Statistical Digital Signal Processing

Chapter 6: Adaptive Filters

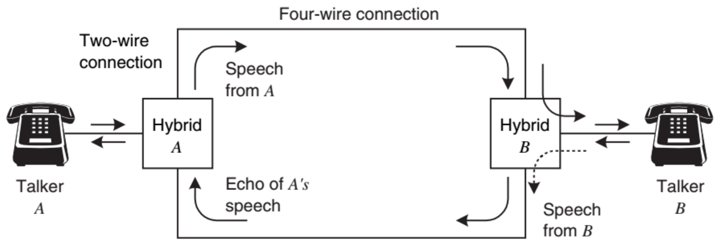
Bisrat Derebssa, SECE, AAiT, AAU

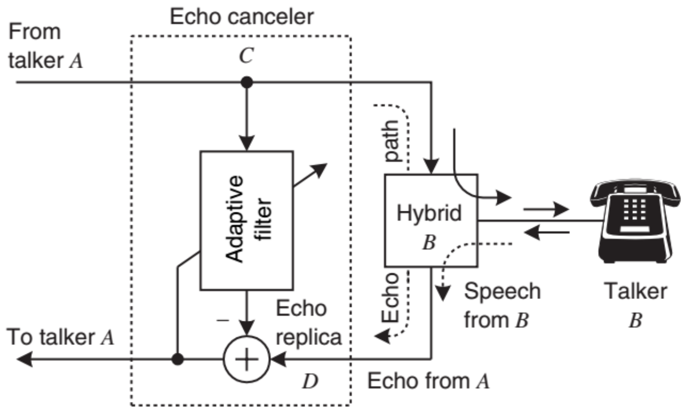
6.1. Introduction

- The theory of optimum filters was developed under the assumption that the filter designer has complete knowledge of the statistical properties of the SOE.
  - Not applicable in real-world applications.
- Adaptive filters can improve their performance, during normal operation, by learning the statistical characteristics through processing current signal observations.

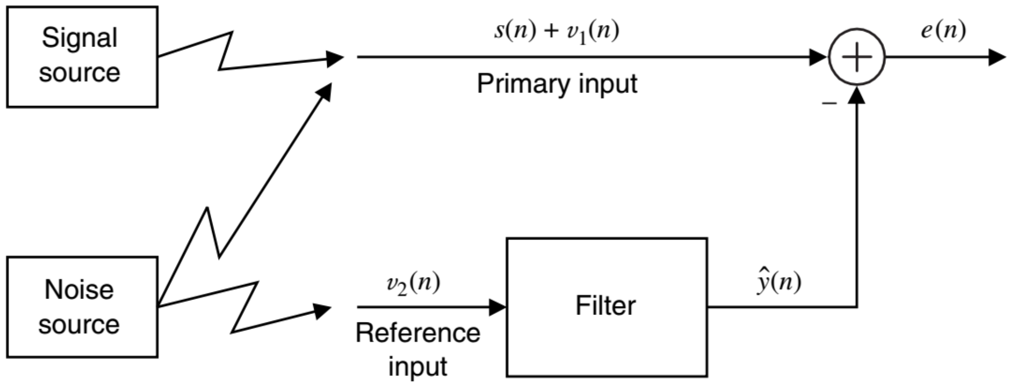
6.2. Applications of Adaptive Filters

- Echo Cancelation
  - Some energy on the incoming branch leaks into the outgoing branch and returns to the source as an echo.
    - Due to impedance mismatch.
  - Echo path is unknown and may be time changing.
  - The main task of the canceller is to estimate the echo signal with sufficient accuracy.





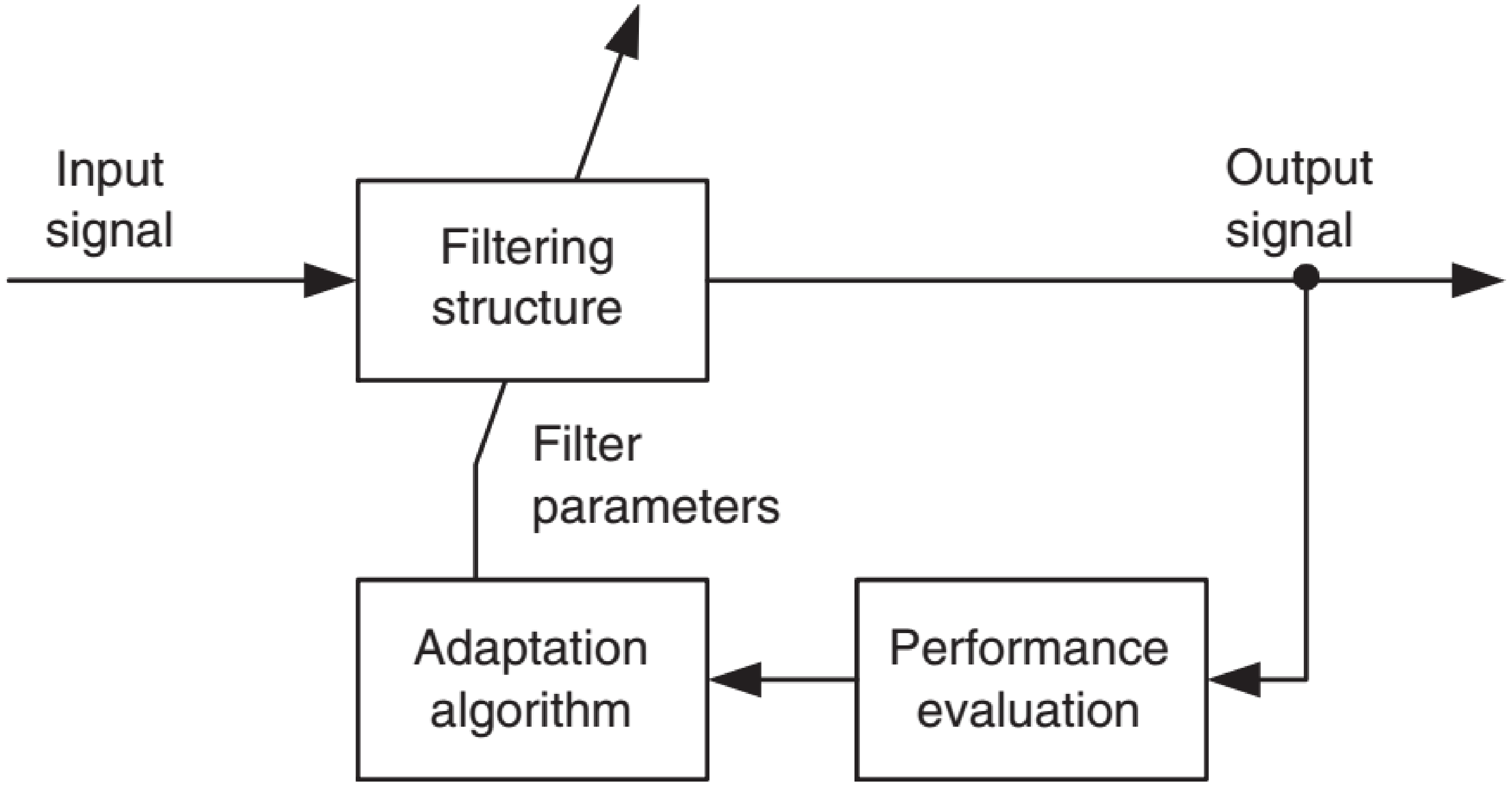
- Noise Cancelation
  - The signal of interest s(n) is corrupted by uncorrelated additive noise $v_1(n)$.
  - A second sensor, located at a different point, acquires a noise $v_2(n)$ (reference input) that is uncorrelated with the signal s(n) but correlated with the noise $v_1(n)$.



