Linguistics 582
Basics of Digital Signal Processing

Formant Analysis using LPC

LPC (linear predictive coefficients) analysis is a technique for estimating the vocal tract transfer function, from which its poles, he formant frequencies, can be analytically calculated. Informally, the logic of the technique is as follows: As the air in the vocal tract resonates in response to a source impulse, sample values of the speech signal will be correlated with the values of that same signal delayed (lagged) by some number of samples. Which lags exhibit the highest correlations will depend on the particula frequencies in the signal. For example, if there were a resonance at the Nyquist frequency, then the signal would be highly negatively correlated with itself at lag=1. The correlation structure can be made precise using the standard statistical technique of multiple linear regression.

Mutiple linear regression. Suppose you make multiple observations on a number of variables and you want to discover the optimal prediction of one of those variables (the **dependent** variable) from the others (the **independent** variables). For example, suppose you kept good notes on your coffee preferences, and for each cup of morning coffee, you have a subjective rating of how well you like it, as well as various objective measures of that coffee--number of days since roasting, number of minutes since grinding, number of cups brewed at once, temperature of the water, source of the beans, etc. You could now use multiple linear regression to find the optimal equation that will predict your preference ratings (the dependent variable) from the objective measures (the independent variables). The form of the equation that multiple regression determines is as follows (where X_i are dependent variable vectors and , Y is the dependent variable vector, and b are coefficients determined by analysis procedure):

(1) $Y = k + b_1 X_1 + b_2 X_2 + b_3 X_3 + \ldots + b_M X_M + Res$

K is a constant term. Res stands for Residual which is the error--the difference between the predicted value given by the rest of the equation and the actual value of Y. The optimal prediction will be the one in which the sum of squared values of Res across all the observations is as small as possible.

It is possible to perform such a regression analysis on the sample values of a chunk of a

speech waveform (the length of the chunk is what is called the window size of the analysis). The analysis will find how to best predict the value of a given sample of waveform within the window from the values of the previous M samples. The equation will be of the following form, where Y is the signal, and **a** are the coefficients returned by the analysis (we ignore the constant term in speech, since its value determines only a dc offset of the signal from 0):

(2) `Y(k) = a(1)*Y(k-1) + a(2)*Y(k-2) + ... + a(M)Y(k-M) + Res`

What is the form of the Res likely to be? If M is small compared to the source periodicity (1/f0) of the speech, the Res will include those properties of the waveform that are predictable only from source periodicity (excitation of the vocal tract filter), but not from vocal tract resonance properties. In an ideal case in which the source is a Impulse train, the residual will be that Impulse train. (Non-ideal cases will work similarly, but properties of the source function that make it different from an impulse train will be folded into the estimate of the vocal tract transfer function). We will view the Residual as an excitation function in what follows (E = Res).

By re-arranging (2), it is possible to re-conceptualize it as a (moving average, feedforward) filter that takes the speech signal as input and generates the excitation as output:

(3) `E = 1*Y(k) - a(1)*Y(k-1) - a(2)*Y(k-2) - ... - a(M)*Y(k-M)`

This filter is called the **inverse filter**. It is a filter that takes speech as input and gives an excitation as output. Its transfer function (A(z)) has the coefficients of the equation in (3) as coefficients of successive powers of z in the numerator (and 1 in the denominator):

(4) $A(z) = 1 - a(1)\, z^{-1} - a(2)\, z^{-2} - \ldots - a(M)\, z^{-(k-M)}$

Now we can represent the filtering action of the inverse filter as multiplication by the z polynomial:

(5) $E = A(z) * Y$

But we can rearrange (5) as (6):

(6) $Y = (1/A(z))*E$

That is, the transfer function of a filter that will generate the speech waveform from an

excitation function is equal to 1/A. That is just the same set of coefficients of z as in (4), but in the denominator, instead of the numerator. This is called the **forward filter**, and is an approximation to the vocal tract transfer function, under certain assumptions. We can evaluate (1/A) at various frequencies, and plot the amptitude response of the vocal tract filter. (See below). We can also solve the polynomial for its roots, thereby finding the frequency of the poles. These will generally correspond to the formant frequencies.

