Linear Prediction Analysis of Speech Sounds

Berlin Chen 2003

References:

1. X. Huang et. al., Spoken Language Processing, Chapters 5, 6
2. J. R. Deller et. al., Discrete-Time Processing of Speech Signals, Chapters 4-6
3. J. W. Picone, "Signal modeling techniques in speech recognition," proceedings of the IEEE, September 1993, pp. 1215-1247

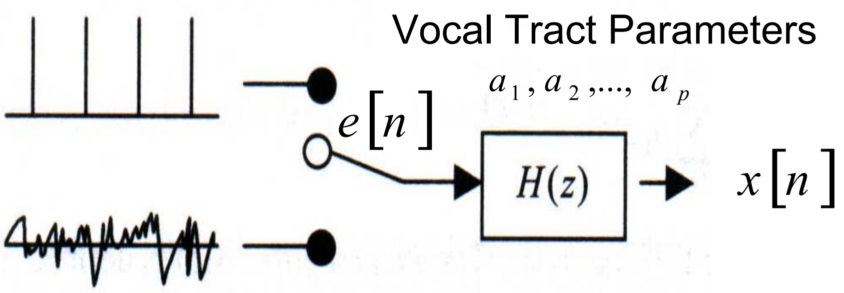
Linear Predictive Coefficients (LPC)

- An all-pole filter with a sufficient number of poles is a good approximation to model the vocal tract (**filter**) for speech signals

$$H(z) = \frac{X(z)}{E(z)} = \frac{1}{1 - \sum_{k=1}^{p} a_k z^{-k}} = \frac{1}{A(z)}$$

$$\therefore \quad x[n] = \sum_{k=1}^{p} a_k x[n-k] + e[n]$$

$$\tilde{x}[n] = \sum_{k=1}^{p} a_k x[n-k]$$



Source-filter model for voiced and unvoiced speech.

  - **It predicts the current sample as a linear combination of its several past samples**
    - Linear predictive coding, LPC analysis, auto-regressive modeling

Short-Term Analysis: Algebra Approach

- Estimate the corresponding LPC coefficients as those that minimize the total short-term prediction error (**minimum mean squared error**)

$$E_m = \sum_n e_m^2[n] = \sum_n \left(x_m[n] - \tilde{x}_m[n]\right)^2 \quad , \quad 0 \le n \le N-1$$

$$= \sum_n \left(x_m[n] - \sum_{j=1}^{p} a_j x_m[n-j] \right)^2$$

Framing/Windowing, The total short-term prediction error for a specific frame *m*

Take the derivative

$$\frac{\partial E_m}{\partial a_i} = \frac{\partial \left[\sum_n \left(x_m[n] - \sum_{j=1}^{p} a_j x_m[n-j] \right)^2 \right]}{\partial a_i} = 0, \qquad \forall\ 1 \le i \le p$$

$$\Rightarrow \sum_n \left[\left(x_m[n] - \sum_{j=1}^{p} a_j x_m[n-j] \right) x_m[n-i] \right] = 0, \qquad \forall\ 1 \le i \le p$$

$e_m[n]$

$$\sum_n \left\{ e_m[n] x_m[n-i] \right\} = 0, \qquad \forall\ 1 \le i \le p$$

The error vector is orthogonal to the past vectors

This property will be used later on!

Short-Term Analysis: Algebra Approach

$$\frac{\partial E_m}{\partial a_i} \Rightarrow$$

$$\sum_n \left[\left(x_m[n] - \sum_{j=1}^{p} a_j x_m[n-j] \right) x_m[n-i] \right] = 0, \ \forall \ 1 \le i \le p$$

$$\Rightarrow \sum_n \left[\sum_{j=1}^{p} a_j x_m[n-i] x_m[n-j] \right] = \sum_n [x_m[n-i] x_m[n]], \ \forall \ 1 \le i \le p$$

$$\Rightarrow \sum_{j=1}^{p} a_j \sum_n [x_m[n-i] x_m[n-j]] = \sum_n [x_m[n-i] x_m[n]], \ \forall \ 1 \le i \le p$$

To be used in next page !

Define correlation coefficients :

$$\phi_m[i,j] = \sum_n [x_m[n-i] x_m[n-j]]$$

$$\Rightarrow \sum_{j=1}^{p} a_j \phi_m[i,j] = \phi_m[i,0] \ , \forall \, 1 \le i \le p$$

$$\Rightarrow \boldsymbol{\Phi a} = \boldsymbol{\Psi}$$

Short-Term Analysis: Algebra Approach

- The minimum error for the optimal, $a_j,\ 1 \le j \le p$

$$E_m = \sum_n e_m^2[n] = \sum_n \left(x_m[n] - \tilde{x}_m[n]\right)^2 = \sum_n \left(x_m[n] - \sum_{j=1}^{p} a_j x_m[n-j] \right)^2$$

$$= \sum_n x_m^2[n] - 2\sum_n \left(x_m[n] \sum_{j=1}^{p} a_j x_m[n-j] \right) + \sum_n \left(\sum_{j=1}^{p} a_j x_m[n-j] \sum_{k=1}^{p} a_k x_m[n-k] \right)$$

equal

$$\sum_n \left(\sum_{j=1}^{p} a_j x_m[n-j] \sum_{k=1}^{p} a_k x_m[n-k] \right)$$

$$= \sum_{j=1}^{p} a_j \left\{ \sum_{k=1}^{p} a_k \sum_n \left(x_m[n-j] x_m[n-k] \right) \right\}$$

$$= \sum_{j=1}^{p} a_j \sum_n x_m[n-j] x_m[n]$$

Use the property derived in the previous page !