- Equalization of Data Communication Channels
  - Every pulse propagating through a channel suffers a certain amount of time dispersion because the frequency response of the channel deviates from the ideal one of constant magnitude and linear phase.
  - As a result, the tails of adjacent pulses interfere with the measurement of the current pulse (intersymbol interference).
  - The channel is unknown and possibly time-varying.
  - The goal of the equalizer is to restore the received pulse, as closely as possible, to its original shape.

6.3 Principles of Adaptive Filters

- Every adaptive filter consists of three modules



- Filtering Structure
  - Forms the output of the filter using measurements of the input signal or signals.
  - The filtering structure is linear if the output is obtained as a linear combination of the input measurements; otherwise it is said to be nonlinear.
  - The structure is fixed by the designer, and its parameters are adjusted by the adaptive algorithm.

- Criterion of performance (COP).
  - Assess the quality of the output of the adaptive filter and the desired response (when available) with respect to the requirements of the particular application.
  - The choice of COP is a compromise between what is acceptable to the user and what is mathematically tractable.
  - The square error is the most used COP.

- Adaptation algorithm.
  - Uses the value of the criterion of performance, or some function of it, and the measurements of the input and desired response (when available) to decide how to modify the parameters of the filter to improve its performance.
  - The complexity and the characteristics of the adaptive algorithm are functions of the filtering structure and the criterion of performance.

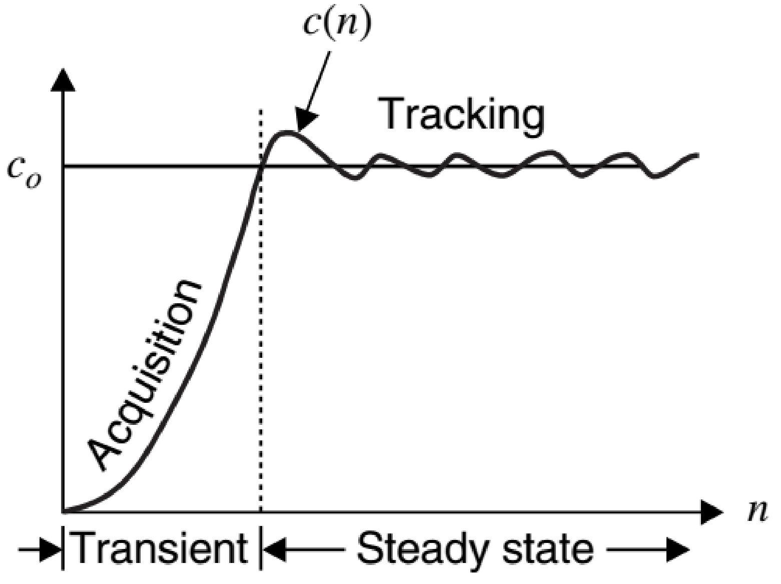
- A priori information about the signal operating environment (SOP) is used to choose
  - Criterion of performance
  - Filtering structure
- The design of an adaptive algorithm from the criterion performance is the most difficult step in the design and application of adaptive filters.

Modes of Operation

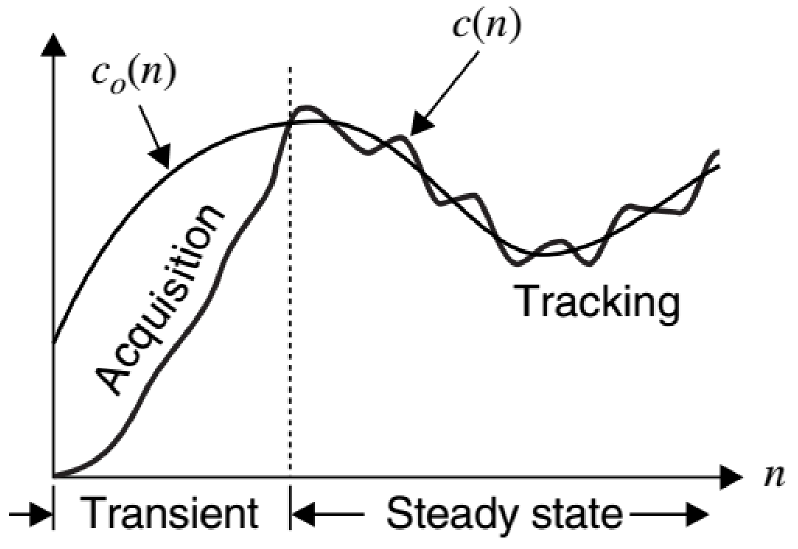
- Acquisition or convergence mode
  - Initial period until it gets reasonably close to its best performance.
- Tracking mode
  - When the SOE change with time, the filter has to follow the change in SOE.

Acquisition and tracking

- Learning curves of adaptive filters are represented by plots of MSD, MSE or misadjustment as a function of n.



(a) Stationary SOE



(b) Nonstationary SOE

Optimum vs. Adaptive Filters

- Optimum filters are a theoretical tool and cannot be used in practical applications because we do not know the statistical quantities (e.g., second-order moments) that are required for their design.

