Linear Prediction

Course Instructor: Dr. Debashis Ghosh

Department of Electronics & Computer Engg.

Indian Institute of Technology Roorkee

Syllabus

- Forward and backward prediction error filters; Levinson—Durbin algorithm; Properties of prediction-error filters; Autoregressive modeling of a stationary stochastic process; All-pole, all-pass lattice filter (8 lectures).

Introduction

- Linear Prediction – Observing the past or future samples and predict the current sample as a linear combination of the observed samples.
  - Forward Prediction – predicting from past samples, i.e. observing the past and predict the future.
  - Backward Prediction – predicting from future samples, i.e. observing the future and predict the past.

Forward linear prediction

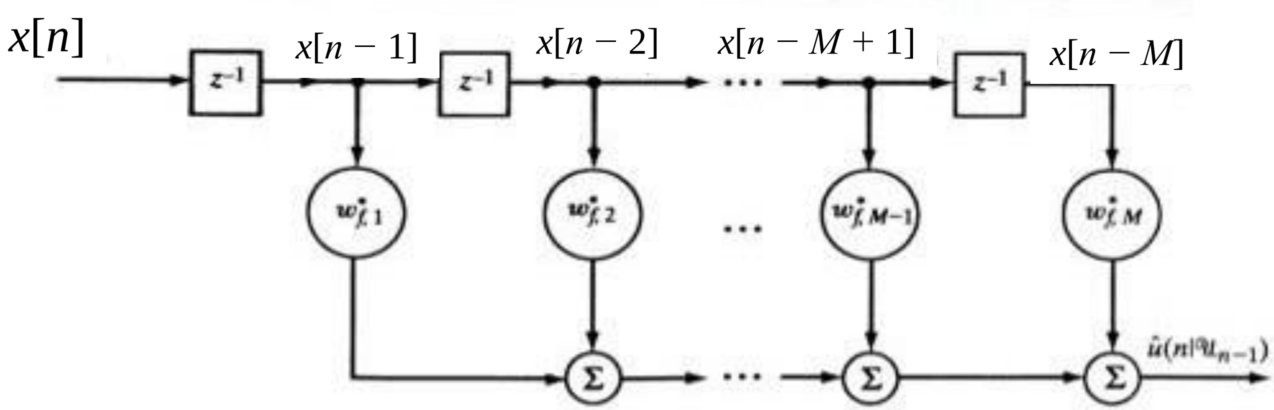
- Problem:
  - Forward Prediction
    - Observing the past

$$[x[n-1] \quad x[n-2] \quad \ldots \quad x[n-M]]$$

    - Predict the future

$$\hat{x}(n|\boldsymbol{X}_{n-1})$$

    - i.e. find the predictor filter taps $w_{f,1}, w_{f,2}, \ldots, w_{f,M}$



One-step predictor

$$\hat{x}(n|\boldsymbol{X}_{n-1}) = \sum_{k=1}^{M} w_{f,opt,k}^{*} x[n-k]$$

Forward linear prediction

- Use Wiener filter theory to find the filter weights of the predictor.
- Input vector $\mathbf{x}[\mathrm{n}] = [x[n-1] \quad x[n-2] \quad \ldots \quad x[n-M]]$
- Desired output $d[n] = x[n]$

$$\mathbf{R}_{XX}\mathbf{w}_{f,opt} = \mathbf{r}$$

- where

$$\mathbf{w}_{f,opt} = [w_{f,opt,1} \quad w_{f,opt,2} \quad \cdots \quad w_{f,opt,M}]^{\boldsymbol{T}}$$

$$\mathbf{r} = [r(-1) \quad r(-2) \quad \ldots \quad r(-M)]^T$$

$$= [r^*(1) \quad r^*(2) \quad \ldots \quad r^*(M)]^T$$

Forward linear prediction

- Forward prediction error

$$f_M[n] = x[n] - \hat{x}(n|\boldsymbol{X}_{n-1}) = x[n] - \sum_{k=1}^{M} w_{f,opt,k}^{*} x[n-k]$$

- Minimum mean-square prediction error (forward prediction error power)

$$P_M = r(0) - \mathbf{r}^H \mathbf{w}_{f,opt}$$

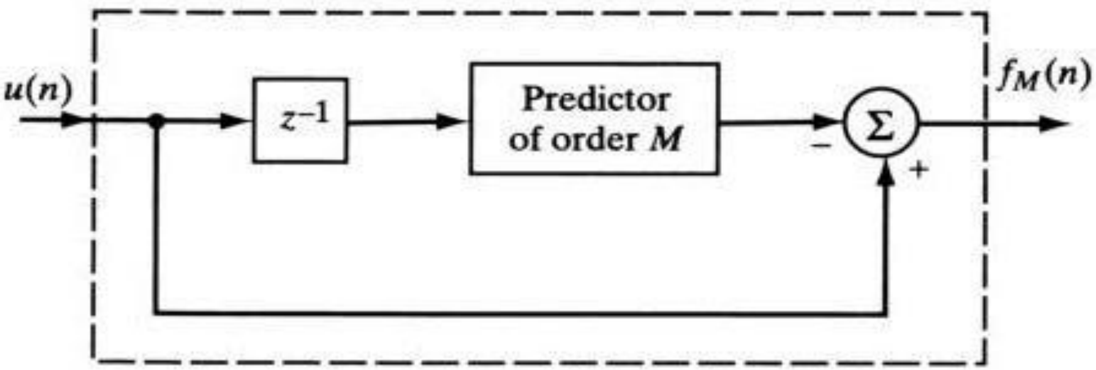
Relation b/w linear prediction and AR Modelling

- Note that the Wiener-Hopf equations for a linear predictor is mathematically identical with the Yule-Walker equations for the model of an AR process.
- If AR model order *M* is known, model parameters can be found by using a forward linear predictor of order *M*.
- If the process is not AR, predictor provides an (AR) model approximation of order *M* of the process.

