EE 679: Assignment 3

Kalpesh V. Patil (130040019)

October 10, 2016

Q.1 Reconstruct sounds from LP parameters

We obtain LP coefficients by specifying filter order in Levinson Durbin algorithm. Filter order was derived using the follwing heuristics.

$$\#formants \simeq \frac{F_s/2}{avg(F_{n+1} - F_n)}$$

$$p = 2\#formants + 2$$

Order was taken to be 10 for /a/, /i/ and /n/ and 18 for /s/. Sound was reconstructed from LP parameters by implementing linear difference equation of the given order. De-emphasis filter was applied for voiced sounds, which were pre-emphasized earlier.

Results and Discussion: .

- After listening to the sounds generated by LP filter, we can state that it is able to reconstruct vowel sounds (/a/ and /i/) quite better than others. Vowel sounds are all pole in nature and hence can be modelled effectively by LP filter.
- But nasal sounds (/n/) and fricatives contain zeros as well, hence they are difficult to model by LP filter. This is observed from the sounds
- Voicing can observed in /a/, /i/ and /n/ by looking at the periodic output, which is not present in /s/
- Synthesized /n/ sound contains essence of /i/ as well because it was extracted from the word "pani" and /n/ is immediately followed by /i/, which somewhat results in transitioning effects.