$$\mathbf{R}(n)\mathbf{c}_o(n) = \mathbf{d}(n)$$

$$\mathbf{R}(n) = E\{\mathbf{x}(n,\zeta)\mathbf{x}^H(n,\zeta)\}$$

$$\mathbf{d}(n) = E\{\mathbf{x}(n,\zeta)y^*(n,\zeta)\}$$

- During normal operations, the filter works with specific realizations of SOE

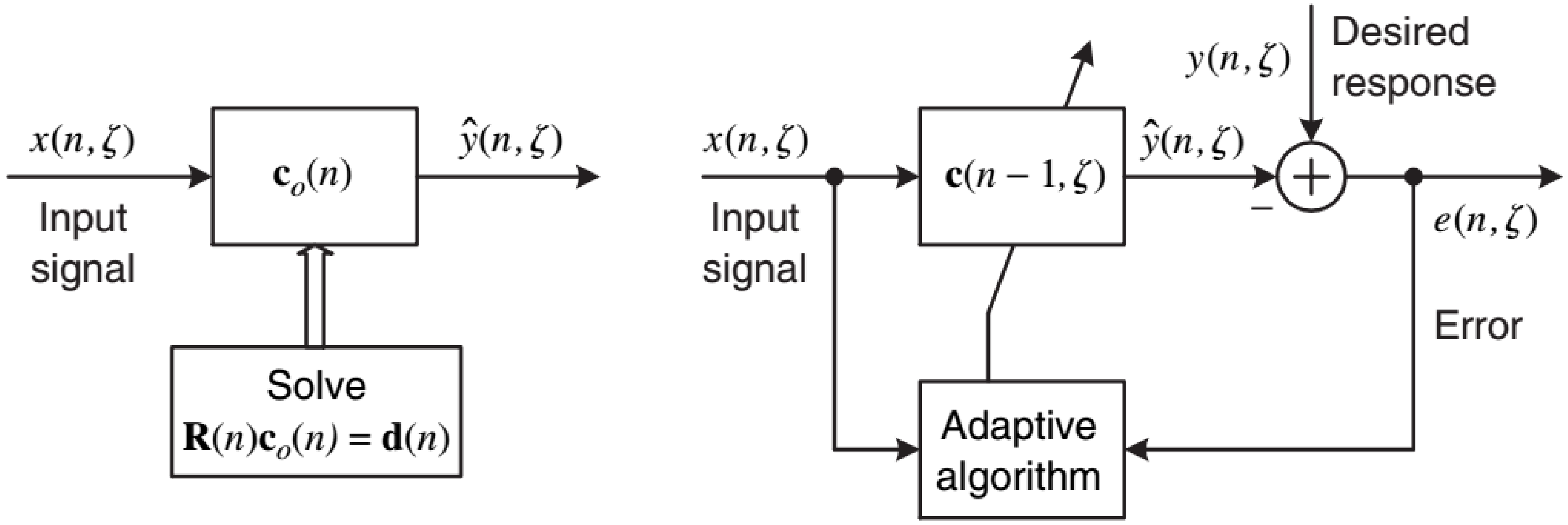
$$\hat{y}_o(n,\zeta) = \mathbf{c}_o^H(n)\mathbf{x}(n,\zeta)$$

$$\varepsilon_o(n,\zeta) = y(n,\zeta) - \hat{y}_o(n,\zeta)$$

- However, the filter is optimized with respect to its average performance across all possible realizations.
- The MMSE shows how well the filter performs on average.

$$P_o(n) = E\{|\varepsilon_o(n,\zeta)|^2\} = P_y(n) - \mathbf{d}^H(n)\mathbf{c}_o(n)$$

- For nonstationary environments, the optimum filter design is repeated every time instant n.
- If the second order moments are unknown, Adaptive filters are the best choice.



6.4 Block Adaptive Filtering

- If the SOE is ergodic, the second order moments can be calculated as

$$\mathbf{R} = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} \mathbf{x}(n,\zeta)\mathbf{x}^H(n,\zeta)$$

$$\mathbf{d} = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} \mathbf{x}(n,\zeta)y^*(n,\zeta)$$

- With sufficient data

$$\hat{\mathbf{R}}_N(\zeta) = \frac{1}{N} \sum_{n=0}^{N-1} \mathbf{x}(n,\zeta)\mathbf{x}^H(n,\zeta) \qquad \hat{\mathbf{d}}_N(\zeta) = \frac{1}{N} \sum_{n=0}^{N-1} \mathbf{x}(n,\zeta)y^*(n,\zeta)$$

- Then for data in interval $0 \le n \le N$

$$\hat{\mathbf{R}}_N(\zeta)\mathbf{c}_N(\zeta) = \hat{\mathbf{d}}_N(\zeta)$$

- The procedure is repeated every time the properties of SOE change *significantly.*
- However, block adaptive filters cannot track statistical variations with in the operating block.

6.5 Priori type adaptive algorithms

- From input data vector **x**(n), the desired response y(n) and the most recent coefficient vector **c**(n-1), the adaptive filter follows the following procedure.
  - Filtering

$$\hat{y}(n,\zeta) = \mathbf{c}^H(n-1,\zeta)\mathbf{x}(n,\zeta)$$

  - Error formation

$$e(n,\zeta) = y(n,\zeta) - \hat{y}(n,\zeta)$$

  - Adaptive algorithm

$$\mathbf{c}(n,\zeta) = \mathbf{c}(n-1,\zeta) + \mathbf{\Delta c}\{\mathbf{x}(n,\zeta), e(n,\zeta)\}$$

- The increment is selected to bring C close to C_o with the passage of time.

$$\|\mathbf{c}(n,\zeta)-\mathbf{c}_o\| < \delta, \text{for some } n > N_\delta$$

- An important requirement is that the increment should vanish if the error vanishes.
- Note that the equation does not need the calculation of **R** and **d**.

Stability

- Since the FIR filtering structure is always stable, the output or the error of the adaptive filter will be bounded if its coefficients are always kept close to the coefficients of the associated optimum filter.
- The presence of a feedback loop though the adaptive algorithm raises the issue of stability.
- For stationary SOE, where c_o is a constant, stability is guaranteed if

$$\lim_{n\to\infty} c_k(n,\zeta) = c_{o,k}(\zeta)$$

- Stability can also be defined in the mean square sense

$$\lim_{n\to\infty} E\{|c_k(n,\zeta) - c_{o,k}|^2\} = \lim_{n\to\infty} E\{|\tilde{c}_k(n,\zeta)|^2\} = 0$$

Performance measure

- The mean square deviation (MSD) measures the average distance between the coefficient vectors of the adaptive with the optimum.

$$\mathcal{D}(n) \triangleq E\{\|\mathbf{c}(n,\zeta) - \mathbf{c}_o(n)\|^2\} = E\{\|\tilde{\mathbf{c}}(n,\zeta)\|^2\}$$

- The excess MSE due to the deviation from c$_o$ is

$$P_{\text{ex}}(n) \triangleq P(n) - P_o(n)$$

$$P(n) \triangleq E\{|e(n,\zeta)|^2\}$$

$$P_o(n) \triangleq E\{|e_o(n,\zeta)|^2\}$$

$$e_o(n,\zeta) \triangleq y(n,\zeta) - \mathbf{c}_o^H(n-1)\mathbf{x}(n,\zeta)$$

- The misadjustment is given as

$$\mathcal{M}(n) \triangleq \frac{P_{\text{ex}}(n)}{P_o(n)} \qquad \text{or} \qquad \mathcal{M}'(n) \triangleq \frac{P'_{\text{ex}}(n)}{P'_o(n)}$$

- Reading assignment
  - Numerical stability,
  - Numerical accuracy

6.5 Steepest Descent Algorithm

- The error performance surface of an optimum filter in stationary SOE is given by

$$P(\mathbf{c}) = P_y - \mathbf{c}^H\mathbf{d} - \mathbf{d}^H\mathbf{c} + \mathbf{c}^H\mathbf{R}\mathbf{c}$$

- Since **R** is positive definite, the error surface is bowl-shaped.
- Iterative methods can be used to obtain the lowest point of this error surface.

- In iterative methods,
  - Starting point is chosen, usually the null vector **0**.
  - A search is started for the “bottom of the bowl” so that each step takes us to a lower point.
- Different iterative methods use different steps.