Forward prediction-error filter

- Input vector here: $\mathbf{x}[\mathrm{n}] = [x[n] \quad x[n-1] \quad \ldots \quad x[n-M]]$
- Desired output = prediction error

$$f_M[n] = x[n] - \hat{x}(n|\boldsymbol{X}_{n-1}) = x[n] - \sum_{k=1}^{M} w_{f,opt,k}^{*} x[n-k]$$



Prediction-error filter

Forward prediction-error filter

- Let, we design the prediction error filter in line with Wiener filter with tap-weight vector

$$\mathbf{a}_M = [a_{M,0} \quad a_{M,1} \quad \cdots \quad a_{M,M}]^{\boldsymbol{T}}$$

- We can then obtain the desired response (prediction error) by taking

$$a_{M,k} = \begin{cases} 1 & k = 0 \\ -w_{f,opt,k} & k = 1, 2, \ldots, M \end{cases}$$

- Note the filter order is still *M* since it uses *M* delay elements.
- Therefore, output: $f_M[n] = \sum_{k=0}^{M} a_{M,k}^{*} x[n-k] = \mathbf{a}_M^H \mathbf{x}[n]$

Augmented W-H equations for forward prediction

- Let us combine the forward prediction filter and forward prediction-error power equations in a single matrix expression, i.e.

$$\mathbf{R}\mathbf{w}_f = \mathbf{r} \quad \text{and} \quad P_M = r(0) - \mathbf{r}^H \mathbf{w}_f$$

$$\boxed{\begin{bmatrix} r(0) & \mathbf{r}^H \\ \mathbf{r} & \mathbf{R} \end{bmatrix} \begin{bmatrix} 1 \\ -\mathbf{w}_f \end{bmatrix} = \begin{bmatrix} P_M \\ \mathbf{0} \end{bmatrix}}$$

- Define the forward prediction-error filter vector

$$\mathbf{a}_M = \begin{bmatrix} 1 \\ -\mathbf{w}_f \end{bmatrix}$$

Augmented Wiener-Hopf Eqn.s of a forward prediction-error filter of order M.

- Then

$$\boxed{\mathbf{R}_{M+1}\mathbf{a}_M = \begin{bmatrix} P_M \\ \mathbf{0} \end{bmatrix}} \quad \text{or} \quad \boxed{\sum_{l=0}^{M} a_{M,l}\, r(l-i) = \begin{cases} P_M, & i = 0 \\ 0, & i = 1, 2, \cdots, M \end{cases}}$$

Backward linear prediction

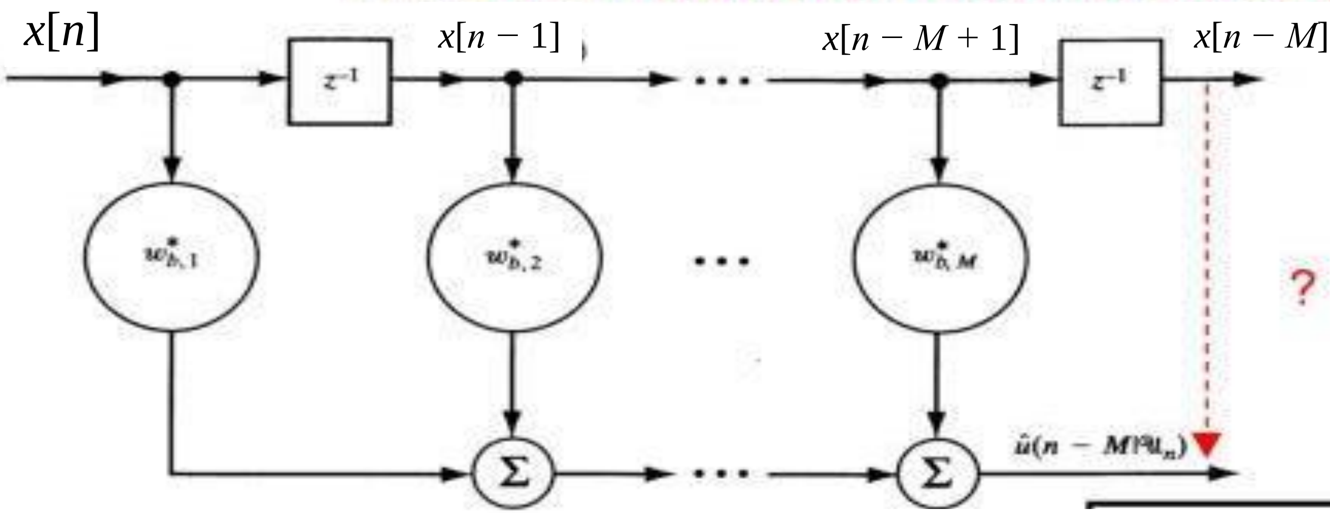
- Problem:
  - **Backward Prediction**
    - Observing the future

$$[x[n] \quad x[n-1] \quad \ldots \quad x[n-M+1]]$$

    - Predict the past

$$\hat{x}(n-M|\boldsymbol{X}_n)$$

    - i.e. find the predictor filter taps $w_{b,1}, w_{b,2},\ldots,w_{b,M}$



$$\hat{x}(n-M|\boldsymbol{X}_n) = \sum_{k=1}^{M} w^*_{b,opt,k}\, x[n-k+1]$$

Backward linear prediction

- Use Wiener filter theory to find the filter weights of the predictor.
- Input vector $\mathbf{x}[\mathrm{n}] = [x[n] \quad x[n-1] \quad \ldots \quad x[n-M+1]]$
- Desired output $d[n] = x[n-M]$

$$\mathbf{R}_{XX}\mathbf{w}_{b,opt} = \mathbf{r}^{B*}$$

- where

$$\mathbf{w}_{b,opt} = [w_{b,opt,1} \quad w_{b,opt,2} \quad \cdots \quad w_{b,opt,M}]^{\boldsymbol{T}}$$

$$\mathbf{r}^{B*} = [r(M) \quad r(M-1) \quad \ldots \quad r(1)]^T$$

Backward linear prediction

- Backward prediction error

$$b_M[n] = x[n-M] - \hat{x}(n-M|\boldsymbol{X}_n)$$
$$= x[n-M] - \sum_{k=1}^{M} w_{b,opt,k}^{*} x[n-k+1]$$