$$E_m = \sum_n x_m^2[n] - \sum_{j=1}^{p} a_j \sum_n \left(x_m[n] x_m[n-j] \right)$$

$$= \phi_m[0,0] - \sum_{j=1}^{p} a_j \phi_m[0,j]$$

Total Prediction Error

The error can be monitored to help establish *p*

Short-Term Analysis: Geometric Approach

- Vector Representations of Error and Speech Signals

$$x_m[n] = \sum_{k=1}^{p} a_k x_m[n-k] + e_m[n], \quad 0 \le n \le N\text{-}1$$

$$\begin{bmatrix} x_m[-1] & x_m[-2] & \dots & x_m[-p] \\ x_m[1-1] & x_m[1-2] & \dots & x_m[1-p] \\ . & . & \dots & . \\ . & . & \dots & . \\ . & . & \dots & . \\ x_m[N-1-1] & x_m[N-1-2] & \dots & x_m[N-1-p] \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ . \\ . \\ . \\ a_p \end{bmatrix} + \begin{bmatrix} e_m[0] \\ e_m[1] \\ . \\ . \\ . \\ e_m[N-1] \end{bmatrix} = \begin{bmatrix} x_m[0] \\ x_m[1] \\ . \\ . \\ . \\ x_m[N-1] \end{bmatrix}$$

the past vectors are as column vectors

$$\boldsymbol{X}\left(= \left[\boldsymbol{x}_m^1 \ \boldsymbol{x}_m^2 \dots \boldsymbol{x}_m^P\right]\right) \qquad \boldsymbol{a} \qquad \boldsymbol{e}_m \qquad \boldsymbol{x}_m$$

$$\boldsymbol{e}_m^T = \left(e_m[0], e_m[1], \dots, e_m[N-1]\right)$$

$$\boldsymbol{x}_m^{iT} = \left(x_m[-i], x_m[1-i], \dots, x_m[N-1-i]\right)$$

$$\boldsymbol{Xa} + \boldsymbol{e}_m = \boldsymbol{x}_m$$

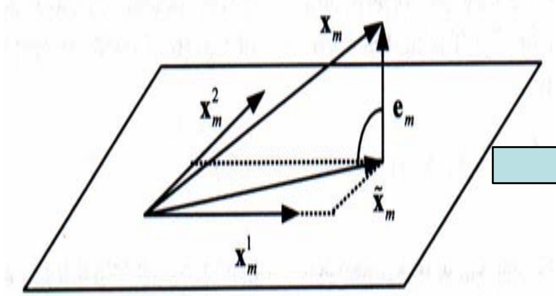
$\boldsymbol{e}_m$ is minimal if $\boldsymbol{X}^T \boldsymbol{e}_m = \boldsymbol{0}$

$$\Rightarrow \boldsymbol{X}^T\left(\boldsymbol{x}_m - \boldsymbol{Xa}\right) = \boldsymbol{0}$$

$$\Rightarrow \boldsymbol{X}^T \boldsymbol{Xa} = \boldsymbol{X}^T \boldsymbol{x}_m$$

$$\Rightarrow \boldsymbol{a} = \left(\boldsymbol{X}^T \boldsymbol{X}\right)^{-1} \boldsymbol{X}^T \boldsymbol{x}_m$$

This property has been shown previously



$$\left\langle \boldsymbol{e}_m, \boldsymbol{x}_m^i \right\rangle = 0, \forall 1 \le i \le p$$

