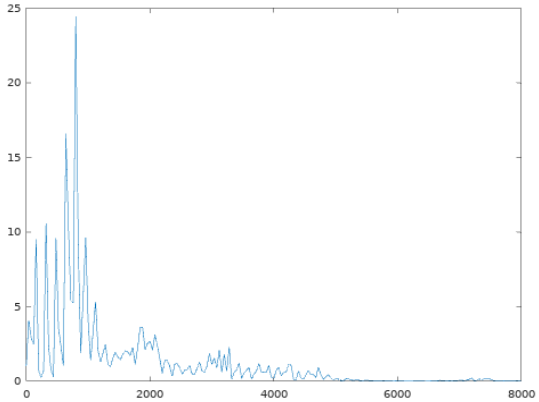
Pisit Wajanasara 5931042721
Auto Speech Recognition

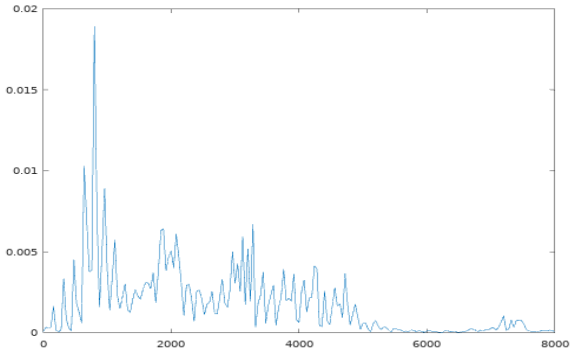
Homework 4 Gowajee

MFCC

1. 1453 ms
2. We use mean of data as our DC offset. The DC offset is 0.000031490
3. We can calculate the frame length as follow
   $frame\ length = 25ms \times \frac{16000}{1000ms} = 400$
   We can calculate the shift length as follow
   $shift\ length = 10ms \times \frac{16000}{1000ms} = 160$
   If we have n frame and 23248 elements of data, we can calculate the value n as follows
   $160(n-1) + 400 = 23248$
   $n = 144\ frames$
4. using Matlab code the calculated logE for frame 50 is 0.83246
5. Before the pre-emphasis

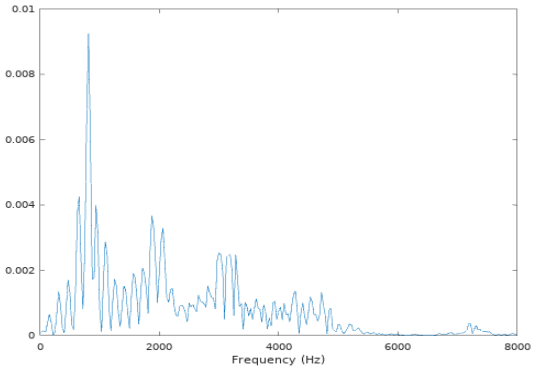


   After the pre-emphasis



   This is considered high pass filter as you can see that the height of the low frequency area are amplified down by the filter.

6. Spectrum[1] = 0 and Spectrum[257] = 8000 as we can calculate spectrum[i] from Fs*(i-1) / N
   Magnitude spectrum at frame 50 are as follow.



7. Minimum frequency is 156.25, Maximum frequency is 7156.2
   Energy with in each band are 00.34833, 0.48773, 0.69450, 0.90630, 2.23125, 4.82668, 6.47116, 3.15728, 2.18790, 2.14554, 2.90877, 5.23196, 4.08475, 2.40343, 3.84652, 5.79413, 3.21661, 2.97349, 3.23975, 1.46081, 0.34961, 0.16198, 0.72817 respectively (We can calculate from transpose(mel_filters) x spectrum of the frame 50 in previous question).