- Minimum mean-square prediction error (backward prediction error power)

$$P_M = r(0) - \mathbf{r}^{BT}\mathbf{w}_{b,opt}$$

Relation b/w forward and backward prediction

- Compare the Wiener-Hopf eqn.s for both cases (**R** and **r** are same)

$$
\begin{tikzcd}
\mathbf{R}\mathbf{w}_b = \mathbf{r}^{B*} \arrow[r, leftrightarrow, "?"] \arrow[d, "\text{order reversal}"'] & \mathbf{R}\mathbf{w}_f = \mathbf{r} \arrow[dd] \arrow[dr] & \\
\mathbf{R}^T\mathbf{w}_b^B = \mathbf{r}^* \arrow[d, "\text{complex conjugate}"'] & & \boxed{\mathbf{w}_b^{B*} = \mathbf{w}_f} \\
\mathbf{R}^H\mathbf{w}_b^{B*} = \mathbf{r} \arrow[r, Rightarrow, "\mathbf{R}^H = \mathbf{R}"] & \mathbf{R}\mathbf{w}_b^{B*} = \mathbf{r} \arrow[ur] &
\end{tikzcd}
$$

$$
P_M = r(0) - \mathbf{r}^{BT}\mathbf{w}_b \longrightarrow P_M = r(0) - \mathbf{r}^H\mathbf{w}_b^{B*} \longrightarrow P_M = r(0) - \mathbf{r}^H\mathbf{w}_f
$$

Backward prediction-error filter

- Input vector here: $\mathbf{x}[\mathrm{n}] = [x[n] \quad x[n-1] \quad \ldots \quad x[n-M]]$
- Desired output = prediction error

$$b_M[n] = x[n-M] - \hat{x}(n-M|\boldsymbol{X}_n)$$

$$= x[n-M] - \sum_{k=1}^{M} w_{b,opt,k}^{*} x[n-k+1]$$

Backward prediction-error filter

- Let

$$c_{M,k} = \begin{cases} -w_{b,k+1}, & k = 0, 1, \cdots, M-1 \\ 1, & k = M \end{cases}$$

- Then

$$b_M(n) = \sum_{k=0}^{M} c_{M,k}^* u(n-k)$$

but we found that

$$\mathbf{w}_b^{B*} = \mathbf{w}_f \longrightarrow \begin{array}{l} w_{b,M-k+1}^* = w_{f,k}, \ k = 1, 2, \cdots, M \\ \text{or} \\ w_{b,k} = w_{f,M-k+1}^*, \ k = 1, 2, \cdots, M \end{array}$$

$$c_{M,k} = \begin{cases} -w_{f,M-k}^*, & k = 0, 1, \cdots, M-1 \\ 1, & k = M \end{cases} = a_{M,M-k}^*, \ k = 0, 1, \cdots, M$$

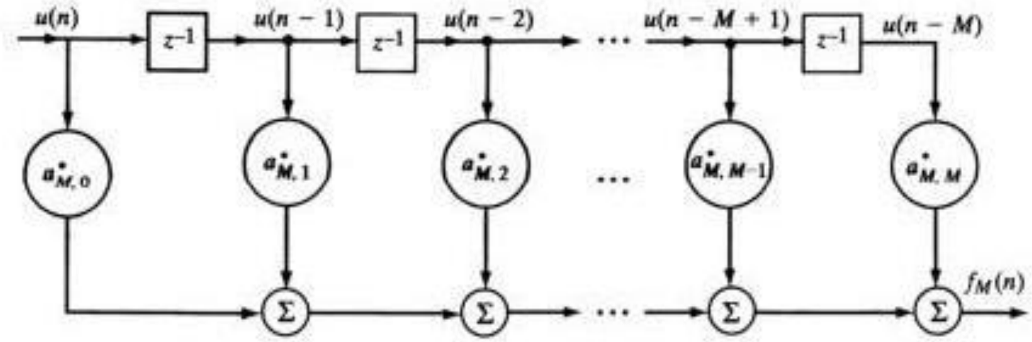
Then

$$\boxed{b_M(n) = \sum_{k=0}^{M} a_{M,M-k} u(n-k)}$$

Backward prediction-error filter

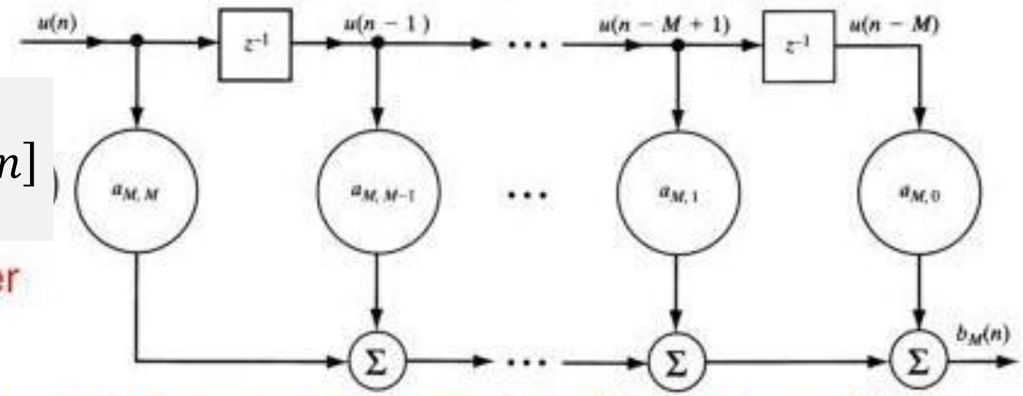
$$f_M[n] = \sum_{k=0}^{M} a_{M,k}^{*} x[n-k] = \mathbf{a}_M^H \mathbf{x}[n]$$

forward prediction-error filter



$$b_M[n] = \sum_{k=0}^{M} a_{M,M-k}^{*} x[n-k] = \mathbf{a}_M^{BH} \mathbf{x}[n]$$

backward prediction-error filter



- For stationary inputs, we may change a forward prediction-error filter into the corresponding backward prediction-error filter by reversing the order of the sequence and taking the complex conjugation of them.