- If the error surface has a continuous derivative, by using Taylor expansion

$$P(\mathbf{c}+\boldsymbol{\Delta}\mathbf{c}) = P(\mathbf{c}) + \sum_{i=1}^{M} \frac{\partial P(\mathbf{c})}{\partial c_i}\Delta c_i + \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M}\Delta c_i \frac{\partial^2 P(\mathbf{c})}{\partial c_i \partial c_j}\Delta c_j + \cdots$$

$$P(\mathbf{c}+\boldsymbol{\Delta}\mathbf{c}) = P(\mathbf{c}) + (\boldsymbol{\Delta}\mathbf{c})^T \underset{\substack{\uparrow \\ \text{Gradient vector}}}{\nabla P(\mathbf{c})} + \tfrac{1}{2}(\boldsymbol{\Delta}\mathbf{c})^T[\underset{\substack{\uparrow \\ \text{Hessian matrix}}}{\nabla^2 P(\mathbf{c})}](\boldsymbol{\Delta}\mathbf{c}) + \cdots$$

- For quadratic function

$$\nabla P(\mathbf{c}) = 2(\mathbf{R}\mathbf{c} - \mathbf{d})$$
$$\nabla^2 P(\mathbf{c}) = 2\mathbf{R}$$

- The higher derivatives are zero.

- Note that

$$(\mathbf{\Delta c})^T [\nabla^2 P(\mathbf{c}_o)] (\mathbf{\Delta c}) > 0 \text{ for any nonzero } \mathbf{\Delta c}$$

- Therefore,

$$P(\mathbf{c} + \mathbf{\Delta c}) < P(\mathbf{c}) \quad \text{Only if} \quad (\mathbf{\Delta c})^T \nabla P(\mathbf{c}) < 0$$

- Since

$$(\mathbf{\Delta c})^T \nabla P(\mathbf{c}) = \|\mathbf{\Delta c}\| \|\nabla P(\mathbf{c})\| \cos\theta$$

- The reduction is maximum when

$$\mathbf{\Delta c} = -\nabla P(\mathbf{c})$$

- Therefore, the iterative minimization algorithm becomes

$$\mathbf{c}_k = \mathbf{c}_{k-1} + \mu[-\nabla P(\mathbf{c}_{k-1})]$$

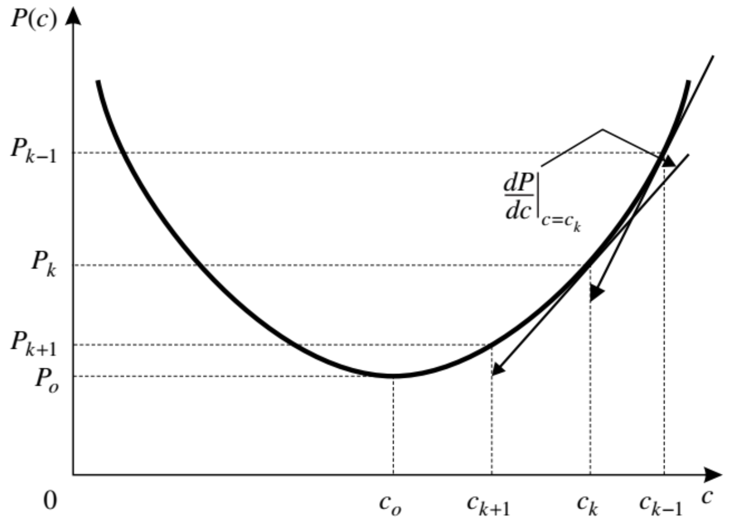
- Where
  - μ is the step-size parameter
- Inserting the value of the gradient

$$\mathbf{c}_k = \mathbf{c}_{k-1} + 2\mu(\mathbf{d} - \mathbf{R}\mathbf{c}_{k-1}) = (\mathbf{I} - 2\mu\mathbf{R})\mathbf{c}_{k-1} + 2\mu\mathbf{d}$$

- This iterative optimization can be combined with filtering

$$e(n,\zeta) = y(n,\zeta) - \mathbf{c}_{n-1}^{H}\mathbf{x}(n,\zeta)$$

$$\mathbf{c}_n = \mathbf{c}_{n-1} + 2\mu(\mathbf{d} - \mathbf{R}\mathbf{c}_{n-1})$$



Stability of SDA

- An algorithm is said to be stable if it converges to the minimum regardless of the starting point.
- Rewriting the SDA algorithm by using principal-component transformation, the necessary and sufficient condition for the convergence of SDA is

$$0 < \mu < \frac{1}{\lambda_{\max}}$$

Rate of convergence of SDA

- The rate (or speed) of convergence depends upon the algorithm and the nature of the performance surface.
- The most influential effect is inflicted by the condition number of the Hessian matrix that determines the shape of the contours of P (c).
- For quadratic surface

$$P(\mathbf{c}_k) \leq \left[\frac{\mathcal{X}(\mathbf{R}) - 1}{\mathcal{X}(\mathbf{R}) + 1}\right]^2 P(\mathbf{c}_{k-1}) \qquad \mathcal{X}(\mathbf{R}) = \lambda_{\max}/\lambda_{\min}$$

- The rate of convergence can be characterized by the time constant.
- If $\mu \ll 1,$ the time constant of $c_{k,l}$

$$\tau_i \simeq \frac{1}{2\mu\lambda_i}$$

- Therefore, the time constant of the SDA is

$$\tau \simeq 1/(\mu\lambda_{\min})$$

- Hence, the larger the eigenvalue spread of the input correlation matrix R, the longer it takes for the SDA to converge.

6.6 Newton's Type of Algorithms

- The basic idea of Newton's method is to achieve convergence in one step when P(c) is quadratic.
- Thus, if c_k is to be the minimum of P(c), the gradient $\nabla P(c_k)$ of P(c) evaluated at c_k should be zero.

$$\nabla P(\mathbf{c}_k) = \nabla P(\mathbf{c}_{k-1}) + \nabla^2 P(\mathbf{c}_{k-1}) \boldsymbol{\Delta} \mathbf{c}_k = \mathbf{0}$$

$$\boldsymbol{\Delta} \mathbf{c}_k = -[\nabla^2 P(\mathbf{c}_{k-1})]^{-1} \nabla P(\mathbf{c}_{k-1})$$

$$\mathbf{c}_k = \mathbf{c}_{k-1} - \mu [\nabla^2 P(\mathbf{c}_{k-1})]^{-1} \nabla P(\mathbf{c}_{k-1})$$

$$\mathbf{c}_k = \mathbf{c}_{k-1} - [\nabla^2 P(\mathbf{c}_{k-1})]^{-1} \nabla P(\mathbf{c}_{k-1}) = \mathbf{c}_{k-1} - (\mathbf{c}_{k-1} - \mathbf{R}^{-1}\mathbf{d}) = \mathbf{c}_o$$

- For the quadratic case, since $\nabla^2 P(\mathbf{c}_{k-1}) = 2\mathbf{R}$

$$\mathbf{c}_k = \mathbf{c}_{k-1} - \mu \mathbf{R}^{-1} \nabla P(\mathbf{c}_{k-1})$$

- Note that this requires the inversion of R.
  - Numerically intensive and can lead to numerically unstable solution.
- Modified Newton-type methods replace the Hessian matrix with another matrix which is guaranteed to be stable.
- Generally, the Newton-type methods provide faster convergence.