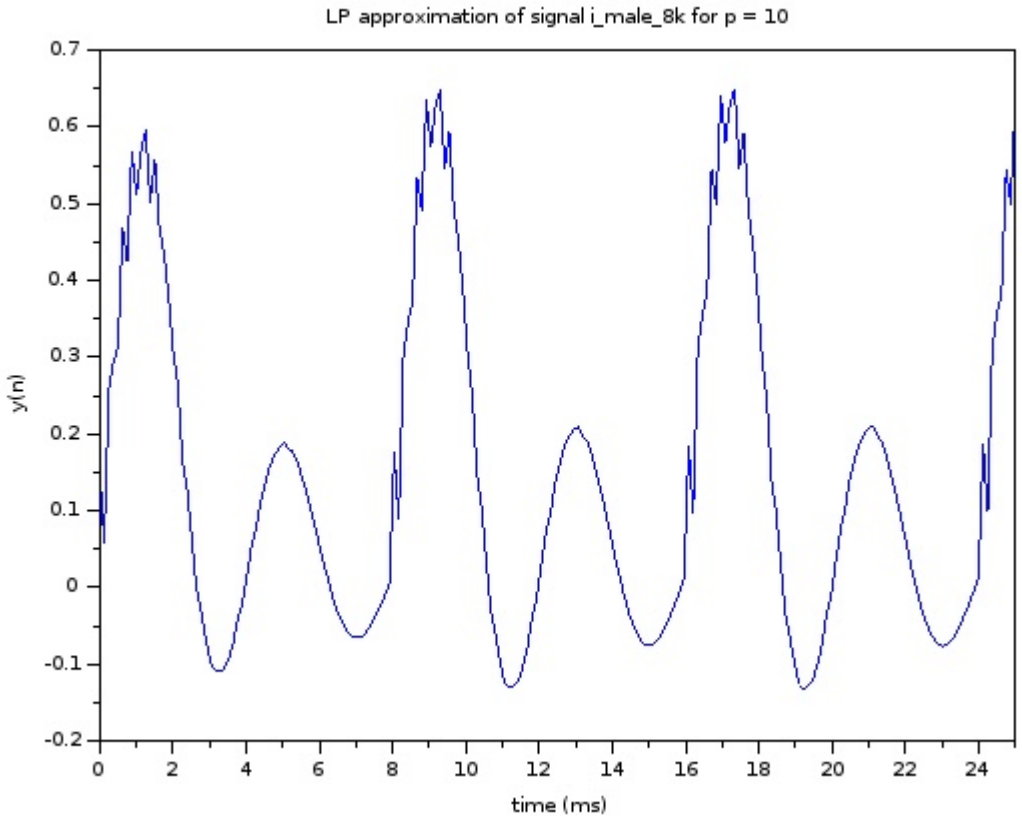

Figure 1: LP approximation of signal i_male_8k for p = 10

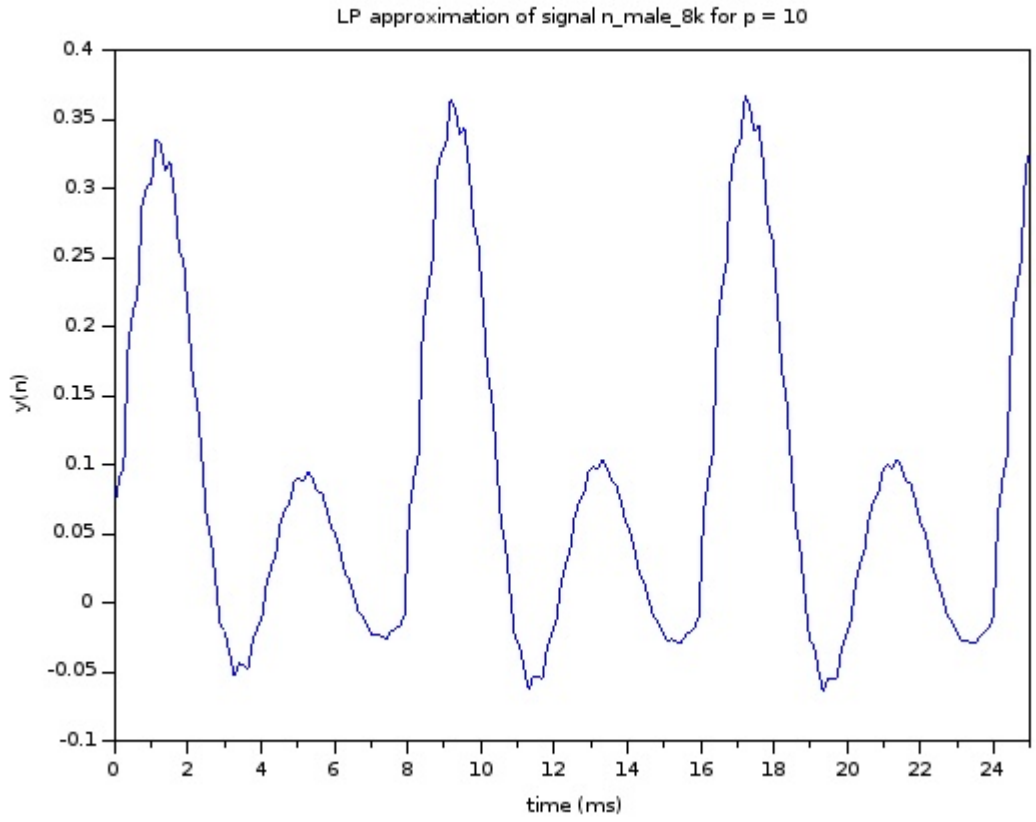

Figure 2: LP approximation of signal n_male_8k for p = 10

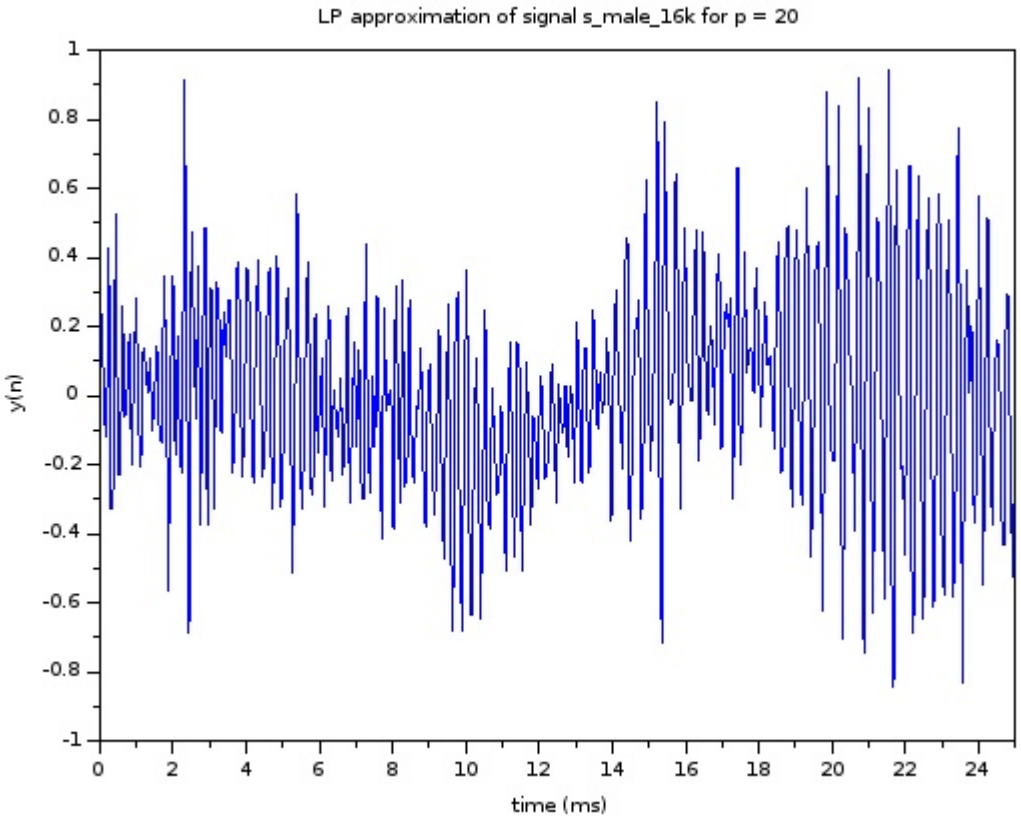

Figure 3: LP approximation of signal s_male_16k for p = 20

Q.2 Spectral envelope via cepstral liftering for LP synthetic sounds

/a/ vowel was synthetically reconstructed by passing an impluse train from LP filter of order 10 whose coefficients were obtained earlier. Real cepstrum was obtained from generated synthetic signal. Spectral envelope is computed using cepstral liftering.

Results and Discussion: .

From the figure below it can be seen cepstral liftering produces appropriate peaks at desired locations for the reconstructed vowel /a/. We can further compare LP envelope and cepstral envelope for given sound do match upto certain extent. The peaks in the LP envelope are sharper compared to that of cepstral envelope.

