Linear Prediction

87203 – Multimedial Signal Processing 1st Module

Politecnico di Milano –
Polo regionale di Como

Agenda

- Ex1: LPC Autocorrelation method
- Ex2: LPC Levinson Durbin algorithm
- Ex3: PSD estimation using LPC
- Ex4: Spectrum envelope and prediction error
- Ex5: Prediction error

Exercise 1 (1/9)

- Basic idea of Linear Prediction: a sample of a discrete-time signal can be approximated (predicted) as a linear combination of past samples.

$$\hat{s}(n) \approx a_1 s(n-1) + \ldots + a_p s(n-p) = \sum_{k=1}^{p} a_k s(n-k)$$

- The prediction error is:

$$e(n) = s(n) - \hat{s}(n) = s(n) - \sum_{k=1}^{p} a_k s(n-k)$$

$$E(z) = \left(1 - \sum_{k=1}^{p} a_k z^{-k}\right) S(z) = A(z) S(z)$$

Exercise 1 (2/9)

- Goal: Find the set of predictor coefficients $\{a_k\}$ that minimizes the mean-squared prediction error over a short segment of the signal $s_n(m)$.

$$a_i = \arg\min_{a_i} E_n$$

$$E_n = \sum_m e_n^2(m) = \sum_m \left[s_n(m) - \sum_{k=1}^{p} a_k s_n(m-k) \right]^2$$

$$E_n = \psi_n(0,0) - \sum_{k=1}^{p} \hat{a}_k \psi_n(0,k)$$

Exercise 1 (3/9)

- Hint: To solve the minimization problem we differentiate E_n with respect to each a_i and set the result to zero:

$$\frac{\partial E_n}{\partial a_i} = 0 \quad \forall i$$

- obtaining: $\psi_n(i,0) = \sum_{k=1}^{p} a_k \psi_n(i,k) \qquad i = 1,2,\ldots, p$

(Wiener-Hopf equations)

- defining the covariance matrix:

$$\psi_n(i,k) = \sum_m s_n(m-k) s_n(m-i)$$

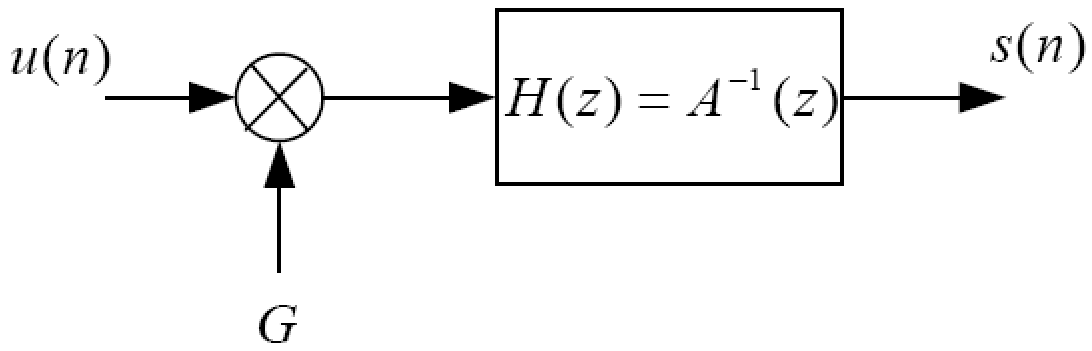
Exercise 1 (4/9)

- Goal: Computation of the LPC parameters.
- **Autocorrelation Method**
  - Defining: $$\hat{r}_n(k) = \sum_{m=0}^{M-1-k} s_n(m)s_n(m+k)$$
  - We find the LPC coefficients solving:

$$\begin{bmatrix} \hat{r}_n(0) & \hat{r}_n(1) & \ldots & \hat{r}_n(p-1) \\ \hat{r}_n(1) & \hat{r}_n(0) & \ldots & \hat{r}_n(p-2) \\ . & . & . & . \\ \hat{r}_n(p-1) & \hat{r}_n(p-2) & \ldots & \hat{r}_n(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ . \\ a_p \end{bmatrix} = \begin{bmatrix} \hat{r}_n(1) \\ \hat{r}_n(2) \\ . \\ \hat{r}_n(p) \end{bmatrix}$$