6.7 Least-Mean-Square Adaptive Filters

- In practice, only the input and the desired response are known.
  - Only estimate of the true or exact gradient can be calculated.
- Replacing the SDA iteration subscript k by time index n and replacing **R** and **d** by their instantaneous estimates $\mathbf{x}(n)\mathbf{x}^H(n)$ $\mathbf{x}(n)y^*(n)$

$$\nabla P(\mathbf{c}_{k-1}) = 2\mathbf{R}\mathbf{c}_{k-1} - 2\mathbf{d} \simeq \mathbf{2x}(n)\mathbf{x}^H(n)\mathbf{c}(n-1) - 2\mathbf{x}(n)y^*(n) = -2\mathbf{x}(n)e^*(n)$$

$$e(n) = y(n) - \mathbf{c}^H(n-1)\mathbf{x}(n)$$

- The coefficient adaptation algorithm becomes

$$\mathbf{c}(n) = \mathbf{c}(n-1) + 2\mu\mathbf{x}(n)e^*(n)$$

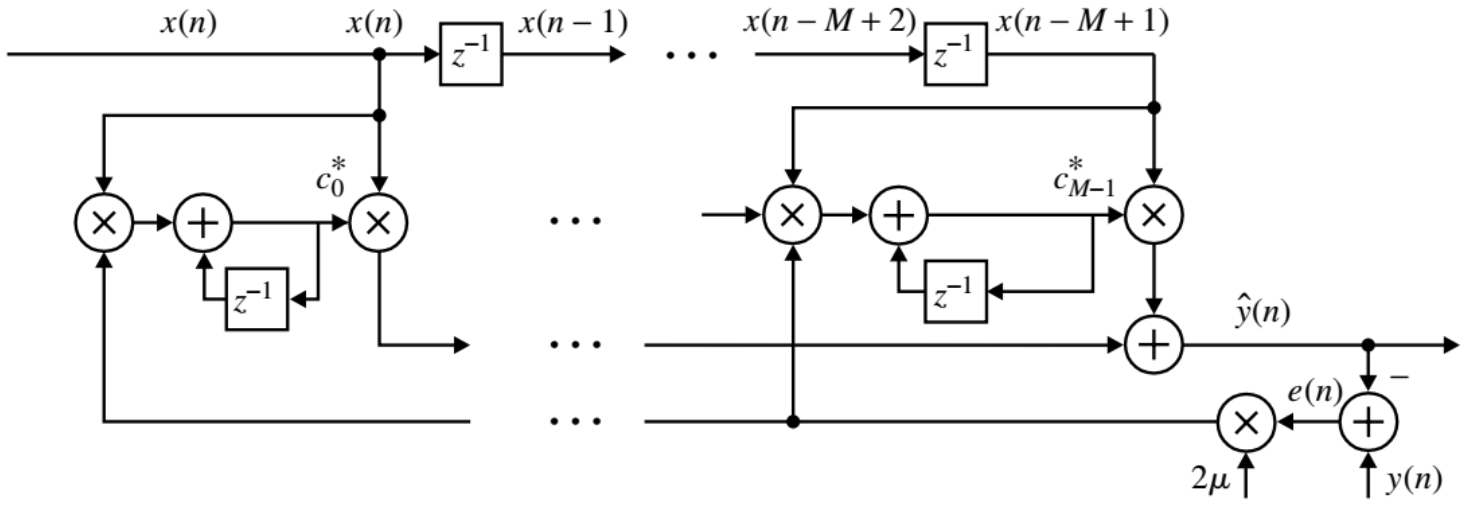
- Note that:
  - SDA contains deterministic components while LMS operates on random quantities.
  - SDA is not adaptive algorithm (it only depends on **R** and **d**). While LMS depends on the SOE.

- The LMS algorithm can be summarized as

$$\hat{y}(n) = \mathbf{c}^H(n-1)\mathbf{x}(n) \qquad \text{filtering}$$

$$e(n) = y(n) - \hat{y}(n) \qquad \text{error formation}$$

$$\mathbf{c}(n) = \mathbf{c}(n-1) + 2\mu\mathbf{x}(n)e^*(n) \qquad \text{coefficient updating}$$



Adaptation in a Stationary SOE

- In theory, the goal of the LMS adaptive filter is to identify the optimum filter coefficients c_o.

$$y(n) = \mathbf{c}_o^H \mathbf{x}(n) + e_o(n)$$

- By subtracting c_o from both sides of the update equation in LMS

$$\tilde{\mathbf{c}}(n) = \tilde{\mathbf{c}}(n-1) + 2\mu \mathbf{x}(n) e^*(n)$$

- By substituting the error formulation

$$\tilde{\mathbf{c}}(n) = [\mathbf{I} - 2\mu \mathbf{x}(n)\mathbf{x}^H(n)]\tilde{\mathbf{c}}(n-1) + 2\mu \mathbf{x}(n) e_o^*(n)$$

- The irreducible error $e_o(n)$ accounts for measurement noise, modeling errors, unmodelled dynamics, quantization effects, and other disturbances.
- The presence of $e_o(n)$ prevents convergence because it forces c˜(n) to fluctuate around zero.
- |c ˜(n)| is bounded in mean square if E{c ˜(n)} → 0 as n → ∞ and var{˜c_k(n)} is bounded for all n.
- It can be shown that c˜(n) converges if the eigenvalues of the system matrix (I – 2μR) are less than 1.

$$0 < 2\mu < \frac{1}{\lambda_{\max}}$$

Summary of the LMS algorithm.

Design parameters

$\mathbf{x}(n)$ = input data vector at time n

$y(n)$ = desired response at time n

$\mathbf{c}(n)$ = filter coefficient vector at time n

M = number of coefficients

μ = step-size parameter

$$0 < \mu \ll \frac{1}{\sum_{k=1}^{M} E\{|x_k(n)|^2\}}$$

Initialization

$\mathbf{c}(-1) = \mathbf{x}(-1) = \mathbf{0}$

Computation

For $n = 0, 1, 2, \ldots,$ compute

$$\hat{y}(n) = \mathbf{c}^H(n-1)\,\mathbf{x}(n)$$
$$e(n) = y(n) - \hat{y}(n)$$
$$\mathbf{c}(n) = \mathbf{c}(n-1) + 2\mu\mathbf{x}(n)e^*(n)$$

- Reading Assignment
  - Stability
  - Rate of convergence
  - Steady-state excess MSE

6.8 Least Square Adaptive Filters

- LS adaptive filters are designed so that the updating of their coefficients always attains the minimization of the total squared error from the time the filter initiated operation up to the current time.
- Therefore, the filter coefficients at time index n are chosen to minimize the cost function

$$E(n) = \sum_{j=0}^{n} \lambda^{n-j} |e(j)|^2 = \sum_{j=0}^{n} \lambda^{n-j} |y(j) - \mathbf{c}^H \mathbf{x}(j)|^2$$

- λ is the forgetting factor.
  - $0 < \lambda < 1$

- The filter coefficients that minimize the above error are given by

$$\hat{\mathbf{R}}(n)\mathbf{c}(n) = \hat{\mathbf{d}}(n)$$

$$\hat{\mathbf{R}}(n) \triangleq \sum_{j=0}^{n} \lambda^{n-j} \mathbf{x}(j)\mathbf{x}^H(j)$$

$$\hat{\mathbf{d}}(n) \triangleq \sum_{j=0}^{n} \lambda^{n-j} \mathbf{x}(j) y^*(j)$$

This step is repeated every index n+1, n=2, . . .