Limitations and assumptions

As shown in (6) the transfer function of the vocal tract that is assumed has 1 in the numerator, so vocal tract transfer function is assumed to consist only of poles, no zeroes. This is unrealistic for nasals and laterals (at least). But it is possible to analyze those by increasing the value of M.

M needs to be determined in advance, and should correspond to the expected order of the vocal tract filter. Start by estimating the number of formants you expect in the frequency range from 0 Hz to the Nyquist frequency. For a 10 KHz sampling rate, the number of formants would be 5 for a large vocal tract, 4 for a smaller vocal tract. Each formant has two poles (complex conjugates of one another), so M must be equal to at least 2 times the number of formants. To the extent that the source function is something other than an impulse, its characteristics are folded into the vocal tract transfer function, and can be described by an additional resonator, thus adding two more to M. Sometimes it is wise to add another pair of points to M to account for aspects of the transfer function that are not well modeled by an all-pole formulation.

Gain. An estimate of the amplitude of the voiced excitation can be calculated from the error signal. The gain is equal to the square root of the sum of the squared error (Residual):

```
G = sqrt (sum(Res.^2));
```

In the ideal case, the only error are the impulses, and thus, the Gain would be determined only by the amplitude of the impulses.

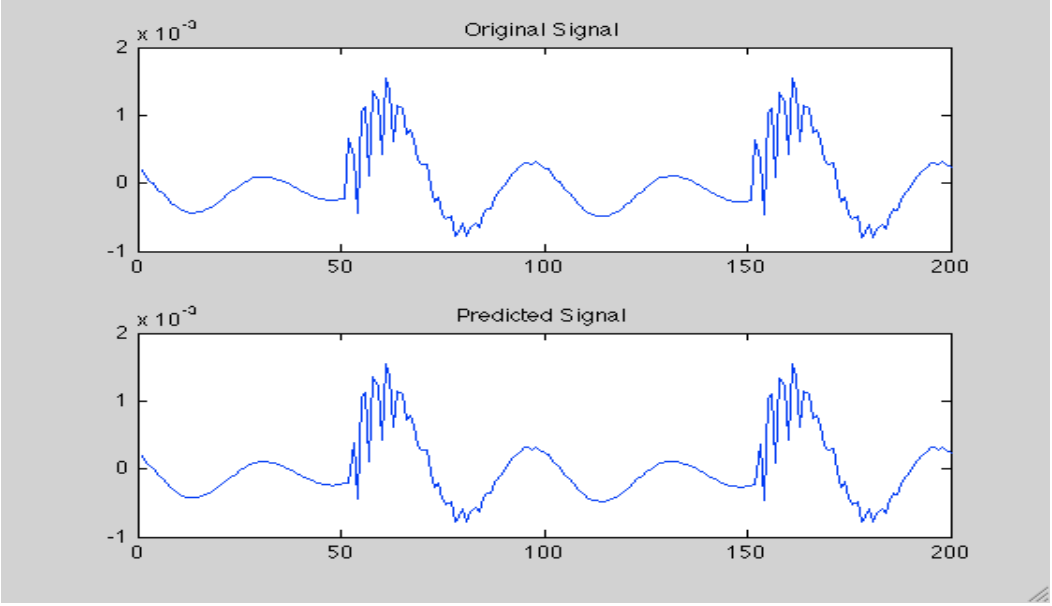
The code below uses an actual multiple regression algorithm to solve for the a coefficients. There are special computational tricks that can be used in this auto-regressive situation that make it much faster than using a general purpose regression algorithm. The point of the code here is to help you to understand the principles behind the analysis.