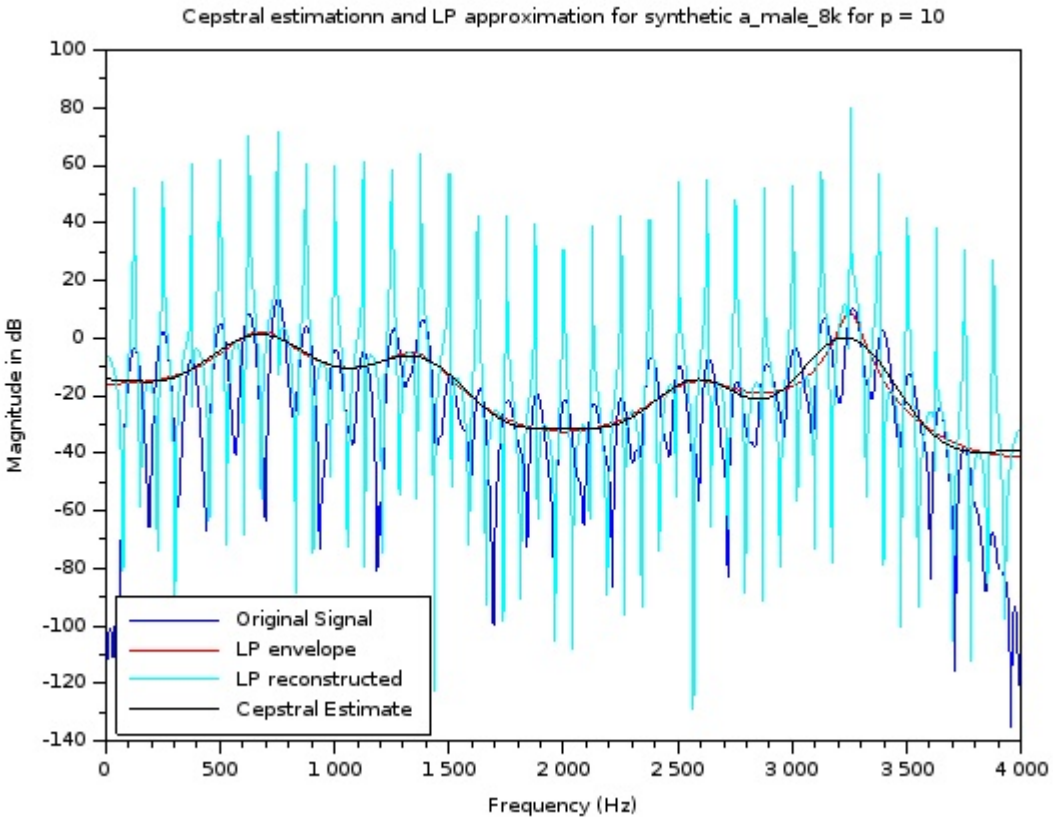

Figure 4: Cepstral estimationn and LP approximation for synthetic a_male_8k for p = 10

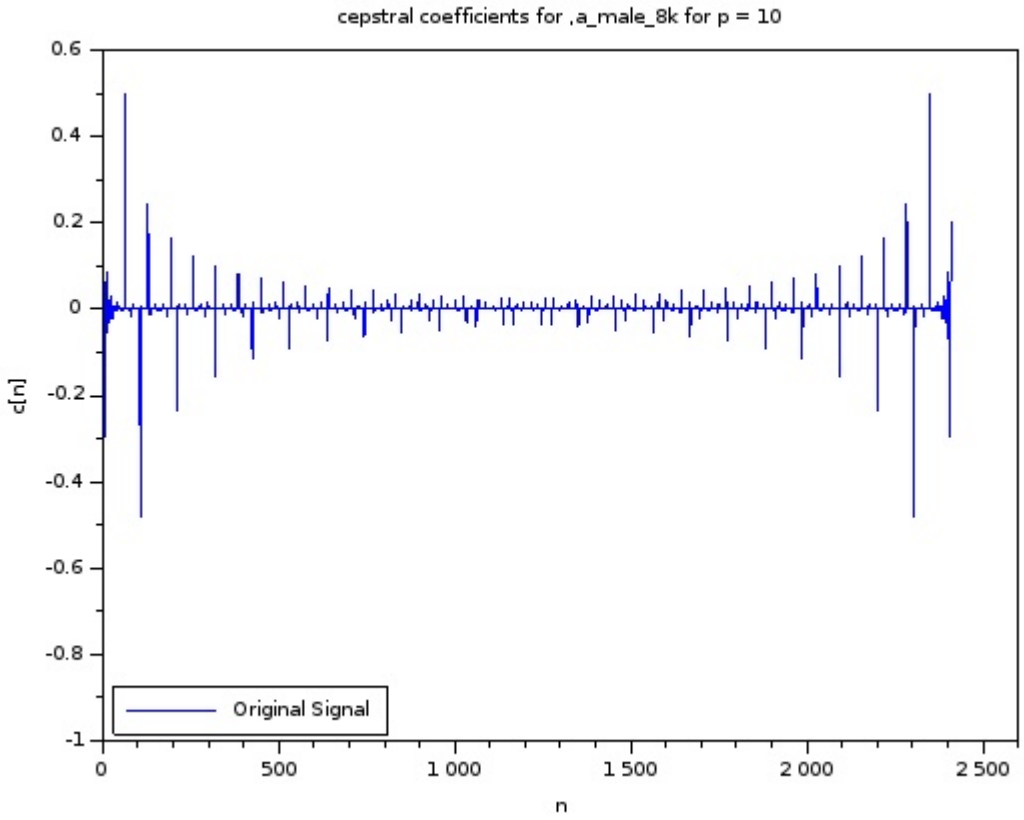

Figure 5: cepstral coefficients for ,a_male_8k for p = 10

Q.3 Cepstral analyses of natural sounds

Real cepstrum for each sound was obtained. Spectral envelope was computed using cepstral liftering. We can compare LP envelope and cepstral envelope by superposing them on actual magnitude spectrum of the windowed signal. Further pitch can be computed by observing repeating pattern in peaks in cepstral plot.