- The error is then

$$E_{\min}(n) = E_y(n) - \hat{\mathbf{d}}^H(n)\mathbf{c}(n)$$

$$E_y(n) \triangleq \sum_{j=0}^{n} \lambda^{n-j} |y(j)|^2$$

- The autocorrelation can be obtained recursively

$$\hat{\mathbf{R}}(n) = \lambda\hat{\mathbf{R}}(n-1) + \mathbf{x}(n)\mathbf{x}^H(n)$$
$$\hat{\mathbf{d}}(n) = \lambda\hat{\mathbf{d}}(n-1) + \mathbf{x}(n)y^*(n)$$

- The “new” correlation matrix can be updated by weighting the “old” correlation matrix with the forgetting factor λ and then incorporating the “new information”.

A priori adaptive LS algorithm

- Substituting the recursive autocorrelation values into the normal equation

$$[\hat{\mathbf{R}}(n) - \mathbf{x}(n)\mathbf{x}^H(n)]\mathbf{c}(n-1) = \hat{\mathbf{d}}(n) - \mathbf{x}(n)y^*(n)$$

$$\hat{\mathbf{R}}(n)\mathbf{c}(n-1) + \mathbf{x}(n)e^*(n) = \hat{\mathbf{d}}(n)$$

$$e(n) = y(n) - \mathbf{c}^H(n-1)\mathbf{x}(n)$$

- If R(n) is invertible

$$\mathbf{c}(n-1) + \hat{\mathbf{R}}^{-1}(n)\mathbf{x}(n)e^*(n) = \hat{\mathbf{R}}^{-1}(n)\hat{\mathbf{d}}(n) = \mathbf{c}(n)$$

- If we define an adaptation gain vector g(n) as

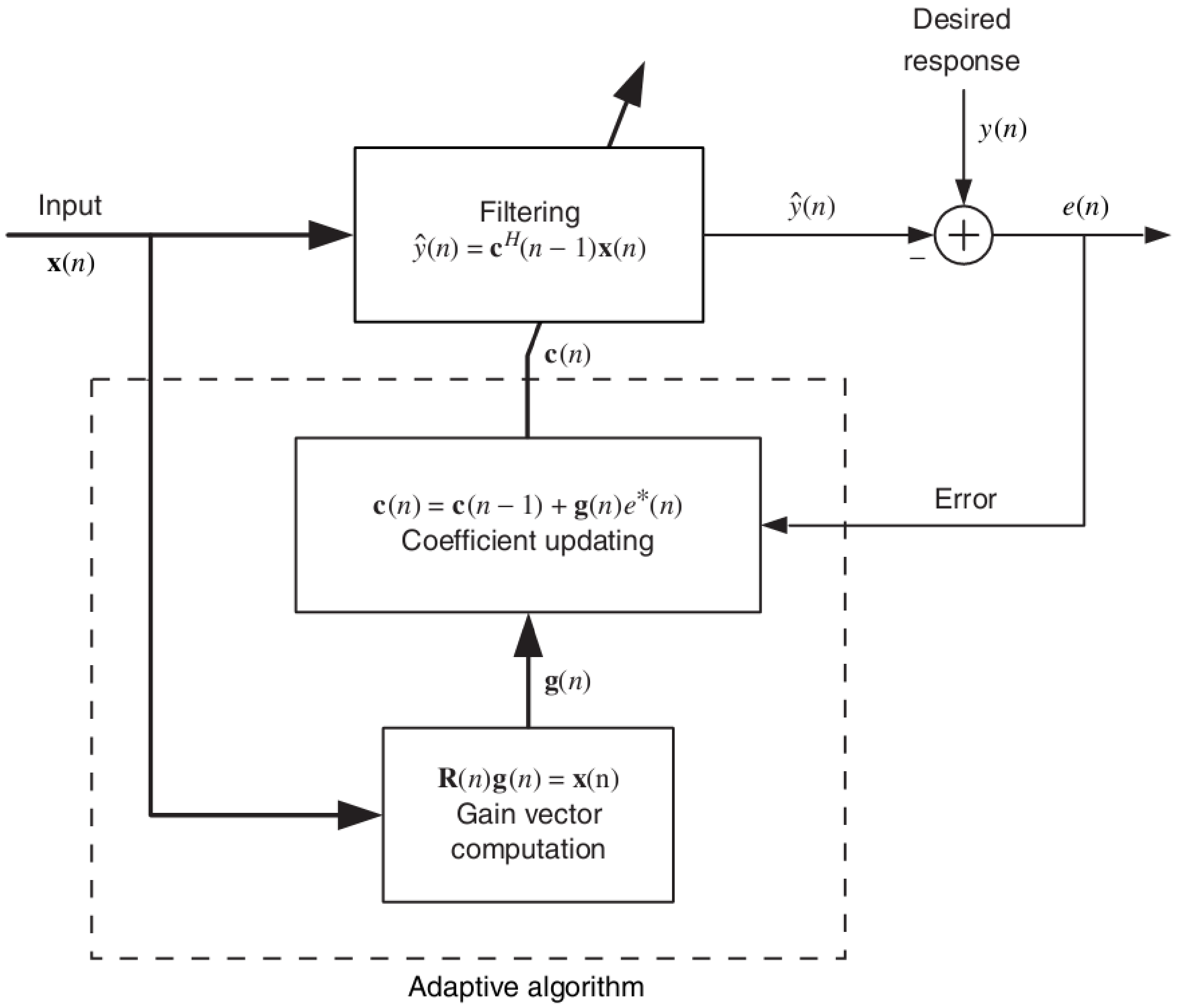
$$\hat{\mathbf{R}}(n)\mathbf{g}(n) \triangleq \mathbf{x}(n)$$

- Then the new coefficients are given as

$$\mathbf{c}(n) = \mathbf{c}(n-1) + \mathbf{g}(n)e^*(n)$$

- Defining a conversion factor

$$\alpha(n) \triangleq 1 - \mathbf{x}^H(n)\hat{\mathbf{R}}^{-1}(n)\mathbf{x}(n) = 1 - \mathbf{x}^H(n)\mathbf{g}(n)$$


Desired response
$y(n)$
Input
$\mathbf{x}(n)$
Filtering
$\hat{y}(n) = \mathbf{c}^H(n-1)\mathbf{x}(n)$
$\hat{y}(n)$
$-$
$e(n)$
$\mathbf{c}(n)$
$\mathbf{c}(n) = \mathbf{c}(n-1) + \mathbf{g}(n)e^*(n)$
Coefficient updating
Error
$\mathbf{g}(n)$
$\mathbf{R}(n)\mathbf{g}(n) = \mathbf{x}(\mathrm{n})$
Gain vector computation
Adaptive algorithm

