

Adaptive Nonlinear Equalization

Adaptive Decision-Feedback Equalizer

- DFE

$$\hat{I}_k = \sum_{j=-K_1}^0 c_j v_{k-j} + \sum_{j=1}^{K_2} c_j \tilde{I}_{k-j}$$

- Steepest-Descent Algorithm

$$\mathbf{C}_{k+1} = \mathbf{C}_k + \Delta \mathbf{E}(\varepsilon_k \mathbf{V}_k^*)$$

$$\varepsilon_k = I_k - \hat{I}_k \quad \mathbf{V}_k = [v_{k+K_1} \cdots v_k \quad I_{k-1} \cdots I_{k-K_2}]^t$$

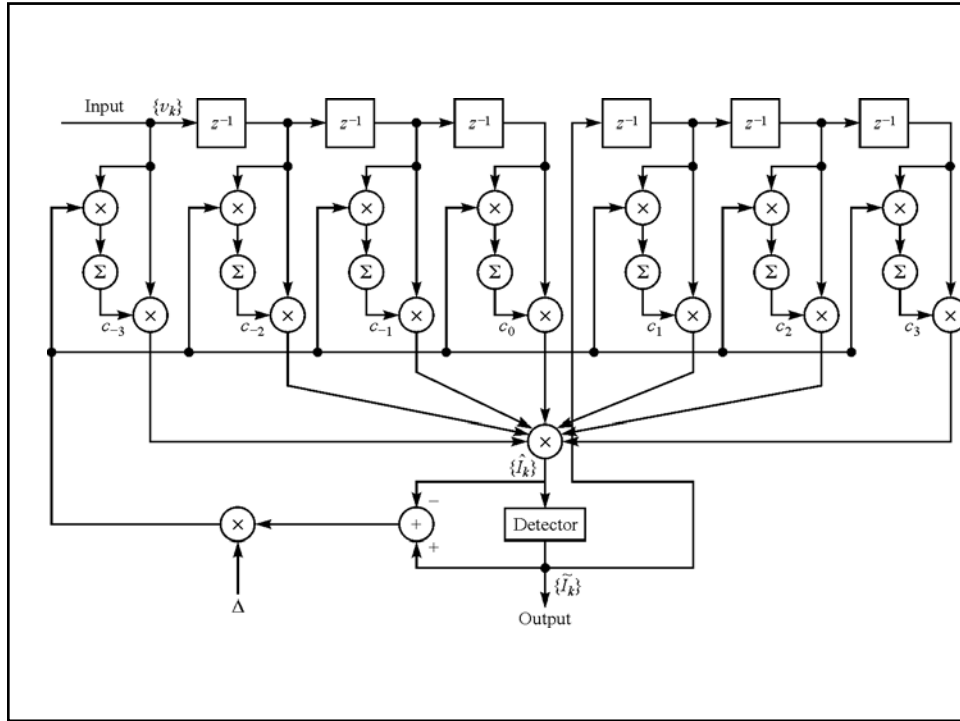
- LMS Algorithm (with estimated gradient, training process)

$$\hat{\mathbf{C}}_{k+1} = \hat{\mathbf{C}}_k + \Delta \varepsilon_k \mathbf{V}_k^*$$

- Adaptive process

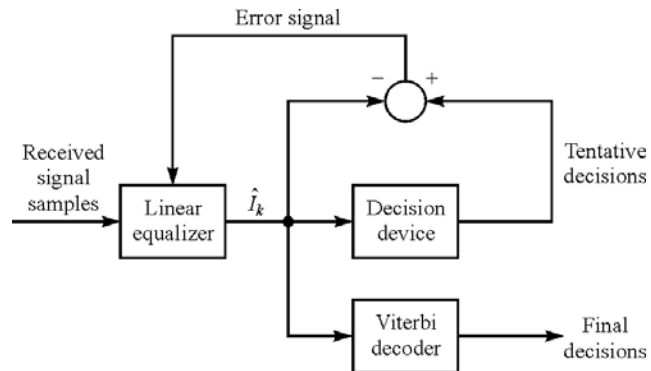
$$\tilde{\mathbf{C}}_{k+1} = \tilde{\mathbf{C}}_k + \Delta \tilde{\varepsilon}_k \mathbf{V}_k^*$$

$$\tilde{\varepsilon}_k = \tilde{I}_k - \hat{I}_k \quad \mathbf{V}_k = [v_{k+K_1} \cdots v_k \tilde{I}_{k-1} \cdots \tilde{I}_{k-K_2}]^t$$