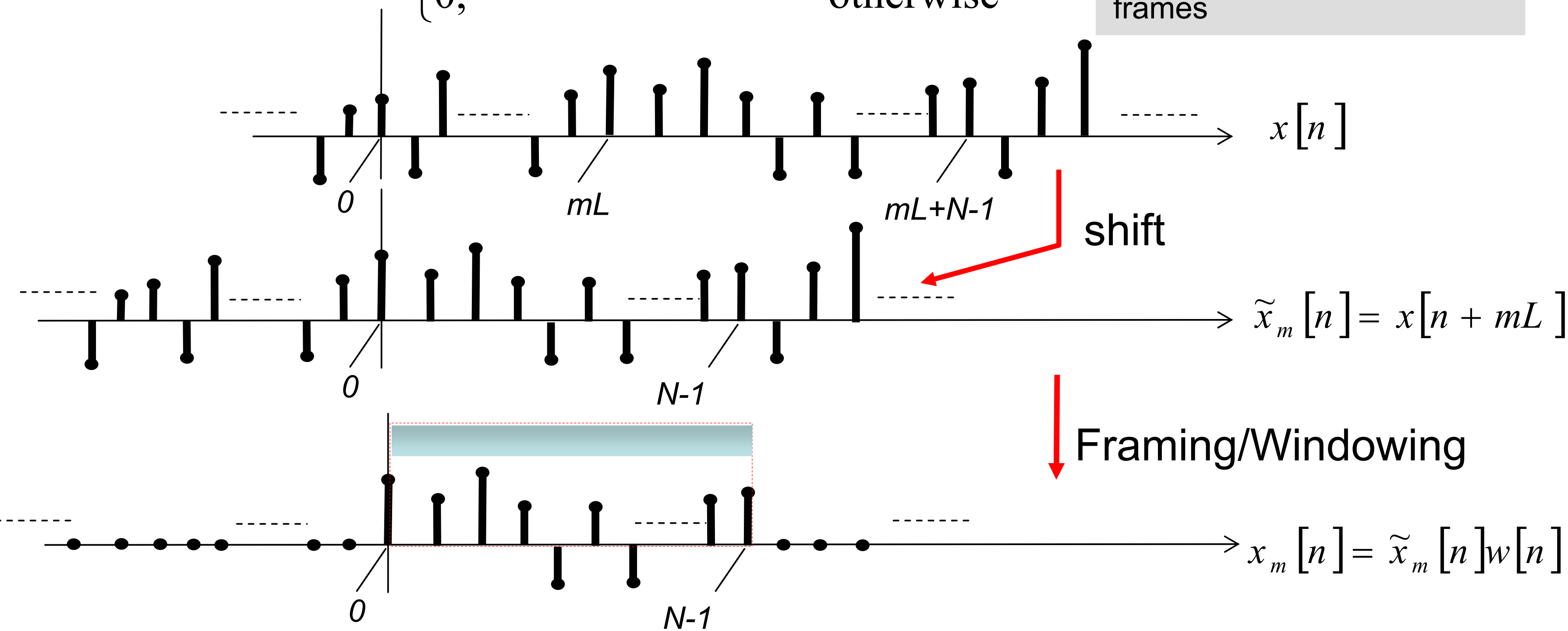
The prediction error vector must be orthogonal to the past vectors

Short-Term Analysis: Autocorrelation Method

- $x_m[n]$ is identically zero outside *0 ≤ n ≤ N-1*
- The mean-squared error is calculated within *n*=0~*N*-1+*p*

$$x_m[n] = \begin{cases} x[n+mL]w[n], & 0 \le n \le N-1 \\ 0, & \text{otherwise} \end{cases}$$

L: **Frame Period** , the length of time between successive frames

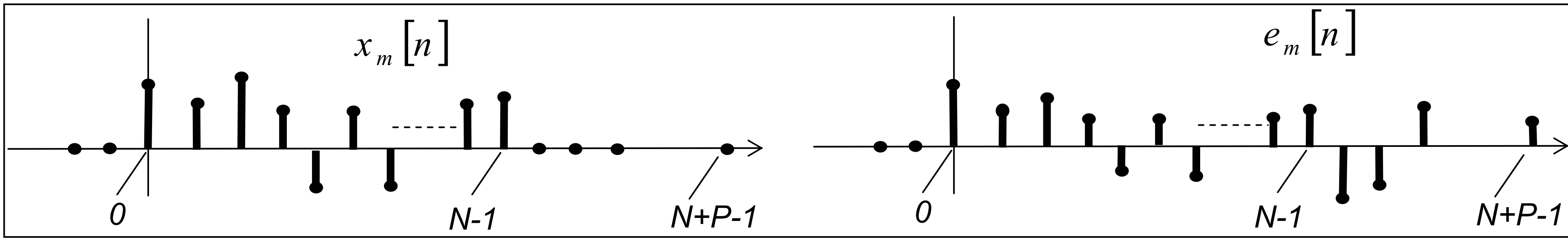


Short-Term Analysis: Autocorrelation Method

- The mean-squared error will be:

Why?

$$E_m = \sum_{n=0}^{N-1+p} e_m^2[n] = \sum_{n=0}^{N-1+p} \left(x_m[n] - \tilde{x}_m[n]\right)^2$$



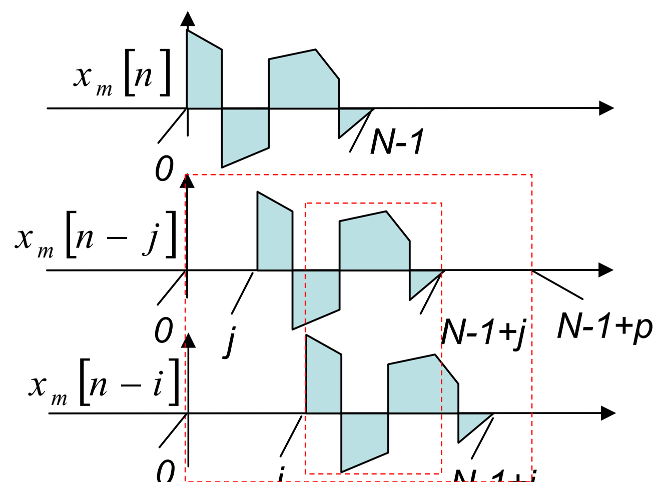
Take the derivative: $\frac{\partial E_m}{\partial a_i}$