Exercise 1 (5/9)

- Goal: Assume that the signal can be modeled as an AR stochastic process defined by:



$$A(z) = 1 - 0.8z^{-1} - 0.1z^{-2} - 0.05z^{-3}$$

$$u(n) \approx N\,(0,1) \qquad\qquad G = 1$$

Exercise 1 (6/9)

- Goal: Estimate the optimal prediction coefficients of order p=3:
- Pseudocode:
  - Generate the input signal s(n).
  - Estimate the autocorrelation sequence.
  - Compute Toeplitz matrix
  - Estimate the LPC parameters.
  - Apply the whitening filter obtaining the prediction error.
  - Compute the mean square error.

Exercise 1 (7/9)

- Generate the input signal s(n).
  - s = filter(1, [1 ; -a], e);
- Estimate the autocorrelation sequence.

  $$\{\hat{r}_n(i)\} \quad i = 0,\dots,p$$

  - r = zeros(p+1, 1);
  - for t=0 : p
    - for m = 0 :M-1-t
      - r(t+1) = r(t+1) + s(m+1)*s(m+t+1);
    - end
  - end
- Compute Toeplitz matrix
  - R = toeplitz(r(1:p));

Exercise 1 (8/9)

- Estimate the LPC parameters.
  - `aest = R^(-1)*r(2:p+1);`
- Apply the whitening filter obtaining the prediction error.
  - `eres = filter([1 ; -aest], 1, s);`
- Compute the mean square error.
  - `mean(eres.^2)`

Exercise 1 (9/9)

- Hint: Matlab provides the functions:
  - “[r lag]=xcorr(x,’biased’)” that produces a biased estimate of the autocorrelation (2N-1 samples) of the stationary sequence “x”. “lag” is the vector of lag indices [-N+1:1:N-1].
  - “R=toeplitz(C,R)” that produces a non-symmetric Toeplitz matrix having C as its first column and R as its first row.

    “R=toeplitz(R)” is a symmetric (or Hermitian) Toeplitz matrix.

Exercise 1b (1/4)

- Load 'mtlb.mat' signal.
- Extract a 20msec long frame.
- Estimate the optimal prediction coefficients of order p = 16.
- Compute the prediction error.
- Estimate the spectra of the signal and the prediction error using the periodogram method.

Exercise 1b (2/4)

- Pseudocode:
  - Load ’mtlb.mat’ signal.
  - Extract a 20msec long frame.
    - x = x(700:700+floor(0.02*Fs));
  - Estimate the optimal prediction coefficients of order p = 16.
    - [r , k] = xcorr(x , p);
    - rv = r(k>= 0);
    - a = levinson(rv);
  - Compute the prediction error.
    - e = filter(a, 1, x);

Exercise 1b (3/4)

- Pseudocode (continued):
  - Estimate the spectra of the signal and the prediction error using the periodogram method.
    - N = length(x);
    - X = (1/N)*abs(fft(x));
    - E = (1/N)*abs(fft(e));
    - f = Fs*[0 :N-1]/N;

Exercise 1b (4/4)

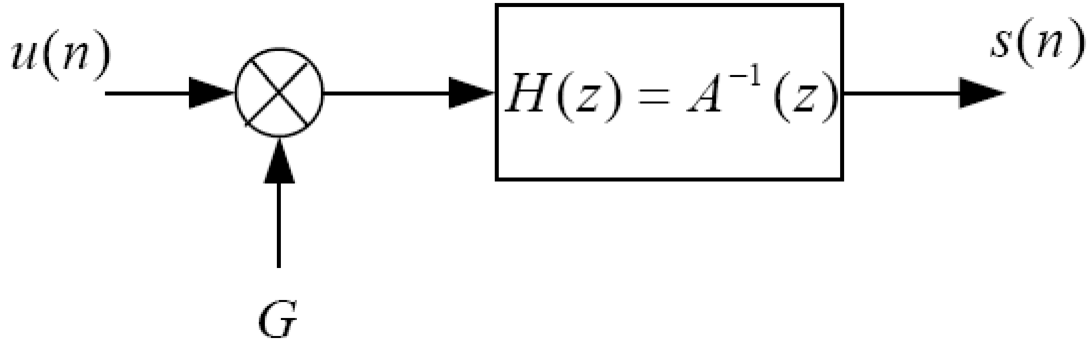
- Hint: Matlab provides the function:
  - “a = levinson(r, N)” that solves the Hermitian Toeplitz system of equations (also known as the Yule-Walker AR equations) using the Levinson-Durbin recursion. Input “r” is typically a vector of autocorrelation coefficients with lag 0 as the first element.