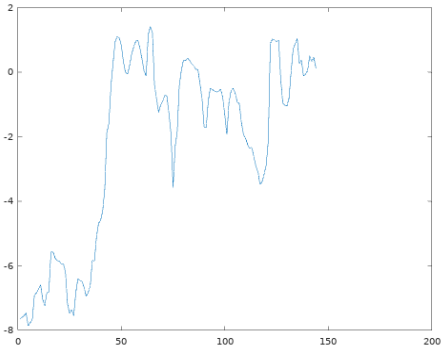
9.

```
function result = compute_mfcc(w)

shift_size = 160;
frame_size = 400;
frame_size_2n = 2**(ceil(log2(frame_size)));

[y,Fs,bits] = wavread(w);
len = length(y);
dc_offset = mean(y);
y = y - dc_offset;
total_frames = ceil(1 + ((len - frame_size) / shift_size));
% padding
y = [y.' zeros(1,(total_frames*frame_size) - len)].';
spe_global = y;
for i = 2:length(y)
  spe_global(i) = y(i) - 0.97*y(i-1);
endfor

for frame = 1:total_frames
  start_frame = shift_size*(frame - 1) + 1;
  stop_frame = start_frame + frame_size - 1;
  sof = y(start_frame:stop_frame);
  logE = max(-50, log(sum(sof.^2)));

  spe = spe_global(start_frame:stop_frame);

  spe_p = spe .* hamming(length(spe));
  temp = fft(spe_p, frame_size_2n);
  spect = abs(fft(spe_p, frame_size_2n));

  load('MFCC/mel_filters.mat');
  [v, args] = max(mel_filters);
  spect_mel = (mel_filters.' * spect(1:257))';
  mfsc = max(-50, log(spect_mel));
```

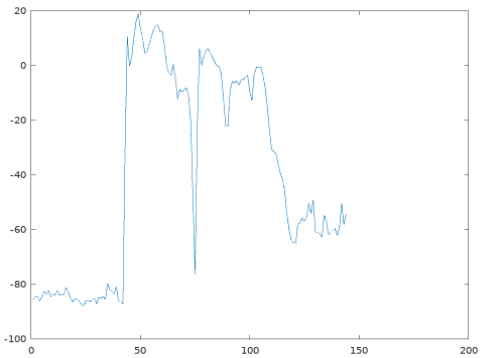
```
 cs = 1:23;
 for i = 1:23
   s = 0;
   for j = 1:23
     s += mfsc(j) * cos((j - 0.5) * (pi*(i - 1)/23));
   end
   cs(i) = s;
 endfor

 x = [[logE] cs(1:13)];

 if (frame == 1)
   result = x.';
 else
   result = horzcat(result, x.');
 endif

endfor

endfunction
```

10. The logE is similar to C0 from the observation in both plotted graph by Octave. Moreover, if we observe the equations for C0, we can see that the cos term will always be 1 making its similar to the logE equation. From the observation of C1 of the word “gas station”, we can observe that when the fricative occurred the C1 will be low and when the vowel occurred the C1 will be high. Moreover, if we derive he equation of C1 we will find that it is a linear combination of a sine like wave which may be correspond to both glottal and noise source.

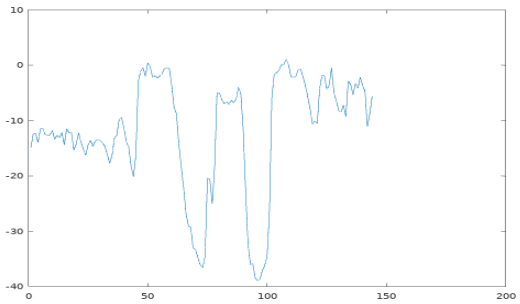
This is logE graph



This is C0 graph



This is C1 graph



11. a) 23248x1 b) 257x1 c) 23x1 d) 23x1

CMN

1. The euclidean distance is 15269.3437494399 (This is value is not a square rooted value)
2. This euclidean distance is 8151.47119875423. It is smaller than the previous question.