$$\Rightarrow \sum_{j=1}^{p} a_j \phi_m[i,j] = \phi_m[i,0] \; , \forall \; 1 \le i \le p$$

$$\phi_m[i,j] = \sum_{n=0}^{N+p-1} x_m[n-i] x_m[n-j]$$

$$= \sum_{n=i}^{N-1+j} x_m[n-i] x_m[n-j]$$

$$= \sum_{n=0}^{N-1-(i-j)} x_m[n] x_m[n+(i-j)]$$



Short-Term Analysis: Autocorrelation Method

- Alternatively,
  - Where $\phi_m[i,j] = R[i-j]$ is the **autocorrelation function** of $x_m[n]$
  - And $R_m[k] = \sum_{n=0}^{N-1-k} x_m[n] x_m[n+k]$
- Therefore:

$$R_m[k] = R_m[-k]$$

Why?

$$\sum_{j=1}^{p} a_j \phi_m[i,j] = \phi_m[i,0] \;, \forall\; 1 \le i \le p$$

$$\Rightarrow \sum_{j=1}^{p} a_j R_m[|k|] = R_m[k] \;, \forall\; 1 \le i \le p$$

A Toeplitz Matrix:
symmetric and all elements of the diagonal are equal

$$\begin{bmatrix} R_m[0] & R_m[1] & \dots & R_m[p-1] \\ R_m[1] & R_m[0] & \dots & R_m[p-2] \\ . & . & \dots & . \\ . & . & \dots & . \\ . & . & \dots & . \\ R_m[P-1] & x_m[P-2] & \dots & R_m[0] \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ . \\ . \\ . \\ a_p \end{bmatrix} = \begin{bmatrix} R_m[1] \\ R_m[2] \\ . \\ . \\ . \\ R_m[p] \end{bmatrix}$$

Short-Term Analysis: Autocorrelation Method

- Levinson-Durbin Recursion

1. Initialization

$$E(0) = R_m[0]$$

2. Iteration. For i=1…,p do the following recursion

$$k(i) = \frac{R_m[i] - \sum_{j=1}^{i-1} a_j(i-1) R_m[i-j]}{E(i-1)}$$

$$a_i(i) = k(i)$$

A new, higher order coefficient is produced at each iteration *i*

$$a_j(i) = a_j(i-1) - k(i) a_{i-j}(i-1), \quad \text{for} \quad 1 \le j \le i\text{-}1$$

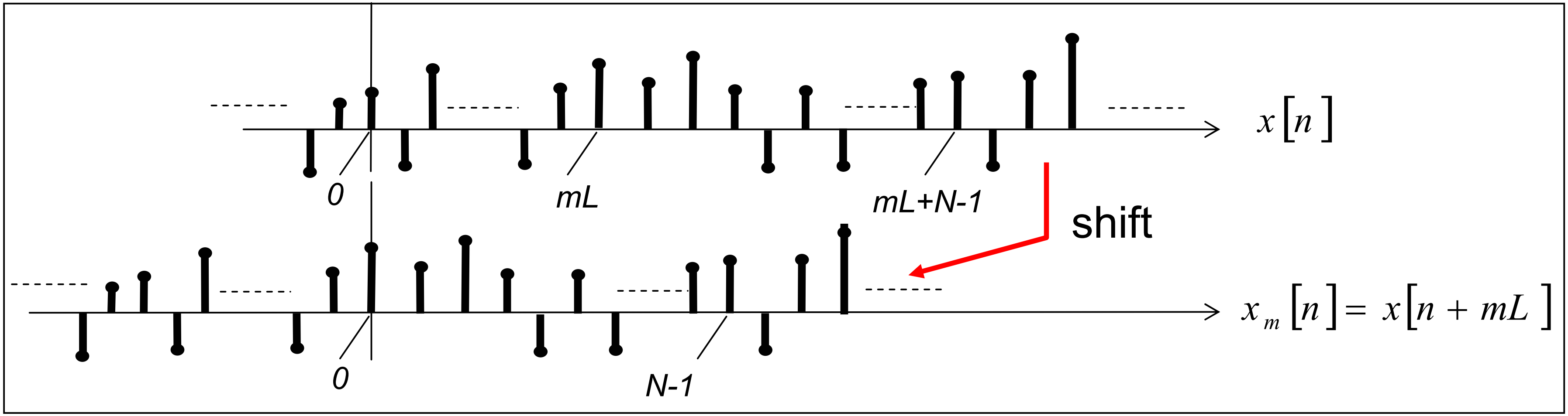
$$E(i) = \left(1 - [k(i)]^2\right) E(i-1), \ \text{where} \quad -1 \le k(i) \le 1$$

3. Final Solution:

$$a_j = a_j(p) \ \text{for} \quad 1 \le j \le p$$

Short-Term Analysis: Covariance Method

- $x_m[n]$ is not identically zero outside $0 \le n \le N\text{-}1$
  - Window function is not applied
- The mean-squared error is calculated within n=0~N-1



- The mean-squared error will be:

$$E_m = \sum_{n=0}^{N-1} e_m^2[n] = \sum_{n=0}^{N-1} \left(x_m[n] - \tilde{x}_m[n]\right)^2$$