Augmented W-H equations for backward prediction

- Let us combine the backward prediction filter and backward prediction-error power equations in a single matrix expression, i.e.
- $$\mathbf{R}\mathbf{w}_b = \mathbf{r}^{B*} \qquad P_M = r(0) - \mathbf{r}^{BT}\mathbf{w}_b$$

$$\boxed{\begin{bmatrix} \mathbf{R} & \mathbf{r}^{B*} \\ \mathbf{r}^{BT} & r(0) \end{bmatrix} \begin{bmatrix} -\mathbf{w}_b \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ P_M \end{bmatrix}}$$

- With the definition

$$\mathbf{a}_M^{B*} = \begin{bmatrix} -\mathbf{w}_b \\ 1 \end{bmatrix}$$

- Then

Augmented Wiener-Hopf Eqn.s of a backward prediction-error filter of order M.

- $$\boxed{\mathbf{R}_{M+1}\mathbf{a}_M^{B*} = \begin{bmatrix} \mathbf{0} \\ P_M \end{bmatrix}} \quad \boxed{\sum_{l=0}^{M} a_{M,M-l}^{*} r(l-i) = \begin{cases} 0, & i = 0, \cdots, M-1 \\ P_M, & i = M \end{cases}}$$

Levinson-Durbin algorithm

- Solve the following Wiener-Hopf eqn.s to find the predictor coef.s

$$\mathbf{R}\mathbf{w}_b = \mathbf{r}^{B*} \qquad \mathbf{R}\mathbf{w}_f = \mathbf{r}$$

- One-shot solution can have high computation complexity.
- Instead, use an (order)-recursive algorithm
  - Levinson-Durbin Algorithm.
  - Start with a first-order (m=1) predictor and at each iteration increase the order of the predictor by one up to (m=M).
  - Huge savings in computational complexity and storage.

Levinson-Durbin algorithm

- The tap-weights of the forward prediction filter may be order-updated as

$$\mathbf{a}_m = \begin{bmatrix} \mathbf{a}_{m-1} \\ 0 \end{bmatrix} + \kappa_m \begin{bmatrix} 0 \\ \mathbf{a}_{m-1}^{B*} \end{bmatrix}$$

- $a_{m,l} = a_{m-1,l} + \kappa_m a_{m-1,m-l}^{*}, \qquad l = 0.1, \dots, m$
- $a_{m,l}$ is the l-th tap-weight of the forward prediction error filter of order m, $a_{m-1,l}$ is the l-th tap-weight of the forward prediction error filter of order $m-1$, $a_{m-1,m-l}^{*}$ is the l-th tap-weight of the backward prediction error filter of order $m-1$.
- $a_{m-1,0} = 1$ and $a_{m-1,m} = 0$

Levinson-Durbin algorithm

- The tap-weights of the backward prediction filter may be order-updated as

$$\mathbf{a}_m^{B*} = \begin{bmatrix} 0 \\ \mathbf{a}_{m-1}^{B*} \end{bmatrix} + \kappa_m^* \begin{bmatrix} \mathbf{a}_{m-1} \\ 0 \end{bmatrix}$$

- $a_{m,m-l}^* = a_{m-1,m-l}^* + \kappa_m^* a_{m-1,l}, \qquad l = 0,1, \dots, m$
- $a_{m,m-l}^*$ is the l-th tap-weight of the backward prediction error filter of order *m*.
- Levinson-Durbin recursion is usually formulated for forward prediction error filter.

Levinson-Durbin algorithm

- Start with the relation bw. correlation matrix $\mathbf{R}_{m+1}$ and the forward-error prediction filter $\mathbf{a}_m$.

indicates order

$$\mathbf{R}_{m+1}\mathbf{a}_m = \begin{bmatrix} P_m \\ \mathbf{0}_m \end{bmatrix}$$

indicates dim. of matrix/vector

- We have seen how to partition the correlation matrix

$$\mathbf{R}_{m+1} = \begin{bmatrix} r(0) & \mathbf{r}_m^H \\ \mathbf{r}_m & \mathbf{R}_m \end{bmatrix} = \begin{bmatrix} \mathbf{R}_m & \mathbf{r}_m^{B*} \\ \mathbf{r}_m^{BT} & r(0) \end{bmatrix}$$

Levinson-Durbin algorithm

- Multiply the order-update eqn. by $\mathbf{R}_{m+1}$ from the left

$$\mathbf{R}_{m+1}\mathbf{a}_m = \underbrace{\mathbf{R}_{m+1}\begin{bmatrix}\mathbf{a}_{m-1}\\ 0\end{bmatrix}}_{1} + \kappa_m \underbrace{\mathbf{R}_{m+1}\begin{bmatrix}0\\ \mathbf{a}_{m-1}^{B*}\end{bmatrix}}_{2}$$

- Term 1:

$$\mathbf{R}_{m+1}\begin{bmatrix}\mathbf{a}_{m-1}\\ 0\end{bmatrix} = \begin{bmatrix}\mathbf{R}_m & \mathbf{r}_m^{B*}\\ \mathbf{r}_m^{BT} & r(0)\end{bmatrix}\begin{bmatrix}\mathbf{a}_{m-1}\\ 0\end{bmatrix} = \begin{bmatrix}\mathbf{R}_m\mathbf{a}_{m-1}\\ \mathbf{r}_m^{BT}\mathbf{a}_{m-1}\end{bmatrix}$$

but we know that (augmented Wiener-Hopf eqn.s)

$$\mathbf{R}_m\mathbf{a}_{m-1} = \begin{bmatrix}P_{m-1}\\ \mathbf{0}_{m-1}\end{bmatrix}$$

- Then

$$\boxed{\mathbf{R}_{m+1}\begin{bmatrix}\mathbf{a}_{m-1}\\ 0\end{bmatrix} = \begin{bmatrix}P_{m-1}\\ \mathbf{0}_{m-1}\\ \Delta_{m-1}\end{bmatrix}} \text{ where } \boxed{\Delta_{m-1} = \mathbf{r}_m^{BT}\mathbf{a}_{m-1}}$$