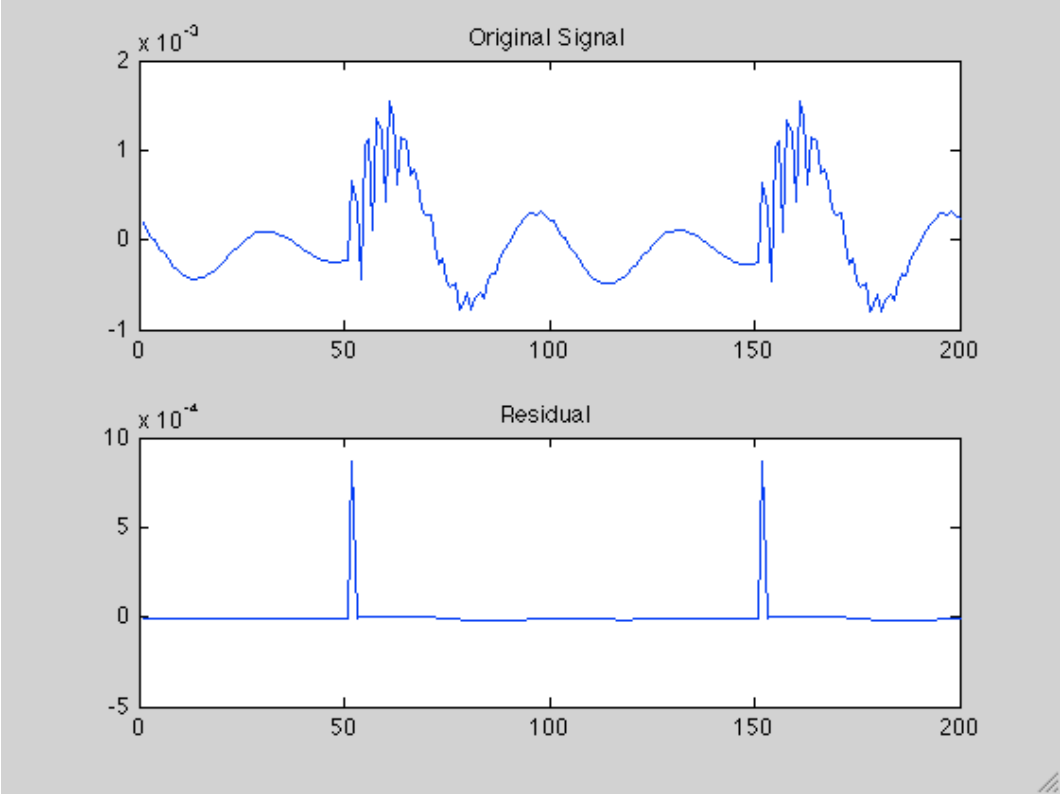
```
function [acoef, Res] = lpc_demo (signal, srate, winsize, M, ibeg)

% LPC Analysis
% Louis Goldstein
% 29 October 1992
%
% input arguments:
% signal    signal to be analyzed
% srate     sampling rate in Hz
% winsize   window size in no.of samples
% M         LPC filter order
% ibeg      first sample in signal to be analyzed
%
% output arguments:
% acoef     LPC coefficients
% Res       Residual from LPC analysis


% iend is last sample in window
iend = ibeg+winsize-1;

% Y is a vector of winsize data points from signal
% Y is the dependent variable for the regression
Y = signal(ibeg:iend)';

% matrix X contains the independent variables for the regression.
% It contains M columns, each of which contains
% elements of signal (y) delayed progressively by one sample

for i=1:M
    X(:,i) = signal(ibeg-i:iend-i)';
end

% perform the regression
% the coefficients are the weights that are applied to each
% of the M columns in order to predict Y.
% Since the columns are delayed versions of Y,
% the weights represent the best prediction of Y from the
% previous M sample points.

[acoef, Res] = regress(X,Y);

% The predicted signal is given by subtracting the residual
% from Y
Pred = Y - Res;

% Plot original window and predicted
subplot (211), plot (signal(ibeg:iend))
title ('Original Signal')
subplot (212), plot (Pred)
title ('Predicted Signal')
subplot
pause

% Plot original window and residual
subplot (211), plot (signal(ibeg:iend))
```