Short-Term Analysis: Covariance Method

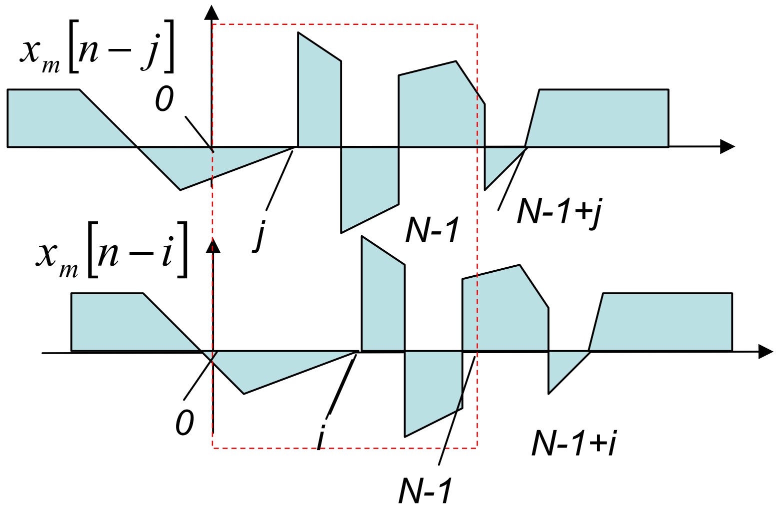
Take the derivative: $\dfrac{\partial E_m}{\partial a_i}$

$$\Rightarrow \sum_{j=1}^{p} a_j \phi_m[i,j] = \phi_m[i,0] \;, \forall\, 1 \le i \le p$$

$$\phi_m[i,j] = \sum_{n=0}^{N-1} x_m[n-i]\,x_m[n-j]$$

$$= \sum_{n=0}^{N-1} x_m[n-i]\,x_m[n-j]$$

$$= \sum_{n=-i}^{N-1-i} x_m[n]\,x_m[n+(i-j)]$$



$$\sum_{j=1}^{P} a_j \phi_m[i,j] = \phi_m[i,0] \;, \forall\, 1 \le i \le P$$

$$\begin{bmatrix} \phi_m[1,1] & \phi_m[1,2] & \dots & \phi_m[1,p] \\ \phi_m[2,1] & \phi_m[2,2] & \dots & \phi_m[2,p] \\ . & . & \dots & . \\ . & . & \dots & . \\ . & . & \dots & . \\ \phi_m[p,1] & \phi_m[p,2] & \dots & \phi_m[p,p] \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ . \\ . \\ . \\ a_P \end{bmatrix} = \begin{bmatrix} \phi_m[1,0] \\ \phi_m[2,0] \\ . \\ . \\ . \\ \phi_m[p,0] \end{bmatrix}$$

Not A Toeplitz Matrix:
symmetric and but not all elements of the diagonal are equal

$$\phi_m[1,1] \ne \phi_m[2,2] \dots \ne \phi_m[p,p]$$

LPC Spectra

- LPC spectrum matches more closely the peaks than the valleys

Parseval's theorem

$$E_m = \sum_{n=0}^{N-1+p} e_m^2[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} \left|E_m\left(e^{j\omega}\right)\right|^2 d\omega = G^2 = \frac{1}{2\pi}\int_{-\pi}^{\pi} \frac{\left|X_m\left(e^{j\omega}\right)\right|^2}{\left|H\left(e^{j\omega}\right)\right|^2} d\omega$$

$$H'\left(e^{jw}\right) = G \cdot H\left(e^{jw}\right)$$

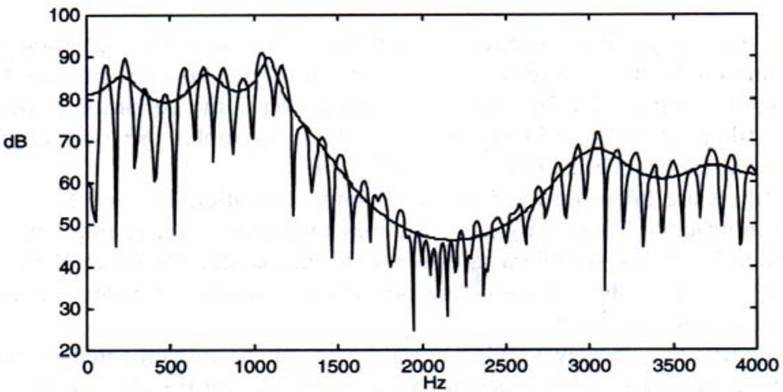


Figure 6.20 LPC spectrum of the */ah/* phoneme in the word *lives* of Figure 6.3. Used here are a 30-ms Hamming window and the autocorrelation method with $p = 14$. The short-time spectrum is also shown.