Levinson-Durbin algorithm

- <u>Term 2:</u>

$$\mathbf{R}_{m+1}\begin{bmatrix} 0 \\ \mathbf{a}_{m-1}^{B*} \end{bmatrix} = \begin{bmatrix} r(0) & \mathbf{r}_m^H \\ \mathbf{r}_m & \mathbf{R}_m \end{bmatrix}\begin{bmatrix} 0 \\ \mathbf{a}_{m-1}^{B*} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_m^H \mathbf{a}_{m-1}^{B*} \\ \mathbf{R}_m \mathbf{a}_{m-1}^{B*} \end{bmatrix}$$

but we know that (augmented Wiener-Hopf eqn.s)

$$\mathbf{R}_m \mathbf{a}_{m-1}^{B*} = \begin{bmatrix} \mathbf{0}_{m-1} \\ P_{m-1} \end{bmatrix}$$

- Then

$$\boxed{\mathbf{R}_{m+1}\begin{bmatrix} 0 \\ \mathbf{a}_{m-1}^{B*} \end{bmatrix} = \begin{bmatrix} \Delta_{m-1}^{*} \\ \mathbf{0}_{m-1} \\ P_{m-1} \end{bmatrix}} \text{ where } \boxed{\Delta_{m-1} = \mathbf{r}_m^{BT} \mathbf{a}_{m-1}}$$

Levinson-Durbin algorithm

$$\mathbf{R}_{m+1}\mathbf{a}_m = \mathbf{R}_{m+1}\begin{bmatrix} \mathbf{a}_{m-1} \\ 0 \end{bmatrix} + \kappa_m \mathbf{R}_{m+1}\begin{bmatrix} 0 \\ \mathbf{a}_{m-1}^{B*} \end{bmatrix}$$

$$\boxed{\begin{bmatrix} P_m \\ \mathbf{0}_m \end{bmatrix} = \begin{bmatrix} P_{m-1} \\ \mathbf{0}_{m-1} \\ \Delta_{m-1} \end{bmatrix} + \kappa_m \begin{bmatrix} \Delta_{m-1}^* \\ \mathbf{0}_{m-1} \\ P_{m-1} \end{bmatrix}}$$

- Then we have
  - from the first line

$$P_m = P_{m-1} + \kappa_m \Delta_{m-1}^*$$

  - from the last line

$$\kappa_m = -\frac{\Delta_{m-1}}{P_{m-1}}$$

$$P_m = P_{m-1}(1 - |\kappa_m|^2)$$

As iterations increase P_m decreases

$$P_0 = r(0)$$

$$0 \le P_m \le P_{m-1},\ m \ge 1$$

Levinson-Durbin algorithm

$$P_m = P_{m-1}(1 - |\kappa_m|^2) \longrightarrow \boxed{P_M = P_0 \prod_{m=1}^{M}(1 - |\kappa_m|^2)}$$

final value of the prediction error power

- κ_m: reflection coef.s due to the analogy with the reflection coef.s corresponding to the boundary bw. two sections in transmission lines

$$|\kappa_m| \le 1, \ \forall m \text{ and } \boxed{\kappa_m = a_{m,m}}$$

- The parameter Δ_m represents the crosscorrelation bw. the forward prediction error and the *delayed* backward prediction error

$$\Delta_{m-1} = E\{b_{m-1}(n-1) f_{m-1}^*(n)\}$$ HW: Prove this!

- Since $f_0[n] = b_0[n] = x[n]$

$$\Delta_0 = E\{b_0[n-1] f_0^*[n]\} = E\{x[n-1]x^*[n]\}$$
$$= r(-1) = r^*(1)$$

Steps of Levinson-Durbin algorithm

- Given: autocorrelation sequence $\{r(0), r(1), \dots, r(M)\}$
- Initialization: For m = 0, $\mathbf{a}_0 = a_{0,0} = 1,\ P_0 = r(0)$
- Start with m = 1
  - We readily have $a_{1,0} = 1$
  - Calculate $\Delta_0 = r(-1) \times a_{0,0} = r(-1)$
  - Calculate $\kappa_1 = -\frac{\Delta_0}{P_0} = -\frac{r(-1)}{r(0)}$; this also equals to $a_{1,1}$
  - So, we have $\mathbf{a}_1 = [a_{1,0} \quad a_{1,1}]^T = [1 \quad \kappa_1]^T$
  - Lastly, calculate $P_1 = P_0\,(1 - |\kappa_1|^2)$ for use in next step.

Steps of Levinson-Durbin algorithm

- Continue for $m = 2, 3, \ldots, M$
  - We readily have $a_{m,0} = 1$
  - Calculate $\Delta_{m-1} = \mathbf{r}_m^{BT}.\mathbf{a}_{m-1}$
  - Calculate $\kappa_m = -\frac{\Delta_{m-1}}{P_{m-1}}$; this also equals to $a_{m,m}$
  - Now compute in-between values of $\mathbf{a}_m$

$$a_{m,l} = a_{m-1,l} + \kappa_m a^*_{m-1,m-l}, \qquad l = 1, \ldots, m-1$$

  - Lastly, calculate $P_m = P_{m-1}(1 - |\kappa_1|^2)$ for use in next step.

Properties of prediction-error filters

- **Property 1:** There is a one-to-one correspondence bw. the two sets of quantities $\{P_0, \kappa_1, \kappa_2, \ldots, \kappa_M\}$ and $\{r(0), r(1), \ldots, r(M)\}$.
  - If one set is known the other can directly be computed by:
  - $\kappa_m = -\frac{1}{P_{m-1}} \sum_{k=0}^{m-1} a_{m-1,k}\, r(k-m)$
  - $r(m) = -\kappa_m^* P_{m-1} - \sum_{k=1}^{m-1} a_{m-1,k}^*\, r(m-k)$
- That means, if we are given one of the two sets of values, we may uniquely determine the other in a recursive manner.