N is the order of the recursion; if omitted, N = length(r)-1.

“a” will be a row vector of length N+1, with a(1)=1.0.

Exercise 2 (1/9)

- Assume that the signal can be modeled as an AR stochastic process defined by:



$$A(z) = 1 - 0.8z^{-1} - 0.1z^{-2} - 0.05z^{-3}$$

$$u(n) \approx N\ (0,1) \qquad\qquad G = 1$$

Exercise 2 (2/9)

- Goal: Estimate the optimal prediction coefficients of order p=6 by means of the Levinson-Durbin algorithm:
- Pseudocode:
  - Generate the input signal s(n).
  - Estimate the LPC parameters.
  - Apply the whitening filter obtaining the prediction error.
  - See how evolves the prediction during the steps.

Exercise 2 (3/9)

- Levinson-Durbin algorithm:

For $i = 1, 2, \ldots, p$, we carry out the following recursion:

1. Initialization

$$E^{(0)} = \hat{r}(0) \tag{8.47}$$

2. Compute 'partial correlation' (PARCOR) coefficients

$$k_i = \left[\hat{r}(i) - \sum_{j=1}^{i-1} a_j^{i-1} \hat{r}(i-j) \right] / E^{(i-1)}, \quad 1 \le i \le p \tag{8.48}$$

Exercise 2 (4/9)

- Levinson-Durbin algorithm (continued):

3. Calculate the predictor coefficients for order i

$$a_i^{(i)} = k_i \tag{8.49}$$

$$a_j^{(i)} = a_j^{(i-1)} - k_i a_{i-j}^{(i-1)}, \quad 1 \le j \le i-1 \tag{8.50}$$

4. Update the predictor error

$$E^{(i)} = (1 - k_i^2) E^{(i-1)} \tag{8.51}$$

5. Obtain the final solution

$$a_j = a_j^{(p)} \tag{8.52}$$

Exercise 2 (5/9)

- PSEUDOCODE: Estimate the LPC parameters:
  - Estimate autocorrelation:
    - r = xcorr(s,p);
    - r = r(p+1: end);
  - Inizializing:
    - Eest = zeros(p+1, 1); % error estimation
    - kest = zeros(p, 1);   % PARCOR coefficients
    - aest = zeros(p, p);   % LPC parameters estimated in each step (in each column)
    - Eest(1) = r(1);

Exercise 2 (6/9)

- PSEUDOCODE: Estimate the LPC parameters (continued):
  - For i=1:p   % for each step
    - temp = 0 ;
    - for j = 1 : i-1
      - temp = temp + aest(j, i-1)*r(i-j + 1);
    - end
    - kest(i) = (r(i+1) - temp)/ Eest(i);

$$k_i = \left[\hat{r}(i) - \sum_{j=1}^{i-1} a_j^{i-1} \hat{r}(i-j) \right] / E^{(i-1)}, \quad 1 \le i \le p$$

Exercise 2 (7/9)

- PSEUDOCODE: Estimate the LPC parameters (continued):
  - For i=1:p   % for each step
    - aest(i,i) = kest(i);
    - for j =1: i-1
      - aest(j,i) = aest(j,i-1) - kest(i)*aest(i-j,i-1);
    - end

$$a_i^{(i)} = k_i$$

$$a_j^{(i)} = a_j^{(i-1)} - k_i a_{i-j}^{(i-1)}, \quad 1 \leq j \leq i-1$$

Exercise 2 (8/9)