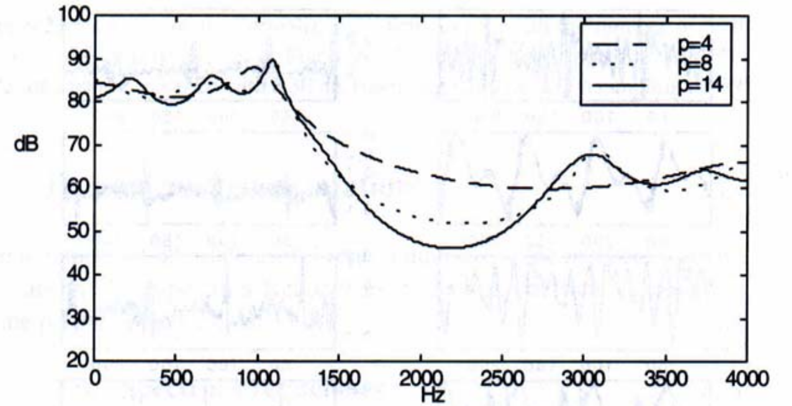


Figure 6.21 LPC spectra of Figure 6.20 for various values of the predictor order p.

  - Because the regions where $\left|X_m\left(e^{j\omega}\right)\right| > \left|H\left(e^{j\omega}\right)\right|$ contribute more to the error than those where $\left|H\left(e^{j\omega}\right)\right| > \left|X_m\left(e^{j\omega}\right)\right|$

LPC Spectra

- LPC provides estimate of a gross shape of the short-term spectrum

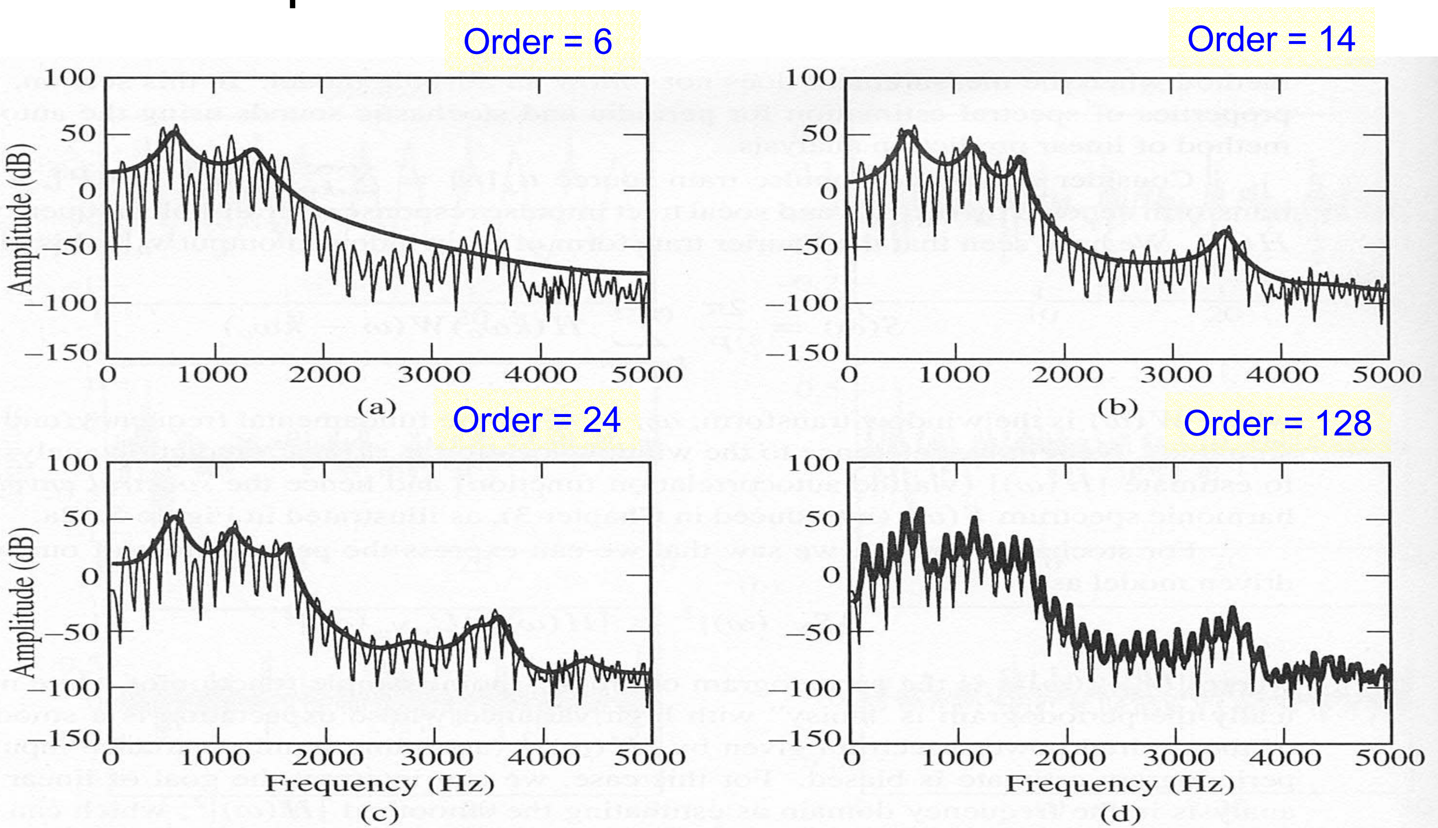


Figure 5.13 Linear prediction analysis of steady vowel sound with different model orders using the autocorrelation method: (a) order 6; (b) order 14; (c) order 24; (d) order 128. In each case, the all-pole spectral envelope (thick) is superimposed on the harmonic spectrum (thin), and the gain is computed according to Equation (5.30).