- Reading Assignments
  - Posteriori Adaptive LS algorithms

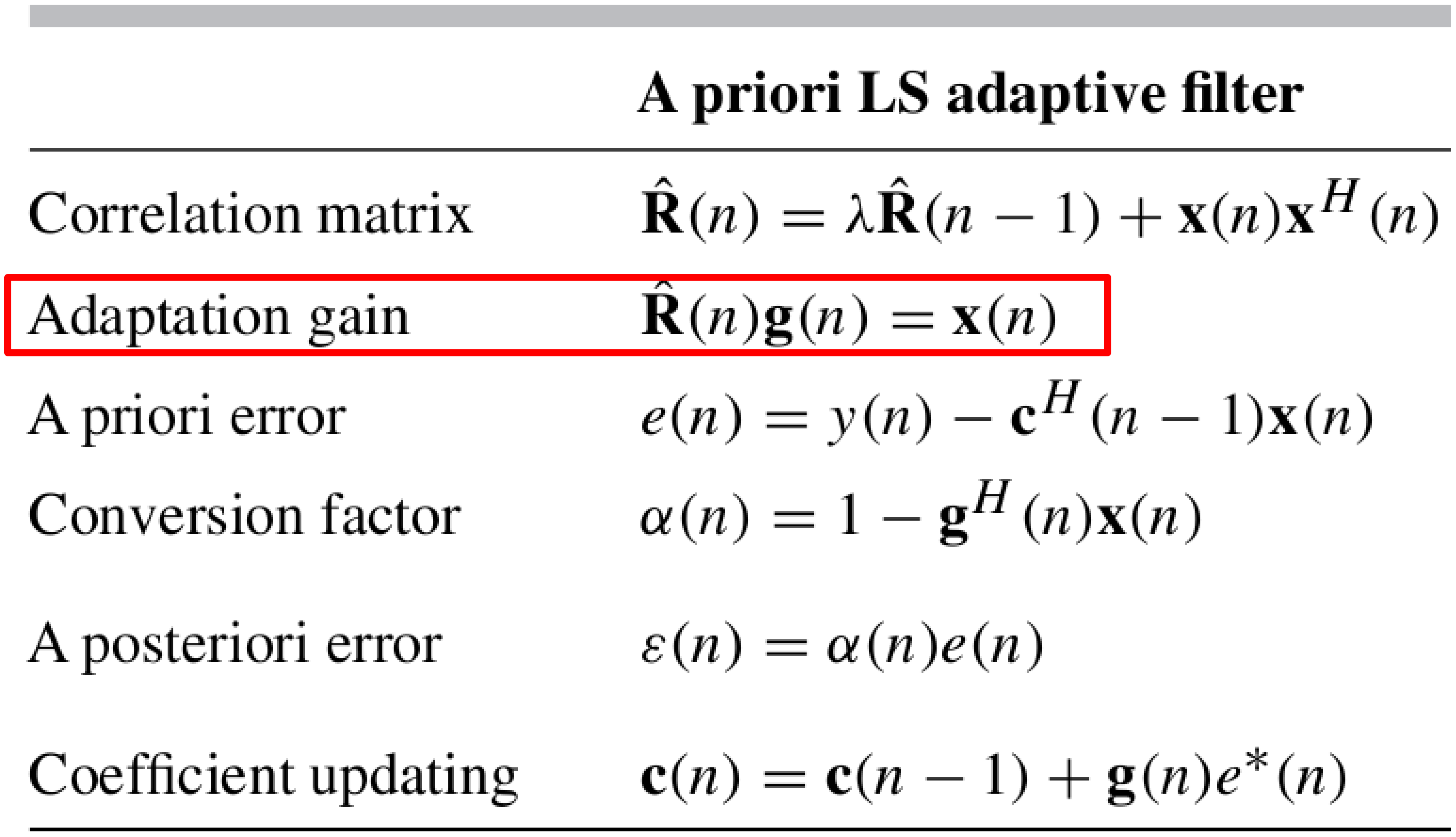
$$\lambda\hat{\mathbf{R}}(n-1)\mathbf{c}(n) - \mathbf{x}(n)\varepsilon^*(n) = \lambda\hat{\mathbf{d}}(n-1)$$

$$\varepsilon(n) = y(n) - \mathbf{c}^H(n)\mathbf{x}(n)$$

$$\mathbf{c}(n) - \lambda^{-1}\hat{\mathbf{R}}^{-1}(n-1)\mathbf{x}(n)\varepsilon^*(n) = \hat{\mathbf{R}}^{-1}(n-1)\hat{\mathbf{d}}(n-1) = \mathbf{c}(n-1$$

$$\mathbf{c}(n) = \mathbf{c}(n-1) + \bar{\mathbf{g}}(n)\varepsilon^*(n)$$

$$\lambda\hat{\mathbf{R}}(n-1)\bar{\mathbf{g}}(n) \triangleq \mathbf{x}(n)$$



	A priori LS adaptive filter
Correlation matrix	$\hat{\mathbf{R}}(n) = \lambda\hat{\mathbf{R}}(n-1) + \mathbf{x}(n)\mathbf{x}^H(n)$
Adaptation gain	$\hat{\mathbf{R}}(n)\mathbf{g}(n) = \mathbf{x}(n)$
A priori error	$e(n) = y(n) - \mathbf{c}^H(n-1)\mathbf{x}(n)$
Conversion factor	$\alpha(n) = 1 - \mathbf{g}^H(n)\mathbf{x}(n)$
A posteriori error	$\varepsilon(n) = \alpha(n)e(n)$
Coefficient updating	$\mathbf{c}(n) = \mathbf{c}(n-1) + \mathbf{g}(n)e^*(n)$

Recursive LS adaptive filters

- The major computational load in LS adaptive filters, that is, the computation of the gain vectors can be reduced if we can find a recursive formula to update the inverse

$$\mathbf{P}(n) \triangleq \hat{\mathbf{R}}^{-1}(n)$$

- By using the rank 1 updating and matrix inversion lemma,

$$(\lambda\mathbf{R} + \mathbf{x}\mathbf{x}^H)^{-1} = \lambda^{-1}\mathbf{R}^{-1} - \frac{(\lambda^{-1}\mathbf{R}^{-1}\mathbf{x})(\lambda^{-1}\mathbf{R}^{-1}\mathbf{x})^H}{1 + \lambda^{-1}\mathbf{x}^H\mathbf{R}^{-1}\mathbf{x}}$$

- The prior and posterior adaptation gains and the inverse are obtained as

$$\bar{\mathbf{g}}(n) = \lambda^{-1}\mathbf{P}(n-1)\mathbf{x}(n)$$

$$\alpha(n) = 1 + \bar{\mathbf{g}}^H(n)\mathbf{x}(n)$$

$$\mathbf{g}(n) = \frac{\bar{\mathbf{g}}(n)}{\bar{\alpha}(n)}$$

$$\mathbf{P}(n) = \lambda^{-1}\mathbf{P}(n-1) - \mathbf{g}(n)\bar{\mathbf{g}}^H(n)$$

- Similarly a recursive updating formula for the minimum error is given as

$$E_{\min}(n) = \lambda E_{\min}(n-1) + \frac{|e(n)|^2}{\alpha(n)}$$

Initialization

$\mathbf{c}(-\mathbf{1}) = \mathbf{0} \qquad \mathbf{P}(-1) = \delta^{-1}\mathbf{I}$