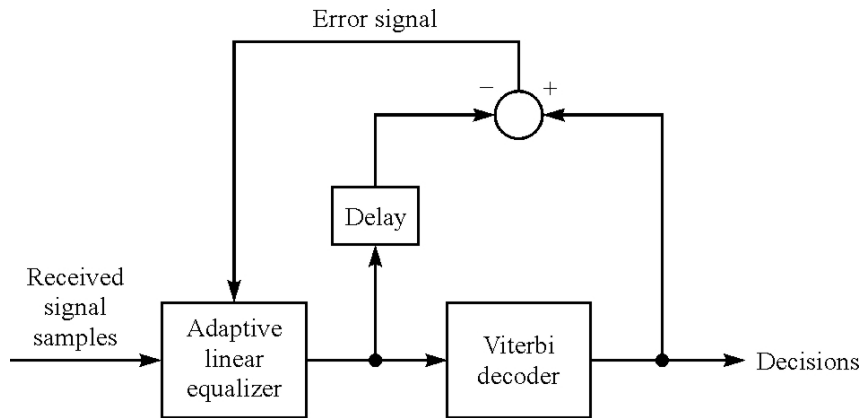
Trellis-coded signals (i.e. CPM signal)

- Decoded based on Viterbi algorithm, i.e., long delay
- **Solution #1: tentative decisions**



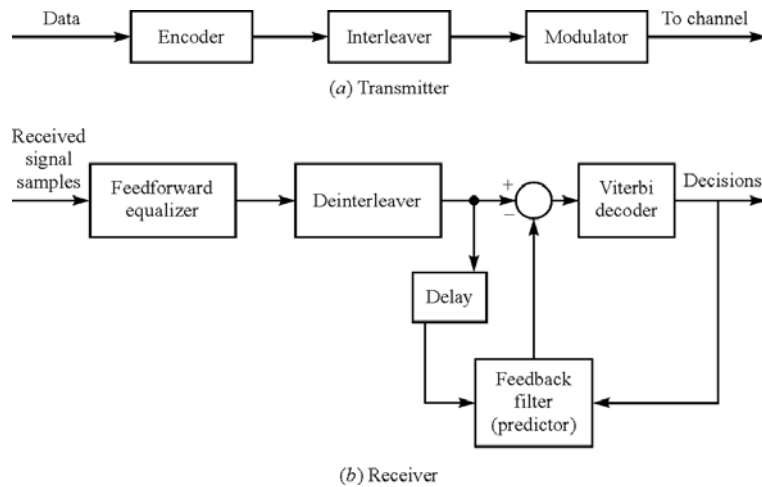
Trellis-coded signals (cont.)

- **Solution #2: Delay of samples** (time-invariant channel?)



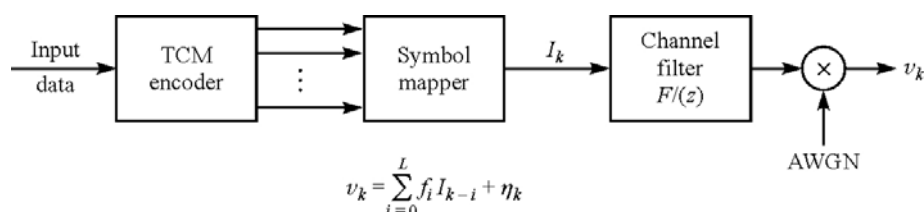
Trellis-coded signals (cont.)