Results and Discussion:

- Poles are sharper and higher in case of LP envelpe compared to cepstral envelope.
- LP envelope is not able to model zeros in the transfer function effectively due to its inherent nature of being an all pole filter. But cepstral envelope has no such limitations and hence it does model zeos effectively. This is prominent in case of /n/ and /s/.
- Cepstral envelope follows signal in a better way compared to LP envelope.
- Cepstral envelope is generally below the LP envelope.
- **Pitch estimation:** Pitch of the sound can be computed by looking at the periodicity in the cepstral coefficients plot against quefrency. The observed pitch is found to be around 64 samples, which corresponds to 125 Hz (8 ms). We obtained the same number in pitch estimation using autocorrelation function in assignment 2. Note that /s/ being unvoiced in nature isn't excited by glottal vibrations and hence there isn't any pitch information and thus repeating peaks in cepstral coefficients of /s/.

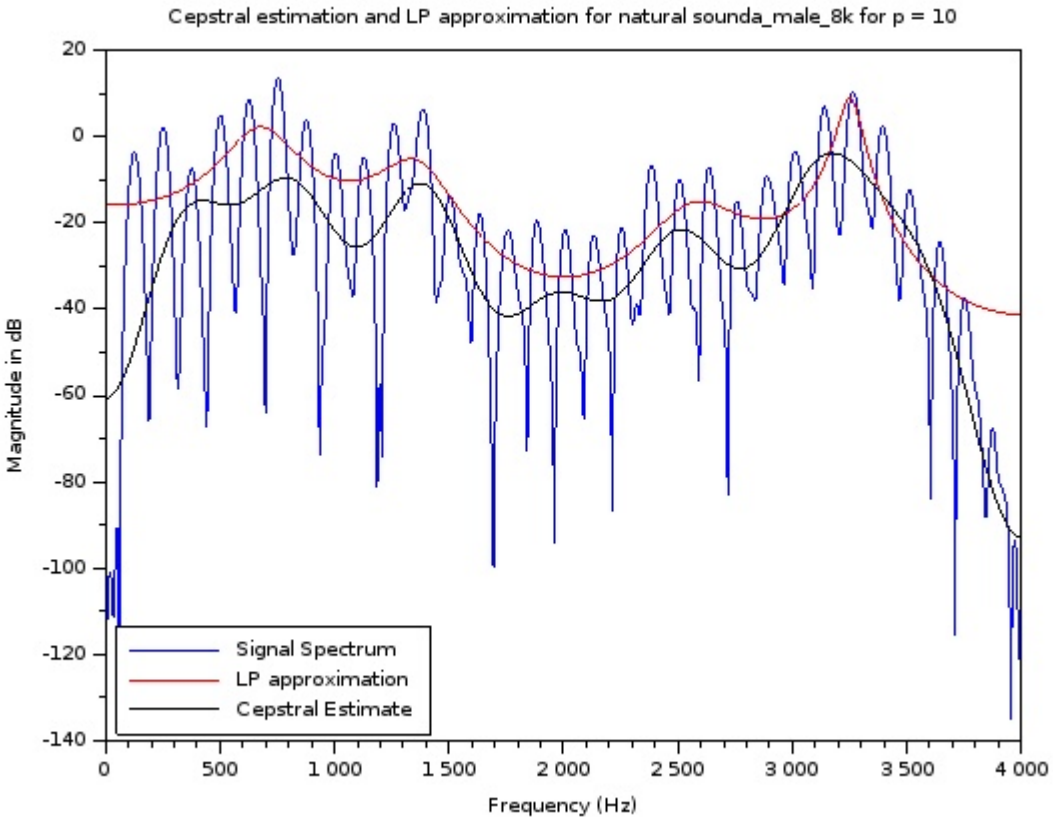

Figure 6: Cepstral estimation and LP approximation for natural sounda_male_8k for p = 10