DTW Theory

1. Using the following python code we can find the answer.

```
template = [int(x) for x in input().split()]
test = [int(x) for x in input().split()]

dp = [[0]*len(test) for i in range(len(template))]

for i in range(len(template)):
    for j in range(len(test)):
        dist = int(abs(template[i] - test[j]))
        p1 = p2 = p3 = 2e9
        if i == 0 and j == 0:
            p1 = p2 = p3 = dist
        if i-1 >= 0 and j-1 >= 0:
            p1 = dp[i-1][j-1] + dist
        if i-2 >= 0 and j-1 >= 0:
            p2 = dp[i-2][j-1] + 2*dist
        if i-1 >= 0 and j-2 >= 0:
            p3 = dp[i-1][j-2] + dist
        dp[i][j] = min(p1, p2, p3)

for i in range(len(template)):
    for j in range(len(test)):
        if dp[i][j] >= 2e9:
            print('X', end=' ')
        else:
            print(dp[i][j], end=' ')
    print()

print(dp[len(template)-1][len(test)-1])
```

Ans: (1, 1) - 1, (2, 3) - 2, (3, 5) - 2, (5, 6) - 2, (7, 7) - 6, (9, 8) - 6

2. Use the following python code with constraint supported

```
template = [int(x) for x in input().split()]
test = [int(x) for x in input().split()]
constraint = int(input())

dp = [[0]*len(test) for i in range(len(template))]
```

```
for i in range(len(template)):
    for j in range(len(test)):
        if i - j >= constraint or j - i >= constraint:
            dp[i][j] = 2e9
            continue
        dist = int(abs(template[i] - test[j]))
        p1 = p2 = p3 = 2e9
        if i == 0 and j == 0:
            p1 = p2 = p3 = dist
        if i-1 >= 0 and j-1 >= 0:
            p1 = dp[i-1][j-1] + dist
        if i-2 >= 0 and j-1 >= 0:
            p2 = dp[i-2][j-1] + 2*dist
        if i-1 >= 0 and j-2 >= 0:
            p3 = dp[i-1][j-2] + dist
        dp[i][j] = min(p1, p2, p3)

for i in range(len(template)):
    for j in range(len(test)):
        if dp[i][j] >= 2e9:
            print('X', end=' ')
        else:
            print(dp[i][j], end=' ')
    print()

print(dp[len(template)-1][len(test)-1])
```

Ans: (1, 1) - 1, (2, 3) - 2, (3, 5) - 2, (5, 6) - 2, (7, 7) - 6, (9, 8) - 6

3. Using the following python code we can calculate the result.

```
from math import sqrt

template = [(1,-1), (3,3), (8,7), (4,-5), (5,4)]
test = [(2,0), (2,4), (4,2), (4,-4), (6,3)]

def cal_dist(x, y):
    return sqrt((x[0] - y[0])**2 + (x[1] - y[1])**2)

dp = [[0]*len(test) for i in range(len(template))]

for i in range(len(template)):
    for j in range(len(test)):
        dist = cal_dist(template[i], test[j])
        p1 = p2 = p3 = 2e9
        if i == 0 and j == 0:
            p1 = p2 = p3 = dist
        if i-1 >= 0 and j-1 >= 0:
            p1 = dp[i-1][j-1] + dist
        if i-2 >= 0 and j-1 >= 0:
            p2 = dp[i-2][j-1] + 2*dist
        if i-1 >= 0 and j-2 >= 0:
            p3 = dp[i-1][j-2] + dist
```

```
        dp[i][j] = min(p1, p2, p3)

for i in range(len(template)):
    for j in range(len(test)):
        if dp[i][j] >= 2e9:
            print('X', end=' ')
        else:
            print(dp[i][j], end=' ')
    print()

print(dp[len(template)-1][len(test)-1])
```

Ans: (1, 1) – 1.414, (2, 3) - 2.828, (4, 4) - 4.828, (5, 5) - 6.243

4. Breath first search as we continue to span the calculation result using dynamic programming with bottom-up approach.

DTW

4. The false alarm rate is 8.2258

5. The false alarm rate is 94.032. If we consider the AUC for both RoC curve directly, the one with cmn will perform better. On the other hand, if we answer from no CMN case oppositely, we could get the better performance compared to with CMN one because of a larger AUC value.