- **Solution #3: Predictive DFE**



Reduced-State Viterbi Detection Algorithm

- Combined the trellis diagram of the encoder with the ISI
i.e., MLSE + Data Decoder
- Examples: Combined the trellis diagram of CPM + MLSE
or TCM + MLSE



Recursive Least-Squares (RLS) Algorithm

- LMS algorithm is slow, especially when $\lambda_{\max}/\lambda_{\min} \gg 1$
- LMS algorithm is for statistical average, i.e., long term time average.
- For fast convergence, we want to do time average, e.g.,

$$\mathcal{E}_N^{LS} = \sum_{n=0}^t w^{t-n} |e_N(n, t)|^2$$

$$\begin{aligned}
 \mathbf{Y}_N(t) &= [y(t) \quad y(t-1) \cdots y(t-N+1)]^t \\
 &= [v_{t+K_1} \cdots v_{t+1} \quad v_t \quad I_{t-1} \cdots I_{t-K_2}]^t
 \end{aligned}$$

Kalman algorithm

- Compute output:

$$\hat{I}(t) = \mathbf{Y}'_N(t)\mathbf{C}_N(t-1) = \sum_{j=0}^{N-1} c_j(t-1)y(t-j)$$

- Compute error:

$$e_N(t) = I(t) - \hat{I}(t)$$

- Compute Kalman gain vector:

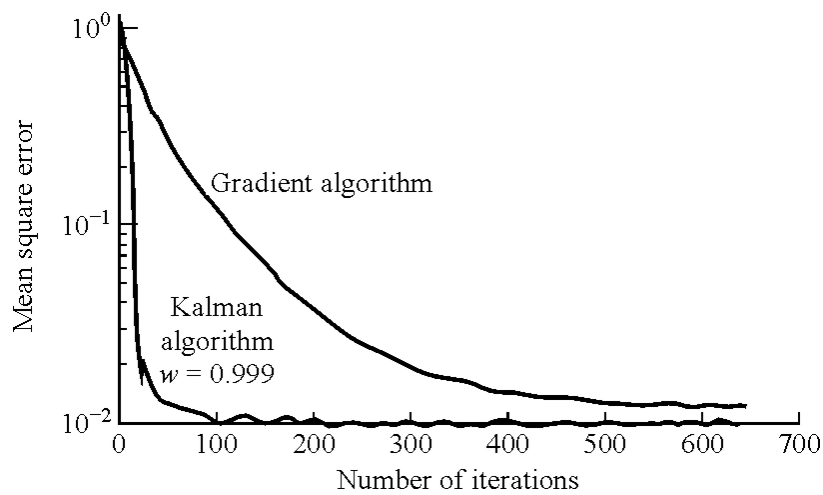
$$\mathbf{K}_N(t) = \frac{\mathbf{P}_N(t-1)\mathbf{Y}'_N(t)}{w + \mathbf{Y}'_N(t)\mathbf{P}_N(t-1)\mathbf{Y}_N^*(t)}$$

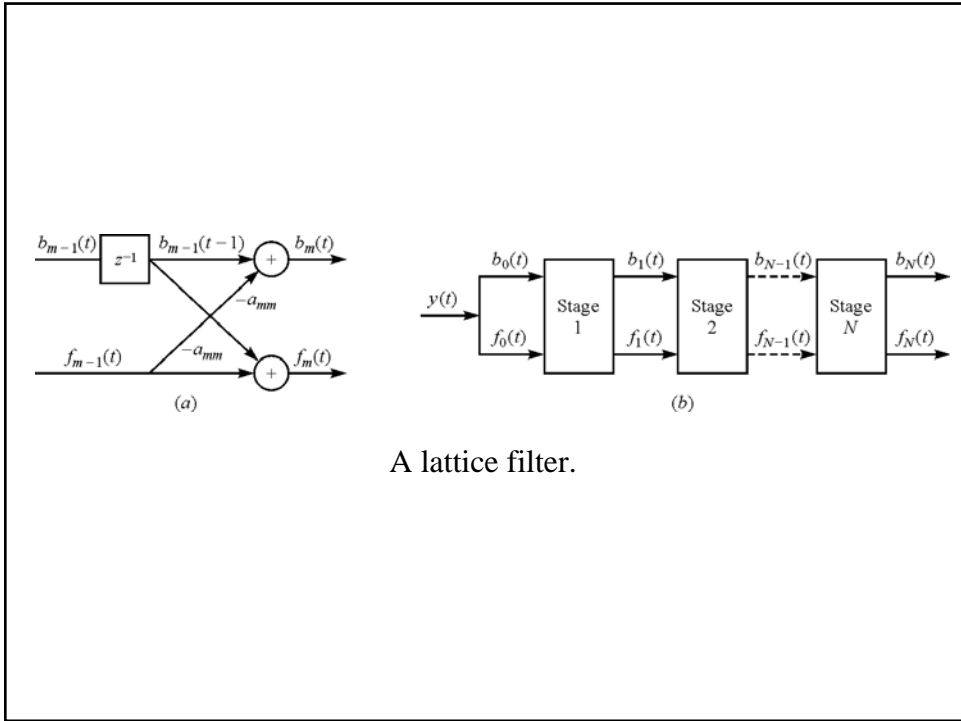
- Update inverse of the correlation matrix:

$$\mathbf{P}_N(t) = \frac{1}{w}[\mathbf{P}_N(t-1) - \mathbf{K}_N(t)\mathbf{Y}'_N(t)\mathbf{P}_N(t-1)]$$

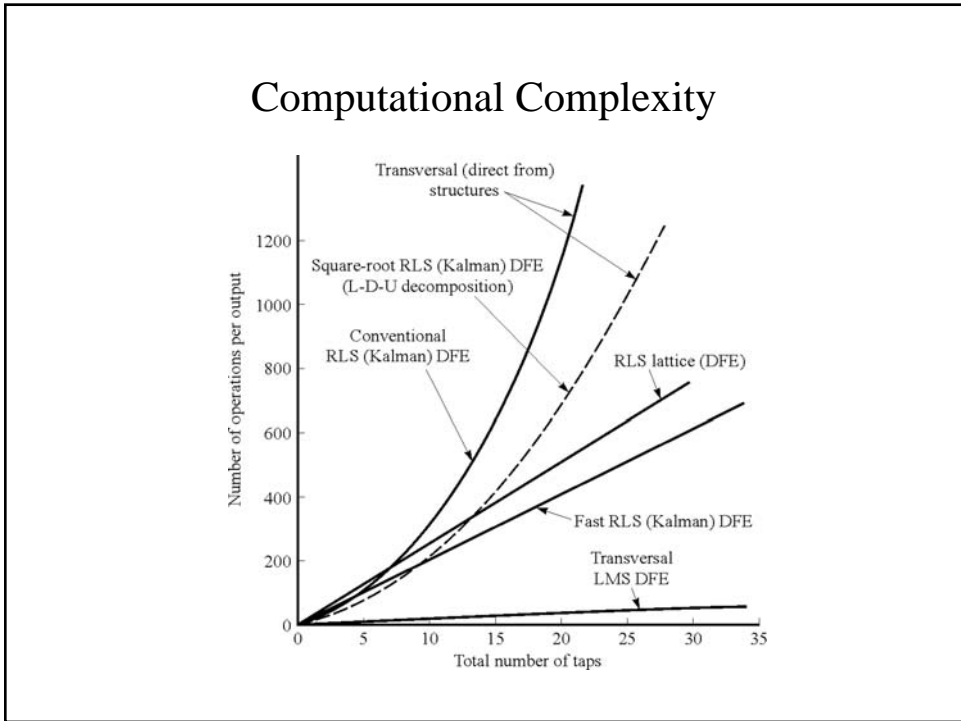
- Update coefficients:

$$\begin{aligned} \mathbf{C}_N(t) &= \mathbf{C}_N(t-1) + \mathbf{K}_N(t)e_N(t) \\ &= \mathbf{C}_N(t-1) + \mathbf{P}_N(t)\mathbf{Y}_N^*(t)e_N(t) \end{aligned}$$





A lattice filter.



Blind Equalization

- Equalization without training sequence
- Simple if you may an eye-opening.
- Three classes of techniques
 - Maximum-Likelihood Criterion
 - Stochastic Gradient Algorithm
 - Second- and Higher-Order Statistics