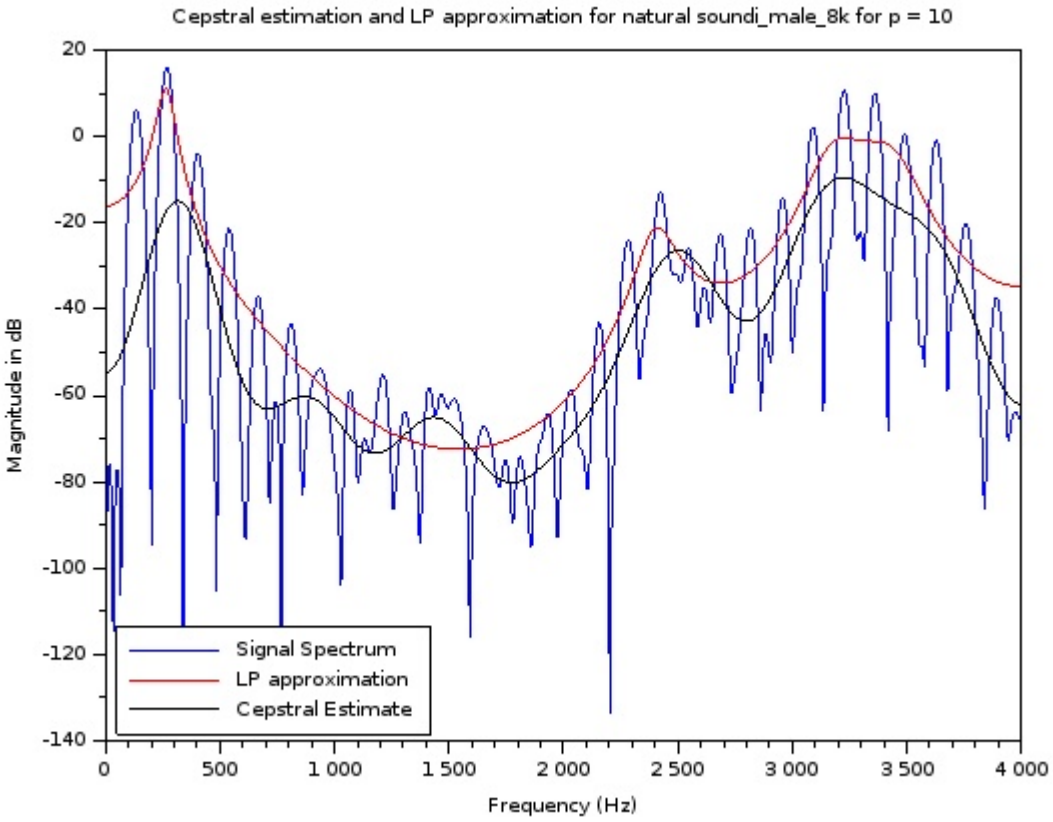

Figure 7: Cepstral estimation and LP approximation for natural soundi_male_8k for p = 10

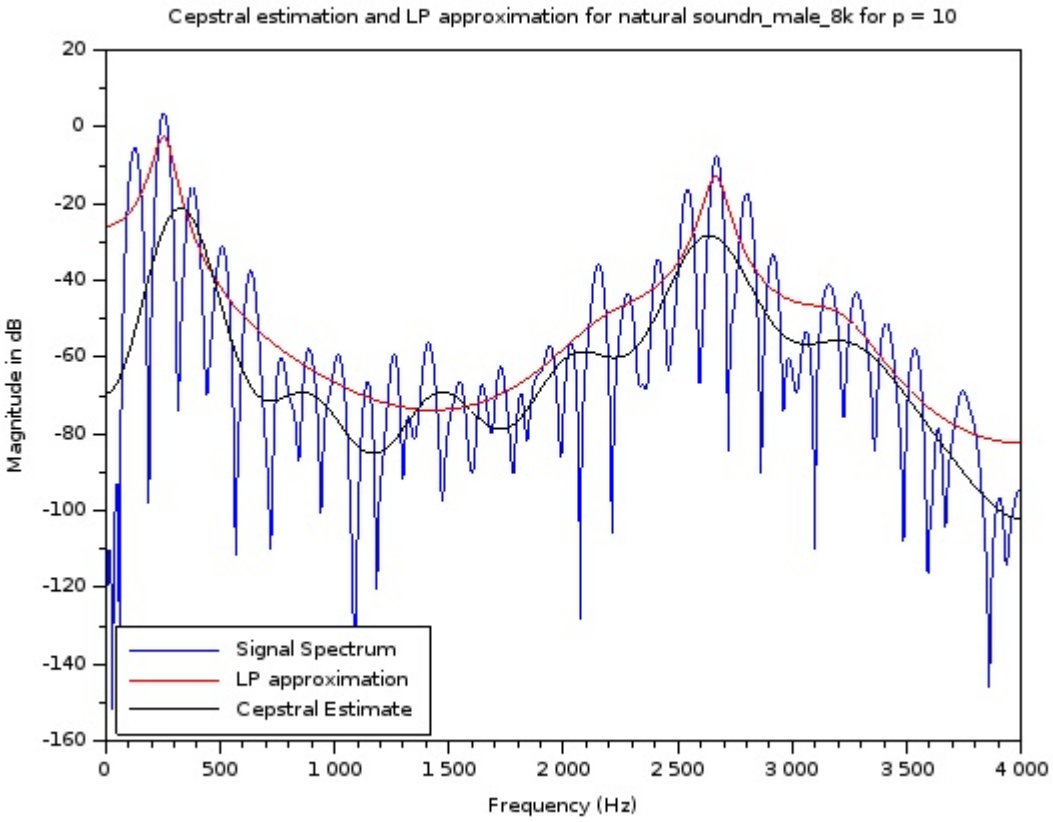

Figure 8: Cepstral estimation and LP approximation for natural soundn_male_8k for p = 10