$\delta =$ small positive constant

For each $n = 0, 1, 2, \ldots$ compute:

Adaptation gain computation

$$\begin{aligned}
\bar{\mathbf{g}}_\lambda(n) &= \mathbf{P}(n-1)\mathbf{x}(n) \\
\alpha_\lambda(n) &= \lambda + \bar{\mathbf{g}}_\lambda^H(n)\mathbf{x}(n) \\
\mathbf{g}(n) &= \frac{\bar{\mathbf{g}}_\lambda(n)}{\alpha_\lambda(n)} \\
\mathbf{P}(n) &= \lambda^{-1}[\mathbf{P}(n-1) - \mathbf{g}(n)\bar{\mathbf{g}}_\lambda^H(n)]
\end{aligned}$$

Filtering

$$e(n) = y(n) - \mathbf{c}^H(n-1)\mathbf{x}(n)$$

Coefficient updating

$$\mathbf{c}(n) = \mathbf{c}(n-1) + \mathbf{g}(n)e^*(n)$$

Applications of adaptive filters

- Manolakis
  - pp. 590 – 607
- Hayes
  - pp. 509-521
  - pp. 530-534

Kalman Filter

- Suppose we want to obtain a linear MMSE estimate of RV y from {x_1, x_2, …, x_m}

$$\hat{y}_m \triangleq E\{y|x_1, x_2, \ldots, x_m\}$$

- The following approach can be followed
  - Estimate one-step prediction of x_m

$$\begin{aligned}\hat{x}_{m|m-1} &\triangleq \{x_m|x_1, x_2, \ldots, x_{m-1}\} \\ &= [\mathbf{R}_{m-1}^{-1}\mathbf{r}_{m-1}^{\mathrm{b}}]^H \mathbf{x}_{m-1} = -\mathbf{b}_{m-1}^H \mathbf{x}_{m-1} \\ &= -\sum_{k=1}^{m-1} [b_k^{(m-1)}]^* x_k\end{aligned}$$

– Determine optimal prediction error of x_m,

$$e_m^{\mathrm{b}} \triangleq x_m - \hat{x}_{m|m-1} = w_m$$

– Estimate y by using this new information

$$E\{y|w_m\} = E\{y_m w_m^*\}(E\{w_m w_m^*\})^{-1} w_m$$

– Finally estimate y_m

$$\hat{y}_m = \hat{y}_{m-1} + E\{y|w_m\} = \hat{y}_{m-1} + E\{y_m w_m^*\}(E\{w_m w_m^*\})^{-1} w_m$$

- Limitations
  - Solution to E{y|w_m} requires inversion.
  - The one step prediction requires infinite memory.
    - If we assume linear data relation model

$$x(n) = H(n)y(n) + v(n)$$

$$E\{v(n)y^*(l)\} = 0 \qquad \text{for all } n, l$$

$$E\{v(n)v^*(l)\} = r_v(n)\delta_{n,l} \qquad \text{for all } n, l$$

    - then

$$\begin{aligned}\hat{x}(n|n-1) &= E\{[H(n)y(n) + v(n)]|x(0), \ldots, x(n-1)\} \\ &= H(n)\hat{y}(n|n-1)\end{aligned}$$

- If we also assume the following linear signal model

$$y(n) = a(n-1)y(n-1) + \eta(n)$$

  – We will have recursive relation for obtaining the one-step prediction step.

- Model

$$\mathbf{y}(n) = \mathbf{A}(n-1)\mathbf{y}(n-1) + \mathbf{B}(n)\boldsymbol{\eta}(n)$$ State vector model

$$\mathbf{x}(n) = \mathbf{H}(n)\mathbf{y}(n) + \mathbf{v}(n)$$ Observation model

A(n-1) – state-transition matrix

H(n)– output matrix

η(n)– modeling error

v(n)– observation error

- The following are assumed

$$
\begin{aligned}
E\{\mathbf{y}(n)\mathbf{v}^H(l)\} &= \mathbf{0} \qquad \text{for all } n, l \\
E\{\boldsymbol{\eta}(n)\mathbf{v}^H(l)\} &= \mathbf{0} \qquad \text{for all } n, l \\
E\{\boldsymbol{\eta}(n)\mathbf{y}^H(-1)\} &= \mathbf{0} \qquad \text{for all } n \\
E\{\mathbf{y}(-1)\} &= \mathbf{0} \\
E\{\mathbf{y}(-1)\mathbf{y}^H(-1)\} &= \mathbf{R}_y(-1)
\end{aligned}
$$

TABLE 7.5

Summary of the Kalman filter algorithm.

1. **Input:**

   (*a*) Signal model parameters: $\mathbf{A}(n-1), \mathbf{B}(n), \mathbf{R}_\eta(n); n = 0, 1, 2, \ldots$

   (*b*) Observation model parameters: $\mathbf{H}(n), \mathbf{R}_v(n); n = 0, 1, 2, \ldots$

   (*c*) Observation data: $\mathbf{y}(n); n = 0, 1, 2, \ldots$

2. **Initialization:** $\hat{\mathbf{y}}(0|-1) = \mathbf{y}(-1) = \mathbf{0}; \mathbf{R}_{\tilde{y}}(-1|-1) = \mathbf{R}_y(-1)$

3. **Time recursion:** For $n = 0, 1, 2, \ldots$

   (*a*) Signal prediction: $\hat{\mathbf{y}}(n|n-1) = \mathbf{A}(n-1)\hat{\mathbf{y}}(n-1|n-1)$

   (*b*) Data prediction: $\hat{\mathbf{x}}(n|n-1) = \mathbf{H}(n)\hat{\mathbf{y}}(n|n-1)$

   (*c*) A priori error covariance:

   $$\mathbf{R}_{\tilde{y}}(n|n-1) = \mathbf{A}(n-1)\mathbf{R}_{\tilde{y}}(n-1|n-1)\mathbf{A}^H(n-1) + \mathbf{B}(n)\mathbf{R}_\eta(n)\mathbf{B}^H(n)$$

   (*d*) Kalman gain:

   $$\mathbf{K}(n) = \mathbf{R}_{\tilde{y}}(n|n-1)\mathbf{H}^H(n)\mathbf{R}_w^{-1}(n)$$

   $$\mathbf{R}_w(n) = \mathbf{H}(n)\mathbf{R}_{\tilde{y}}(n|n-1)\mathbf{H}^H(n + \mathbf{R}_v(n)$$

   (*e*) Signal update: $\hat{\mathbf{y}}(n|n) = \hat{\mathbf{y}}(n|n-1) + \mathbf{K}(n)[\mathbf{x}(n) - \hat{\mathbf{x}}(n|n-1)]$

   (*f*) A posteriori error covariance:

   $$\mathbf{R}_{\tilde{y}}(n|n) = [\mathbf{I} - \mathbf{K}(n)\mathbf{H}(n)]\mathbf{R}_{\tilde{y}}(n|n-1)$$

4. **Output:** Filtered estimate $\hat{\mathbf{y}}(n|n), n = 0, 1, 2, \ldots$