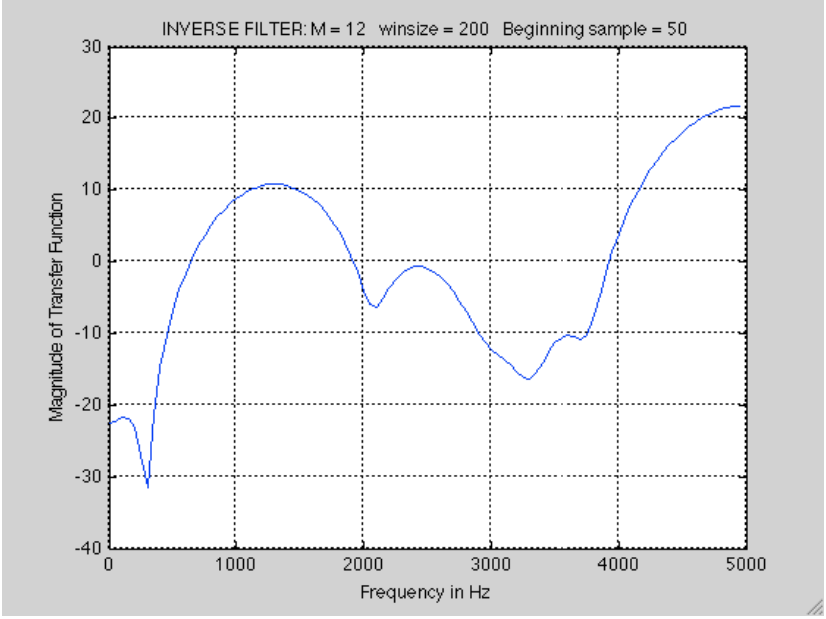
```
title ('Original Signal')
subplot (212), plot (Res)
title ('Residual')
subplot
pause

% Get rid of the M+1st term returned by the regression.
% It just represents the mean (DC level of the signal).
acoef(M+1) = [ ];

% The returned coefficients (acoef) can be used to predict
% Y(k) from preceding M samples:
% E1: Y(k) = a(1)*Y(k-1) + a(2)*Y(k-2) + ... + a(M)Y(k-M) + Res
% We want what is called the INVERSE filter, which will
% take Y as input and output the Res signal.
% That is, it will take all the structure out of the signal
% and give us an inpulse sequence. This can be obtained from
% equation E1 by moving Y(k) and Res to the other side of E1:
% E2: Res = 1*Y(k) - a(1)*Y(k-1) - a(2)*Y(k-2) - ... - a(M)*Y(k-M)
% Thus, the first coefficient of the inverse filter is 1;
% The rest are the negatives of the coefficients returned
% by the regression.
% Note -acoef' is used to convert from a column to row vector:
acoef = [1 -acoef'];

% Plot magnitude of transfer function
% for the INVERSE FILTER
% freqz will return value of the transfer function, h,
% for a given filter numerator and denominator
% at npoints along the frequency scale.
% The frequency of each point (in rad/sample) is returned in w.
num = acoef;
den = 1;
npoints = 100;
[h, w] = freqz(num,den,npoints);
plot (w*srate./(2*pi), log(abs(h)))
xlabel ('Frequency in Hz')
ylabel (' Magnitude of Transfer Function')
title (['INVERSE FILTER: M = ', num2str(M), '   winsize = ',
num2str(winsize),'   Beginning sample = ', num2str(ibeg)])
pause

% Plot magnitude of transfer function if the coefficients are
% used as coefficients of the FORWARD filter (in denominator).
% freqz will return value of the transfer function, h,
% for a given filter numerator and denominator
% at npoint along the frequency scale.
% The frequency of each point (in rad/sample) is returned in w.
num = 1;
den = acoef;
npoints = 100;
[h, w] = freqz(num,acoef,npoints);
plot (w*srate./(2*pi), log(abs(h)))
xlabel ('Frequency in Hz')
ylabel ('Log Magnitude of Transfer Function')
title (['FORWARD FILTER: M = ', num2str(M), '   winsize = ',
num2str(winsize),'   Beginning sample = ', num2str(ibeg)])
```