Properties of prediction-error filters

- **Property 2a:** Transfer function of a forward prediction error filter $\{a_{m,k}^*\} \rightarrow H_{f,m}(z) = \sum_{k=0}^{m} a_{m,k}^* z^{-k}$

- Utilizing Levinson-Durbin recursion

$$
\begin{aligned}
H_{f,m}(z) &= \sum_{k=0}^{m} a_{m-1,k}^* z^{-k} + \kappa_m^* \sum_{k=0}^{m} a_{m-1,m-k} z^{-k} \\
&= \underline{\sum_{k=0}^{m-1} a_{m-1,k}^* z^{-k}} + \kappa_m^* \underline{\sum_{k=0}^{m-1} a_{m-1,m-1-k} z^{-k}} \times z^{-1}
\end{aligned}
$$

- but we also have

$$H_{f,m-1}(z) = \sum_{k=0}^{m-1} a_{m-1,k}^* \, z^{-k} \qquad H_{b,m-1}(z) = \sum_{k=0}^{m-1} a_{m-1,m-1-k} \, z^{-k}$$

- Then

$$\boxed{H_{f,m}(z) = H_{f,m-1}(z) + \kappa_m^* z^{-1} H_{b,m-1}(z)}$$

Properties of prediction-error filters

- **Property 2b:** Transfer function of a backward prediction error filter

$$\{a^*_{m,m-k}\} \rightarrow H_{b,m}(z) = \sum_{k=0}^{m} a^*_{m,m-k} z^{-k}$$

Utilizing Levinson-Durbin recursion $a^*_{m,m-l} = a^*_{m-1,m-l} + \kappa^*_m a_{m-1,l}$

$$\boxed{H_{b,m}(z) = z^{-1} H_{b,m-1}(z) + \kappa^*_m H_{f,m-1}(z)}$$

- Given the reflection coefficients κ_m and the transfer functions of the forward and backward prediction-error filters of order $m - 1$, we can uniquely determine the transfer function of the corresponding forward (and backward) prediction-error filter of order m.

Properties of prediction-error filters

- **Property 3:** Both the forward and backward prediction error filters have the same magnitude response

$$|H_{f,m}(z)| = |H_{b,m}(z)|,\ z = e^{j\omega}$$

- **Property 4:** Forward prediction-error filter is minimum-phase.
- **Property 5:** Backward prediction-error filter is maximum-phase.
- **Property 6**: Forward prediction-error filter is a whitening filter – a prediction-error filter is capable of whitening an input stationary discrete-time stochastic process, provided that the order of the filter is high enough.

Properties of prediction-error filters

- **Property 6**: The tap-weight vector of a forward prediction-error filter of order *M* and the resultant prediction-error power are uniquely defined by specifying the (*M* + 1) eigenvalues and the corresponding (*M* + 1) eigenvectors of the correlation matrix of the tap inputs of the filter.

$$\mathbf{a}_M = P_M \sum_{k=0}^{M} \left(\frac{q_{k,0}^*}{\lambda_k}\right) \mathbf{q}_k \quad \text{and} \quad P_M = \frac{1}{\sum_{k=0}^{M} |q_{k,0}|^2 \lambda_k^{-1}}$$

- where λ_k is the eigenvalue and $q_{k,0}^*$ is the first element of the *k*-th eigenvector $\mathbf{q}_k$ of the correlation matrix $\mathbf{R}_{M+1}$

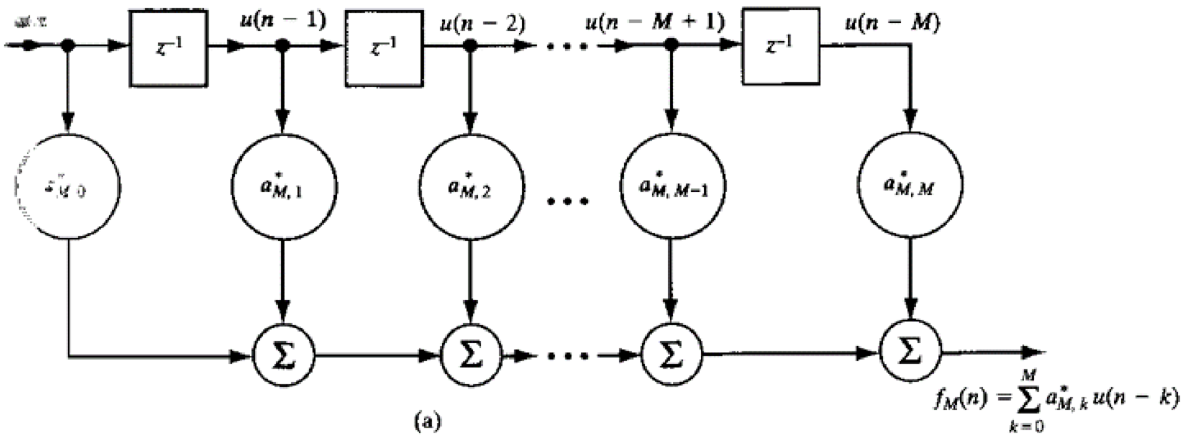
Properties of prediction-error filters

- **Property 7:** Backward prediction errors are orthogonal to each other.

$$\boxed{E\{b_m(n)b_i^*(n)\} = \begin{cases} P_m, & i = m \\ 0, & i \leq m \end{cases}}$$