- PSEUDOCODE: Estimate the LPC parameters (continued):
  - For i=1:p   % for each step
    - Eest(i+1) = (1- kest(i)^2)* Eest(i);
  - end
  - aest(:,p)

$$E^{(i)} = (1 - k_i^2)E^{(i-1)}$$

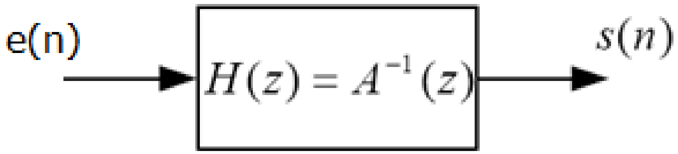
$$a_j = a_j^{(p)}$$

Exercise 2 (9/9)

- PSEUDOCODE: Generate the input signal s(n).
  - s = filter(1, [1 ; -a], e);
- Estimate the LPC parameters
  - a=aest(:,p);
- Apply the whitening filter obtaining the prediction error.
  - eres = filter([1 ; -a], 1, s);
- See how evolves the prediction during the steps (growing the number of known samples for the autocorrelation computation).
  - plot(Eest(2:end))

Exercise 3 (1/4)

- Frequency Domain interpretation of LPC parameters:



$$E(z) = A(z)\hat{S}(z)$$

- Goal: Estimate the PSD of the signal s(n) obtained by means LPC:

$$\left|\hat{S}(\omega)\right|^2 = \sigma_e^2 \left|H_p(\omega)\right|^2$$

Exercise 3 (2/4)

- Pseudocode:
  - Load ‘mtlb.mat’ signal.
  - Extract a Nfft sample long frame.
  - Estimate the optimal prediction coefficients of order p.
  - Compute the prediction error.
  - Estimate the PSD of ‘mtlb’ using LPC coefficients
  - Estimate the PSD of ‘mtlb’ using periodogram.

Exercise 3 (3/4)

- Pseudocode:
  - Load ‘mtlb.mat’ signal.
  - Extract a Nfft sample long frame.
    - s = mtlb(701:701 + Nfft);
  - Estimate the optimal prediction coefficients of order p.
    - [rs,k] = xcorr(s,p);
    - r = rs(k>= 0);
    - a = levinson(r);
  - Compute the prediction error.
    - e = filter(a, 1, x);

Exercise 3 (4/4)

- Pseudocode (continued):
  - Estimate the PSD of ‘mtlb’ using LPC coefficients
    - w = 2*pi*[0 : Nfft-1]/Nfft;
    - H= freqz(1, a, w);
    - sigmae=var(e);
    - phi_LPC=sigmae*abs(H).^2;
  - Estimate the PSD of ‘mtlb’ using periodogram
    - phip = (1/Nfft)*abs(fft(s,Nfft)).^2;
    - phip(1 : Nfft/2);

Exercise 4 (1/7)

- **Goal**: compute the spectrum envelope and the prediction error by means of linear prediction
- **Procedure**:
  - Load the signal voiced_a.wav
  - Segment it in frames of duration 25 ms
  - For each segment
    - Compute the autocorrelation function
    - Set up the Wiener-Hopf equations with a prediction order of P = 12
    - Continue...

Exercise 4 (2/7)

- Continued...
- Compute the spectral envelope and plot it in the frequency domain
- Compute the prediction error both in the time and frequency domain
- Compute the coding gain

Exercise 4 (3/7)

- Pseudocode:
  - Load ‘voiced_a.wav’ signal.
  - Segment it in frames of duration 25 ms
    - fl = 25/1000;
    - M = floor(fl*Fs);
    - N=10;   % number of frames
- For each segment
    - for n=0:N-1
      - sn = s(n*M + 1: n*M + M);

Exercise 4 (4/7)

- Compute the autocorrelation function
  - [rs, lags] = xcorr(sn, p);
- Set up the Wiener-Hopf equations with a prediction order of P = 12
  - R = toeplitz(rs(lags>=0 & lags < p));
  - rp = rs(lags>=1);
  - theta = -inv(R)*rp;

Exercise 4 (5/7)