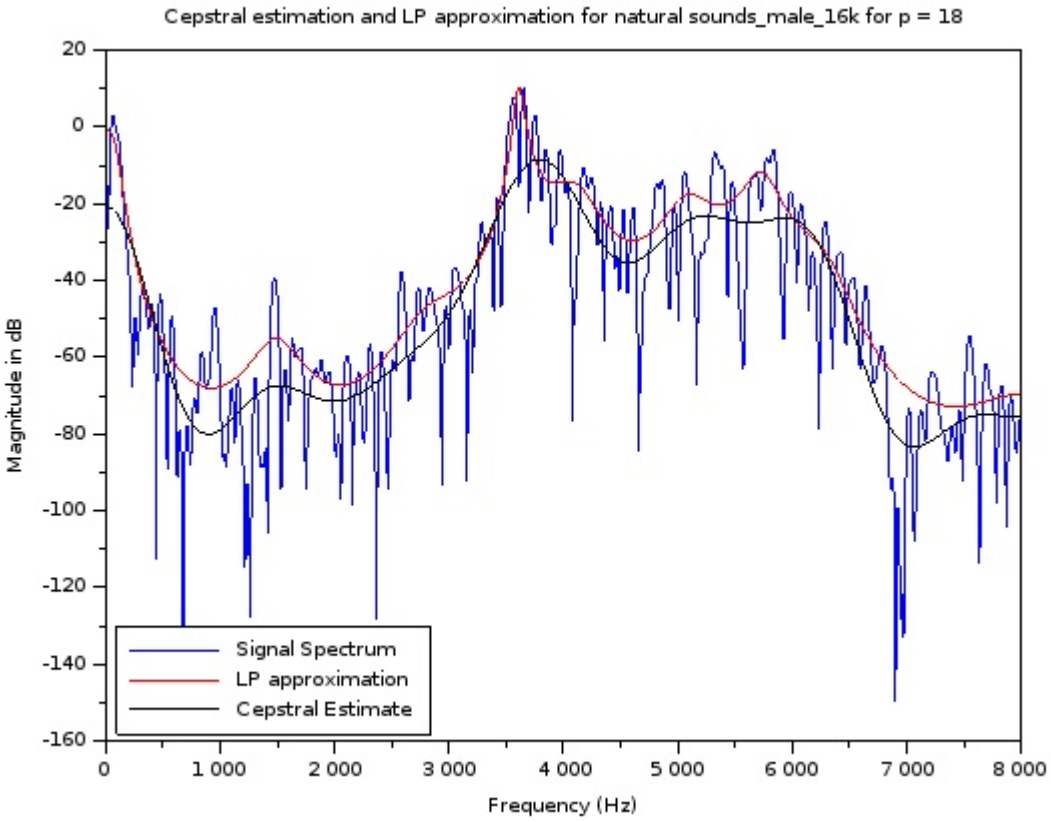

Figure 9: Cepstral estimation and LP approximation for natural sounds_male_16k for p = 18