AR modeling of stationary stochastic process

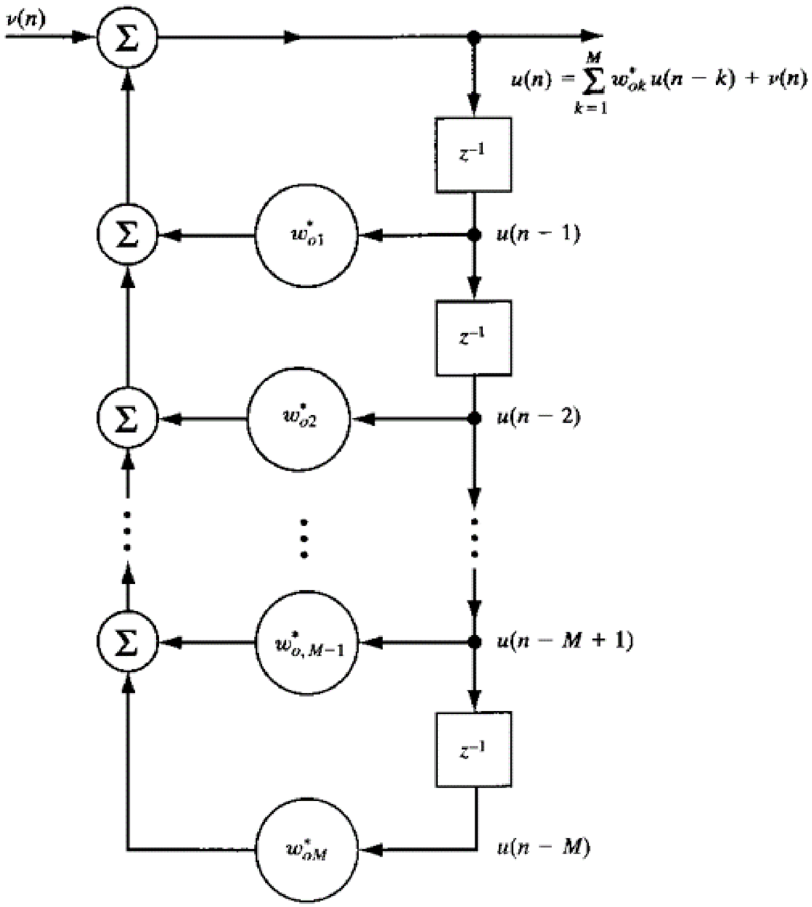
- **Analysis filter:** The input process $x[n]$ is whitened by choosing the filter order M sufficiently large so that the output prediction error process $f_M[n]$ consists of uncorrelated samples.
- It is an all-zero FIR filter.



(a)

AR modeling of stationary stochastic process

- **Synthesis filter:** The AR process $x[n]$ be generated by applying a white-noise process $v[n]$ of zero-mean and variance σ_v^2 to a filter whose parameters are set to the AR parameters $w_{opt,k}$, $k = 1, 2, \ldots, M$.
- It is an all-pole IIR filter.



Lattice Predictors

- A very efficient structure to implement the forward and backward predictors.
- Rewrite the prediction error filter coef.s

$$\mathbf{a}_m = \begin{bmatrix} \mathbf{a}_{m-1} \\ 0 \end{bmatrix} + \kappa_m \begin{bmatrix} 0 \\ \mathbf{a}_{m-1}^{B*} \end{bmatrix} \qquad \mathbf{a}_m^{B*} = \begin{bmatrix} 0 \\ \mathbf{a}_{m-1}^{B*} \end{bmatrix} + \kappa_m \begin{bmatrix} \mathbf{a}_{m-1} \\ 0 \end{bmatrix}$$

- The input signal to the predictors {u(n), n(n-1),...,u(n-M)} can be stacked into a vector

$$\mathbf{u}_{m+1}(n) = \left[\frac{\mathbf{u}_m(n)}{u(n-m)}\right] = \left[\frac{u(n)}{\mathbf{u}_m(n-1)}\right]$$

- Then the output of the predictors are

$$f_m(n) = \mathbf{a}_m^H \mathbf{u}_{m+1}(n) \qquad b_m(n) = \mathbf{a}_m^{B*} \mathbf{u}_{m+1}(n)$$

(forward) (backward)

Lattice Predictors

- Forward prediction-error filter

$$f_m(n) = \mathbf{a}_m^H \mathbf{u}_{m+1}(n) \Longleftarrow \mathbf{a}_m = \begin{bmatrix} \mathbf{a}_{m-1} \\ 0 \end{bmatrix} + \kappa_m \begin{bmatrix} 0 \\ \mathbf{a}_{m-1}^{B*} \end{bmatrix}$$

- First term

$$\begin{aligned} \begin{bmatrix} \mathbf{a}_{m-1}^H & | & 0 \end{bmatrix} \left[\frac{\mathbf{u}_m(n)}{u(n-m)} \right] &= \mathbf{a}_{m-1}^H \mathbf{u}_m(n) \\ &= f_{m-1}(n) \end{aligned}$$

- Second term

$$\begin{aligned} \begin{bmatrix} 0 & | & \mathbf{a}_{m-1}^{BT} \end{bmatrix} \left[\frac{u(n)}{\mathbf{u}_m(n-1)} \right] &= \mathbf{a}_{m-1}^{B*} \mathbf{u}_m(n-1) \\ &= b_{m-1}(n-1) \end{aligned}$$

- Combining both terms

$$f_m(n) = f_{m-1}(n) + \kappa_m^* b_{m-1}(n-1)$$

Lattice Predictors

- Similarly, Backward prediction-error filter

$$b_m(n) = \mathbf{a}_m^{BT}\mathbf{u}_{m+1}(n) \Longleftarrow \mathbf{a}_m^{B*} = \begin{bmatrix} 0 \\ \mathbf{a}_{m-1}^{B*} \end{bmatrix} + \kappa_m \begin{bmatrix} \mathbf{a}_{m-1} \\ 0 \end{bmatrix}$$

- First term

$$\begin{aligned} \begin{bmatrix} 0 \mid \mathbf{a}_{m-1}^{BT} \end{bmatrix} \left[\frac{u(n)}{\mathbf{u}_m(n-1)}\right] &= \mathbf{a}_{m-1}^{B*}\mathbf{u}_m(n-1) \\ &= b_{m-1}(n-1) \end{aligned}$$

- Second term

$$\begin{aligned} \begin{bmatrix} \mathbf{a}_{m-1}^{H} \mid 0 \end{bmatrix} \left[\frac{\mathbf{u}_m(n)}{u(n-m)}\right] &= \mathbf{a}_{m-1}^{H}\mathbf{u}_m(n) \\ &= f_{m-1}(n) \end{aligned}$$