LPC Prediction Errors

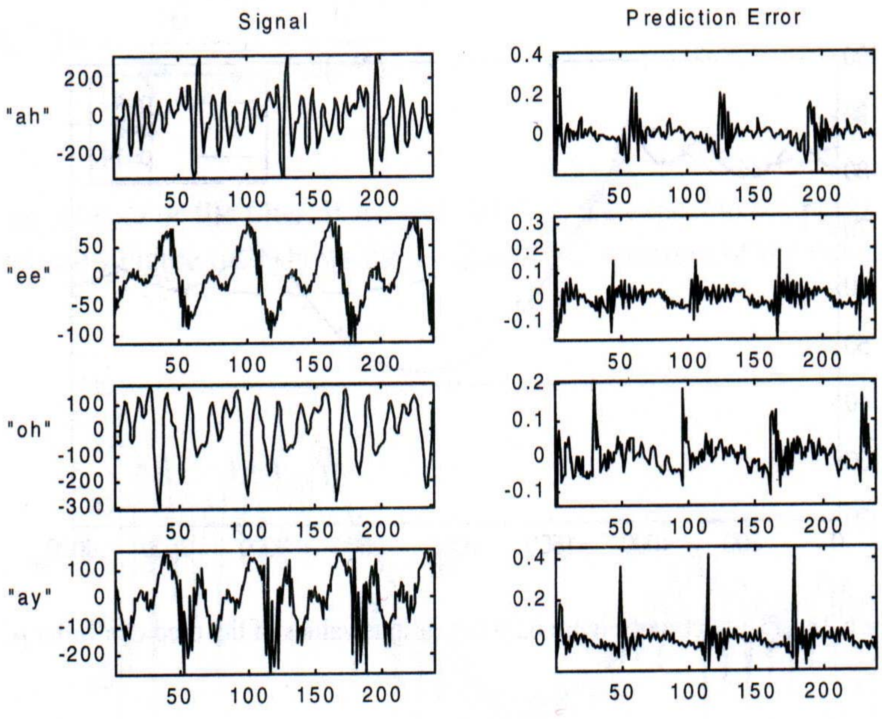


Figure 6.22 LPC prediction error signals for several vowels.

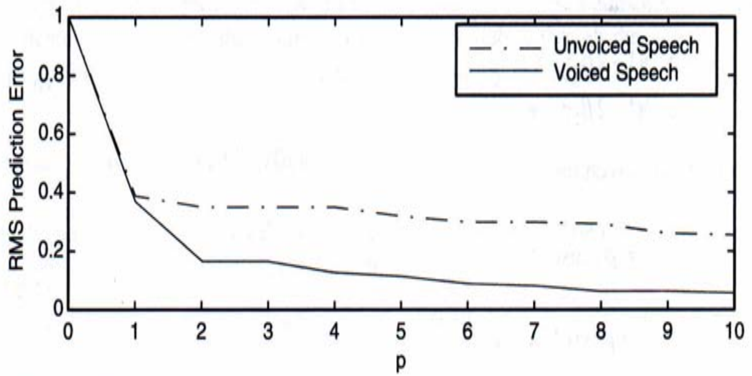


Figure 6.23 Variation of the normalized prediction error with the number of prediction coefficients p for the voiced segment of Figure 6.3 and the unvoiced speech of Figure 6.5. The autocorrelation method was used with a 30 ms Hamming window, and a sampling rate of 8 kHz.

MFCC vs. LPC Cepstrum Coefficients

- MFCC outperforms LPC Cepstrum coefficients
  - Perceptually motivated mel-scale representation indeed helps recognition

Table 9.2 Relative error reduction with different features. The reduction is relative to that of the preceding row.

Feature Set	Relative Error Reduction
13th-order LPC cepstrum coefficients	Baseline
13th-order MFCC	+10%
16th-order MFCC	+0%
+1st- and 2nd-order dynamic features	+20%
+3rd-order dynamic features	+0%

- Higher-order MFCC does not further reduce the error rate in comparison with the 13-order MFCC
- Another perceptually motivated features such as first- and second-order delta features can significantly reduce the recognition errors

Description of Project-2 **Fall 2003**

- Try to implement the short-term linear prediction coding (LPC) for speech signals

- You should follow the following instructions:

  1. Using the autocorrelation method with Levinson-Durbin Recursion and Rectangular/Hamming windowing

  2. Analyzing the vowel (or FINAL) portions of speech signal with different model orders (different P, e.g. P=6, 14, 24 and 128)

  3. Plotting the LPC spectra as well as the original speech spectrum

  4. Using the speech wave file, bk6_1.wav (no header, PCM 16KHz raw data), as the exemplar

Description of Project-2

Fall 2003

- Hints:
  1. When the LPC coefficients a_j are derived, you can construct impulse response signal *h*[*n*], $0 \leq n \leq N-1$ (*N*: frame size) by:

$$h[n] = \sum_{j=1}^{P} a_j \cdot h[n-j] + \delta[n]$$

$$or$$

$$h[n] = \begin{cases} 1, & \text{if } n = 0 \\ \sum_{j=1}^{P} a_j \cdot h[n-j], & \text{if } n \neq 0 \end{cases}$$