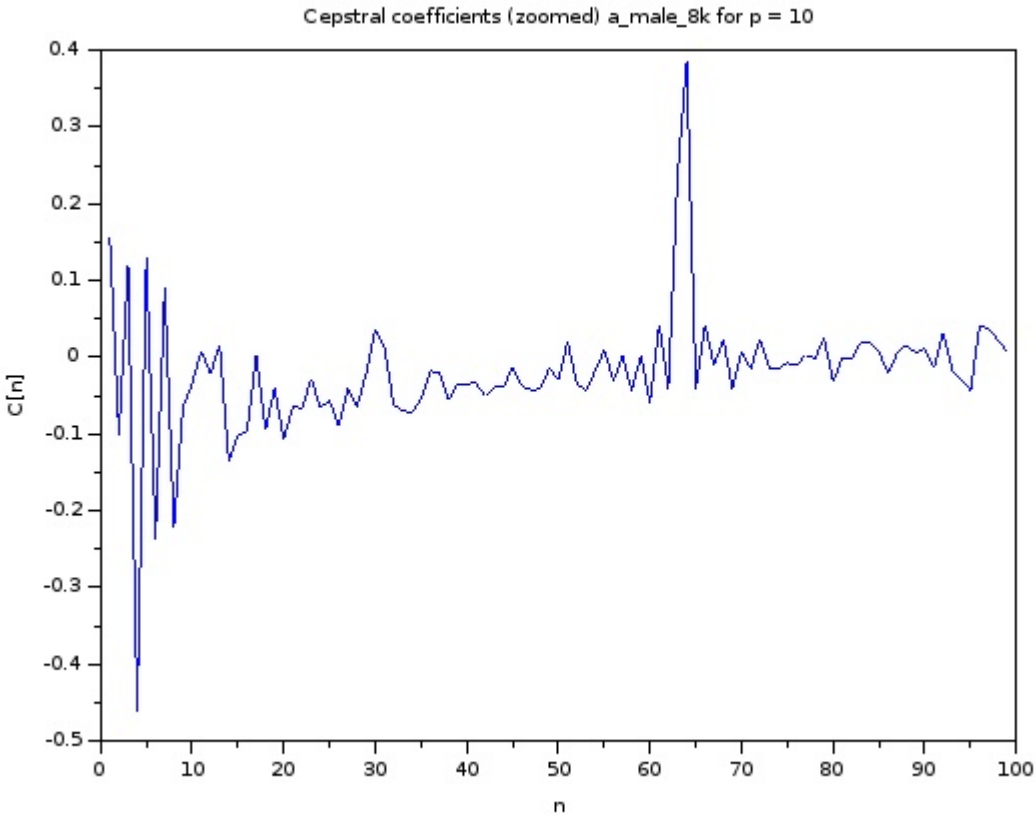

Figure 10: Cepstral coefficients (zoomed) a_male_8k for p = 10

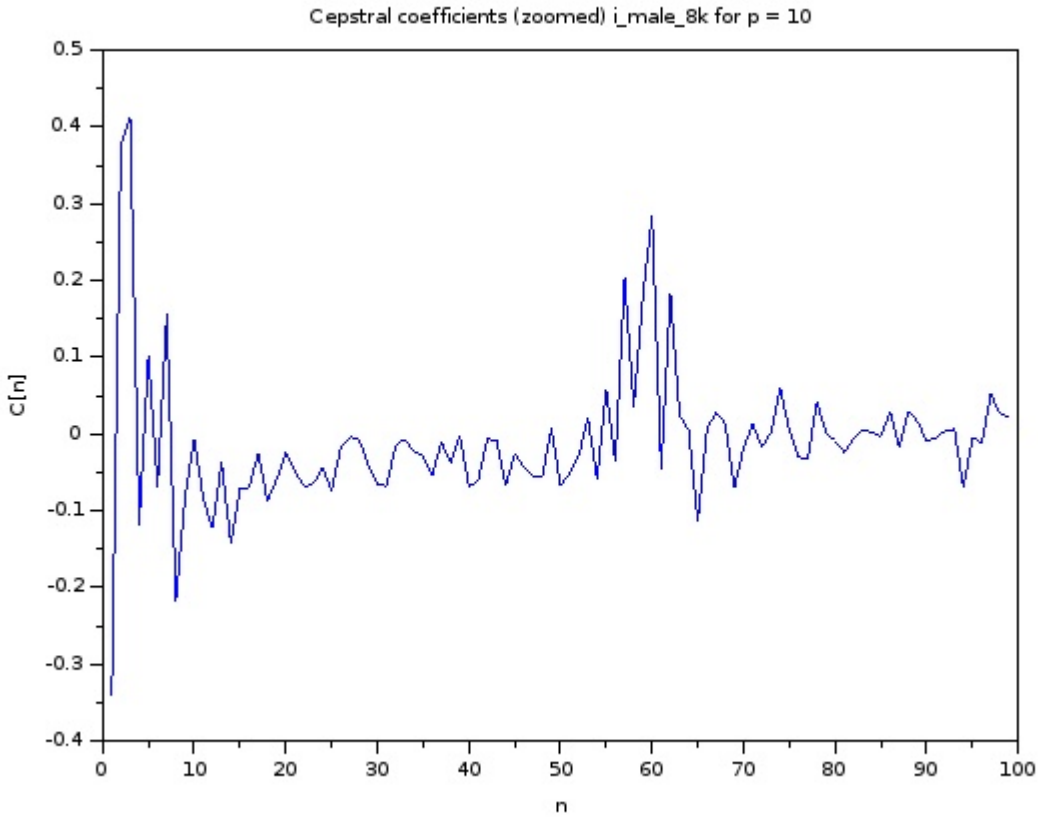

Figure 11: Cepstral coefficients (zoomed) i_male_8k for p = 10

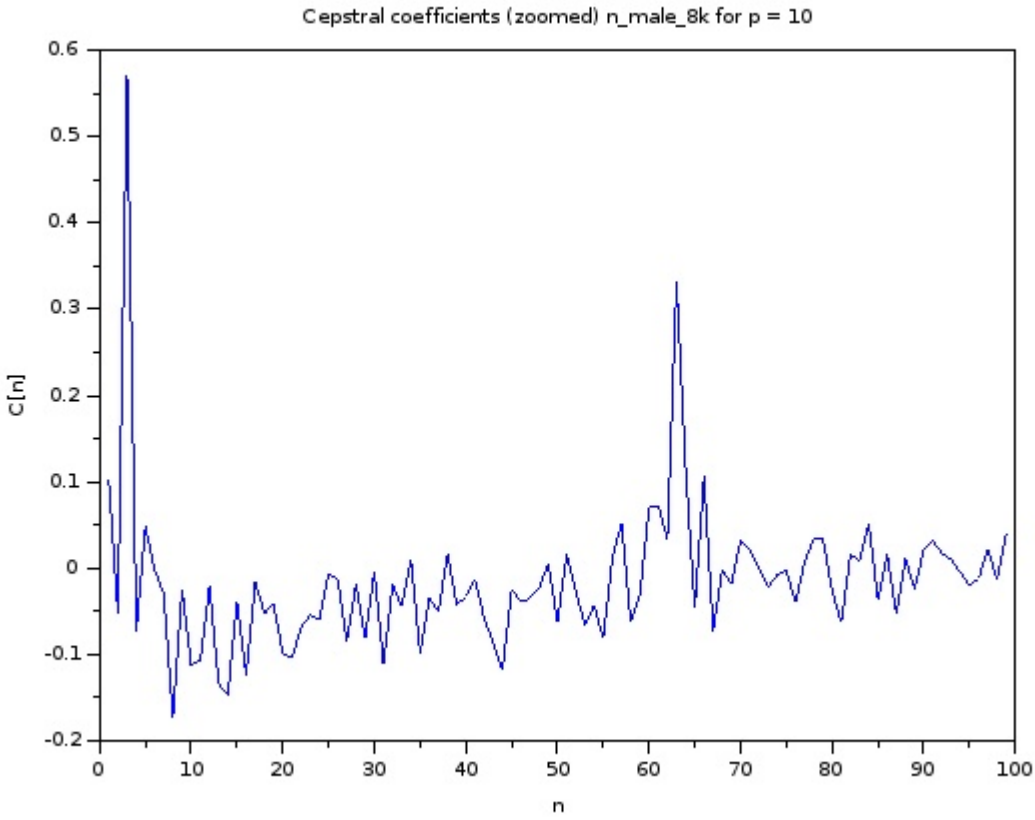

Figure 12: Cepstral coefficients (zoomed) n_male_8k for p = 10

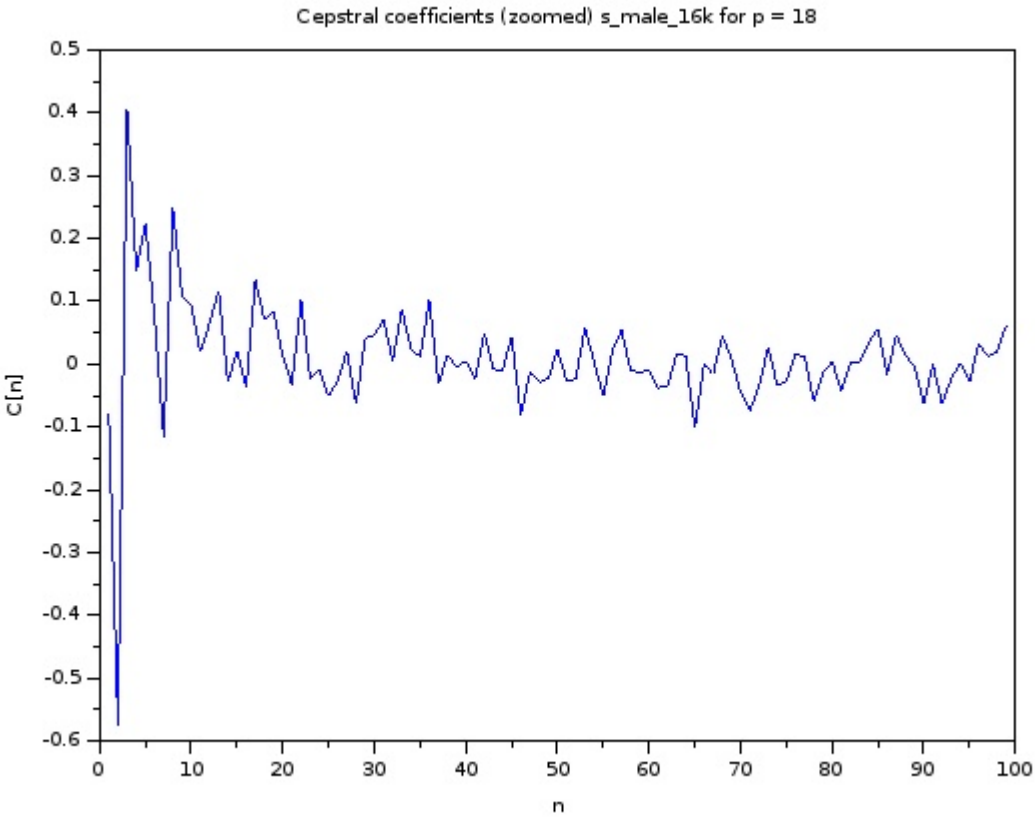

Figure 13: Cepstral coefficients (zoomed) s_male_16k for p = 18

Appendix

Major functions used in codes

```
//pre_emphasis filter
function y = pre_emphasis_filter(x,alpha)
        y = zeros(length(x))
        y(1) = x(1)
        for i = 2:length(x)
                y(i) = x(i)-alpha*x(i-1)
        end
endfunction

//de_emphasis filter
function y = de_emphasis_filter(x,alpha)
        y = zeros(length(x))
        y(1) = alpha*x(1)
        for i = 2:length(x)
                y(i) = alpha*y(i-1) + x(i)
        end
endfunction



//function to find output of LP aproximation
function y=find_output_of_LP(num, den, input_type, Fs, F0, t_duration)
        n_samples = t_duration*Fs + 1
        x = zeros(n_samples,1)
        //impulse train generation
        if(input_type == "voiced")
            for i = 2:t_duration*F0
                x(floor((i-1)*Fs/F0))= 1
            end
        else
            x = rand(x,'normal')
        end
        y = zeros(n_samples,1)
        y = time_response(x,num,den)

        if(input_type == 'voiced')
            y = de_emphasis_filter(y,0.95)
        end
endfunction
```