- Combining both terms

$$b_m(n) = b_{m-1}(n-1) + \kappa_m f_{m-1}(n)$$

Lattice Predictors

- Forward and backward prediction-error filters

$$f_m(n) = f_{m-1}(n) + \kappa_m^* b_{m-1}(n-1)$$

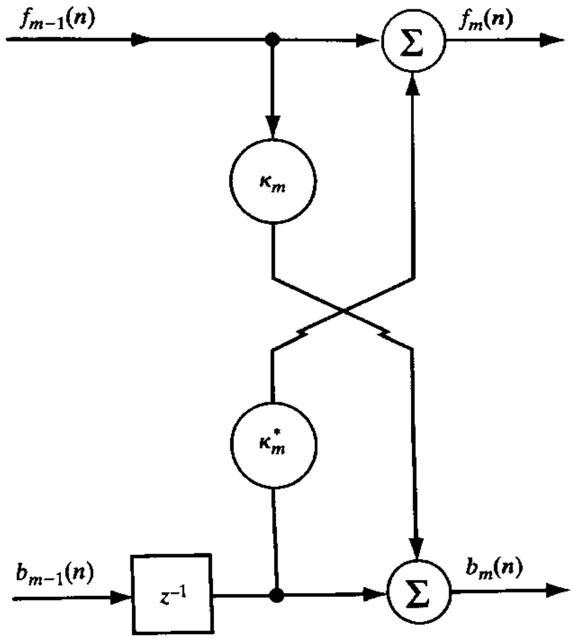
$$b_m(n) = b_{m-1}(n-1) + \kappa_m f_{m-1}(n)$$

in matrix form

$$\begin{bmatrix} f_m(n) \\ b_m(n) \end{bmatrix} = \begin{bmatrix} 1 & \kappa_m^* \\ \kappa_m & 1 \end{bmatrix} \begin{bmatrix} f_{m-1}(n) \\ b_{m-1}(n-1) \end{bmatrix}$$

and

$$b_{m-1}(n-1) = z^{-1} b_{m-1}(n)$$



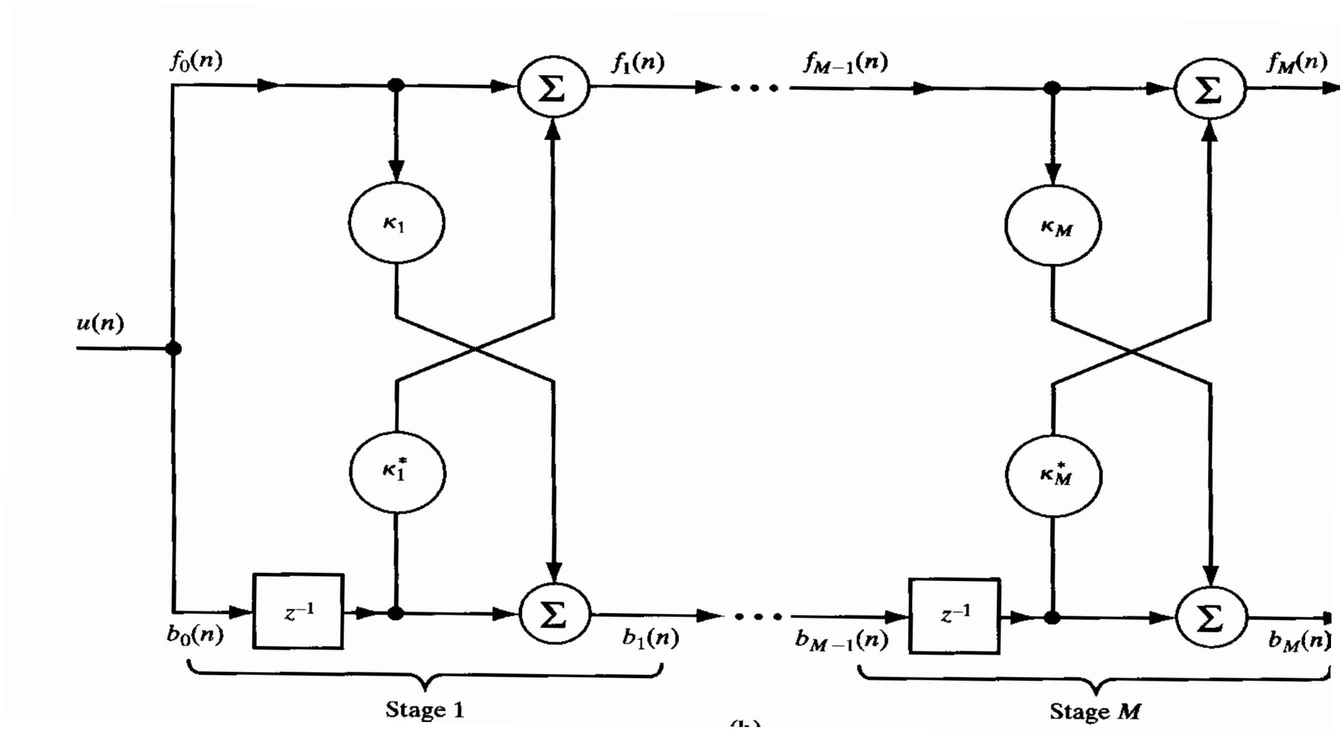
Last two equations define the *m*-th stage of the lattice predictor

Lattice Predictors

- For *m* = 0 we have $f_0(n) = b_0(n) = u(n)$,
- Hence for M stages

$$f_m(n) = f_{m-1}(n) + \kappa_m^* b_{m-1}(n-1)$$

$$b_m(n) = b_{m-1}(n-1) + \kappa_m f_{m-1}(n)$$



Lattice Predictors

- Highly efficient structure for generating sequence of forward prediction errors and corresponding sequence of backward prediction errors simultaneously.
- The various stages are decoupled from each other; the backward prediction errors produced at different stages are orthogonal to each other (property 7).
- Modular in structure – order can be easily increased.
- Similar structure in every stage – useful for VLSI implementation.