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
```