```
// function implementing Levinson Durbin algorithm
function [num, den]=find_LP_coefs_using_levinson(y, p_max)
        r = zeros(length(y),1)
        for ki=0:length(x)
             for ni=ki:N-1
                r(ki+1)=r(ki+1)+y(ni+1)*y(ni-ki+1);
             end
        end
        //LP recursion
        e_vec = [r(1)]
        i = 1
        k = r(2)/e_vec(1)
        a_vec = [1,k]
        e_vec = [e_vec,(1-(k)^2)*e_vec(1)]
        for i = 2:p_max
            k = r(i+1)
            for j = 1:i-1
                k = k - a_vec(j+1)*r(i-j+1)
            end
            k = k/e_vec(i)
            a_vec_old = a_vec
            a_new = k
            for j = 1:i-1
                a_vec(j+1) = a_vec_old(j+1)-k*a_vec_old(i-j+1)
            end
            a_vec = [a_vec , a_new]
            e_vec(i+1) = (1-k^2)*e_vec(i)
        end
        num = sqrt(e_vec(i+1))
        den = -a_vec(2:length(a_vec))
        den = [1,den]
endfunction


// function to find ceptral coefficients
function ceptral_coefs=find_ceptral_coefs(x, n_fft)
    N = length(x)
    //zero padding
    x_padded = zeros(n_fft,1)
    x_padded(1:length(x)) = x'
    //finding fft
    freq_x = fftshift(fft(x_padded))
    //X_mag = abs(freq_x(n_fft/2 +1:n_fft))
    X_mag = (abs(freq_x))
    log_mag = log(X_mag)
    log_mag_padded = zeros(n_fft,1)
    log_mag_padded(1:length(log_mag)) = log_mag
   // ceptral_coefs = ifft(log_mag_padded)
   ceptral_coefs = ifft(log(abs(fft(x_padded))))
endfunction
```