- Compute the spectral envelope and plot it in the frequency domain
  - var_s = rs(lags == 0) + theta'*rp;
  - [H, w] = freqz(1, [1; theta], w);
  - f = w.*Fs/(2*pi);
  - P = var_s.*H.^2;
  - plot(f(1:end/2), 10*log10(P(1:end/2)),'r')
- Compare with the real spectrum:
  - Sn = fft(sn, Nfft);
  - plot(f(1:end/2), 10*log10(abs(Sn(1:end/2).^2)))

Exercise 4 (6/7)

- Compute the prediction error both in the time and frequency domain
  - e = filter([1, theta'], 1, sn);
  - E = fft(e, length(f));

- Compute the coding gain
  - Dp = var(e)
  - var(sn)
  - Gp = var(sn) / Dp;
  - display(['Coding Gain (p = ' int2str(p) '): ' num2str(Gp)])

Exercise 4 (7/7)

- **Question**:
  - Try to change the prediction order and observe the effect on:
    - Prediction gain;
    - Shape of the spectral envelope.

Exercise 5 (1/8)

- **Goal**: compute the prediction error using linear prediction and its PSD using Bartlett method.
- An all-pole LTI filter is defined by the following:

$$A(z) = 1 - \sum_{i=1}^{7} a_i z^{-i}$$

- Feed a white noise process e(n) into the system to produce one realization y(n) of N = 1000 samples.
- Estimate the optimal prediction coefficients of order p = 0 : 2 : 20.

Exercise 5 (2/8)

- For each value of p, compute the prediction error and the prediction gain Gp.
- Estimate the PSD of the prediction error using a non-parametric method. Check that the PSD tends to become flat (white spectrum) when $p \rightarrow \infty$.
- Estimate the PSD using a parametric method with the appropriate number of parameters.
- Plot the prediction gain Gp vs. p

Exercise 5 (3/8)

- Hint: *Bartlett method* :split up the available sample of N observations into L = N/M subsamples of M observations each, then average the periodograms obtained from the subsamples for each value of ω.

$$y_i(n) = y((i-1)M + n) \quad n = 1, \ldots, M$$

$$\hat{\phi}_i(\omega) = \frac{1}{M}\left|\sum_{n=1}^{M} y_i(n)\, e^{-j\omega n}\right|^2 \qquad \hat{\phi}_B(\omega) = \frac{1}{L}\sum_{i=1}^{L} \hat{\phi}_i(\omega)$$

Exercise 5 (4/8)

- Pseudocode:
  - Feed a white noise process e(n) into the system to produce one realization y(n) of N = 1000 samples.
    - poles = [...];
    - N = 1000;
    - a = poly(poles)

    a = [1.0000 -3.5988 6.9925 -9.4813 9.0849 -6.1870 2.9012 -0.7093]
    - z = randn(N,1);
    - x = filter(1,a,z);

Exercise 5 (5/8)

- Pseudocode:
- For each value of p, compute the prediction error and the prediction gain Gp.
  - for p = 2:2:P
    - [r,k] = xcorr(x,p);
    - rv = r(k>=0);
    - a = levinson(rv);
    - e = filter(a, 1, x);
    - G(p) = var(x)./var(e);

Exercise 5 (6/8)

- Pseudocode:
- For each value of p, estimate the PSD of the prediction error using Bartlett method. (S=10; M=N/S;)
  - s = 1 ;
  - for m = 0 : M : N-M
    - em = e(m+1 : m+M);
    - E(s, :) = (1/M)*abs(fft(em,M)).^2;
    - s = s + 1 ;
  - end
  - PHI = mean(E, 1);

Exercise 5 (7/8)

- Pseudocode:
- For each value of p, estimate the PSD using a parametric method with the appropriate number of parameters.
  - [re , k] = xcorr(e, p);
  - rve = re(k>= 0);
  - ae = levinson(rve);
  - [He , we] = freqz (1, ae, Nfft);
  - phi_AR=abs(He).^2

Exercise 5 (8/8)

- Feed a different realization of white noise e(n) to produce a new realization y(n).
- Check that the new realization has a different waveform in the time domain but has the same PSD.