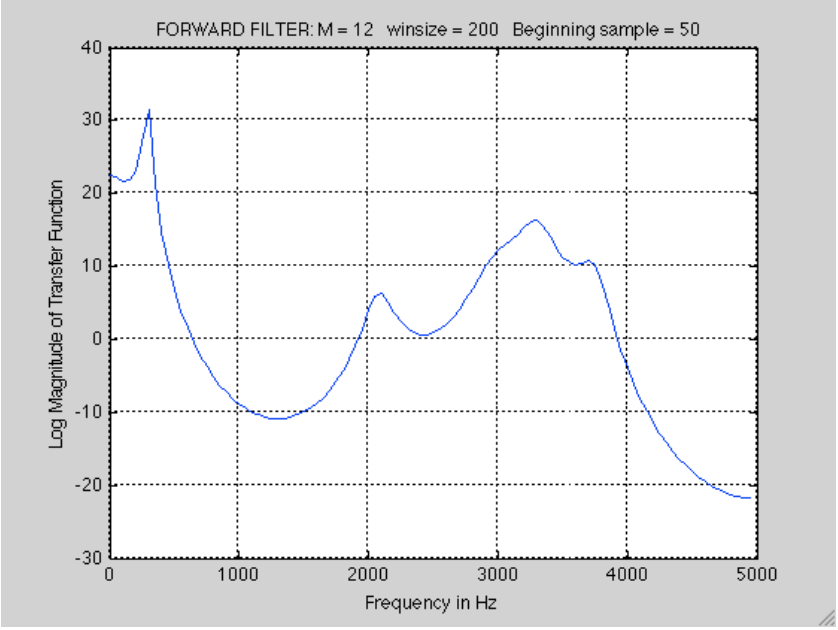
These results were obtained by analyzing the synthetic vowel iy, that we have generated previously (F=290 2070 2960 3300 3750), with an f0=100.
To run the lpc analysis I typed:

```
»acoef=lpc_demo(out,srate,200,12,5
```










```
»F = formants(acoef, srate)
F =
   1.0e+03 *
   0.2924
   2.0714
   2.9626
   3.30114
   3.7507
```

```
function [A, G] = get_lpc (signal, srate, M, window, slide)
%
% get_lpc
% Louis Goldstein
% March 2005
%
% Output arguments:
% A         filter coefficients (M+1 rows x  nframes columns)
% G         Gain coefficients (vecor length = nframes)
%
% input arguements:
% signal    signal to be analyzed
% srate     sampling rate in Hz
% M         LPC order  (def. = srate/1000 +4)
% window size of analysis window in ms (def. = 10)
% slide     no. of ms. to slide analysis window for each frame (def. = 5)

if nargin < 3, M = floor(srate/1000) + 4; end
if nargin < 4, window = 20; end
if nargin < 5, slide = 10; end

samp_tot = length(signal);
samp_win = fix((window/1000)*srate);
samp_slide = fix((slide/1000)*srate);
nframes = floor(samp_tot/samp_slide) - ceil((samp_win-samp_slide)/
samp_slide);

A = [];
G = [];

for i = 1:nframes
    begin = 1 + (i-1)*samp_slide;
    [Ai,Gi] = LPC (hamming(samp_win).*signal(begin:begin+samp_win-1),M);
    A = [A Ai'];
    G = [G Gi];
end
```