  2. The prediction Error E can be expressed by the autocorrelation function:

$$E = R_m[0] - \sum_{j=1}^{P} a_j \cdot R_m[j]$$

Description of Project-2

Fall 2003

3. The MATLab example code:

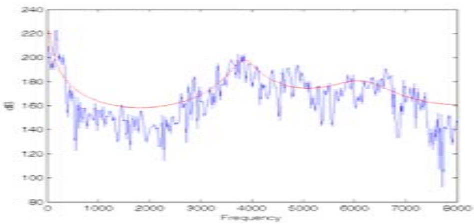
```
x=[184.6400 184.1251 . . . . . . .  197.7890 -26.8000 ]; % original signal, dimension: frame size
y=[1.0000 2.0105  . . . . . . .   0.0738 0.0565 ]; % filter's impulse response h[n], dimension: frame size
gain=valG; %   valG: the prediction Error E
X=fft(x,512); % fast Fourier Transform, so the frame size < 512
Y=fft(y,512);  % fast Fourier Transform
X(1)=[]; % remove the X(1), the DC
Y(1)=[]; % remove the Y(1), the DC
M=512;
powerX=abs(X(1:M/2)).^2;  % the power spectrum of X
logPX=10*log(powerX); % the power spectrum of X in dB
powerY=abs(Y(1:M/2)).^2; % the power spectrum of Y
logPY=10*log(powerY)+10*log(gain); % the power spectrum of Y in dB
                                    % plus the gain (Error) in dB
nyquist=8000; % maximal frequency index
freq=(1:M/2)/(M/2)*nyquist; % an array store the frequency indices
figure(1);

plot(freq,logPX,'b',freq,logPY,'r');   % plot the result,
                                       % b: blue line for the power spectrum of the original signal
                                       % r: red line for the power spectrum of the filter
```

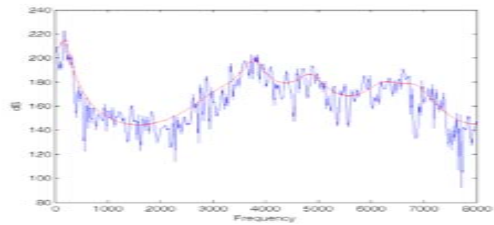
Description of Project-2

Fall 2003

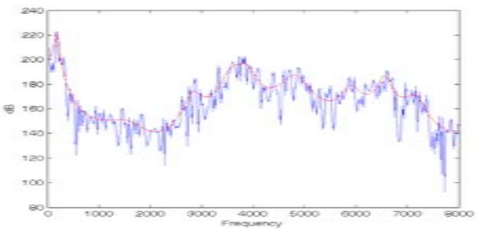
- Example Figures of LPC Spectra



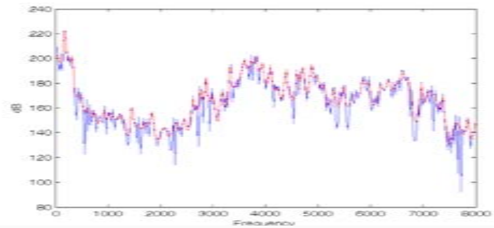
Order = 6
Rectangle window
No pre-emphasis



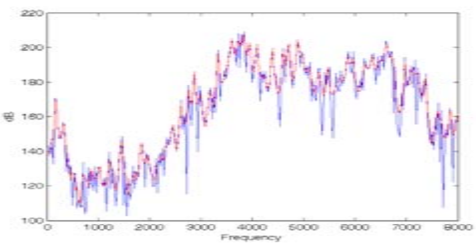
Order = 14
Rectangle window
No pre-emphasis



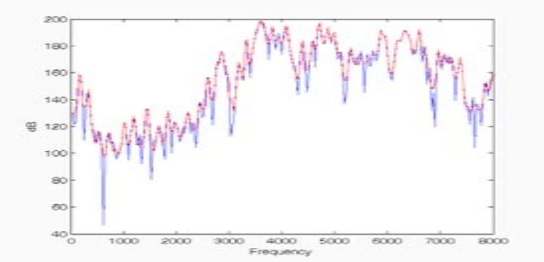
Order = 24
Rectangle window
No pre-emphasis



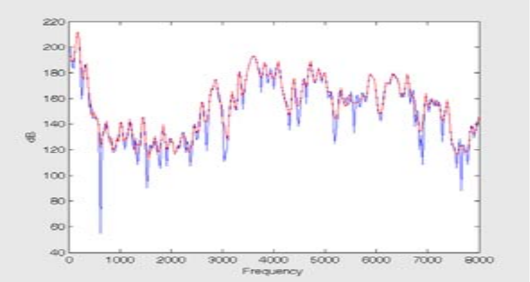
Order = 128
Rectangle window
No pre-emphasis



Order = 128
Rectangle window
Pre-emphasis



Order = 128
Hamming window
Pre-emphasis



Order = 128
Hamming window
No pre-emphasis

Plotted by Roger Kuo, Fall 2002