Adaptive Filtering - Theory and Applications

José C. M. Bermudez

Department of Electrical Engineering
Federal University of Santa Catarina
Florianópolis – SC
Brazil

IRIT - INP-ENSEEIHT, Toulouse
May 2011

Introduction

Estimation Techniques

Several techniques to solve estimation problems.

- Classical Estimation

  Maximum Likelihood (ML), Least Squares (LS), Moments, etc.

- Bayesian Estimation

  Minimum MSE (MMSE), Maximum A Posteriori (MAP), etc.

Linear Estimation

Frequently used in practice when there is a limitation in computational complexity – *Real-time operation*

Linear Estimators

Simpler to determine: depend on the first two moments of data

- Statistical Approach – Optimal Linear Filters
  - Minimum Mean Square Error
  - Require second order statistics of signals

- Deterministic Approach – Least Squares Estimators
  - Minimum Least Squares Error
  - Require handling of a data observation matrix

Limitations of Optimal Filters and LS Estimators

- Statistics of signals may not be available or cannot be accurately estimated
- There may not be available time for statistical estimation (real-time)
- Signals and systems may be non-stationary
- Memory required may be prohibitive
- Computational load may be prohibitive

Iterative Solutions

- Search the optimal solution starting from an initial guess
- Iterative algorithms are based on classical optimization algorithms
- Require reduced computational effort per iteration
- Need several iterations to converge to the optimal solution
- These methods form the basis for the development of adaptive algorithms
- Still require the knowledge of signal statistics

Adaptive Filters

Usually approximate iterative algorithms and:

- Do not require previous knowledge of the signal statistics
- Have a small computational complexity per iteration
- Converge to a neighborhood of the optimal solution

Adaptive filters are good for:

- Real-time applications, when there is no time for statistical estimation
- Applications with nonstationary signals and/or systems

Properties of Adaptive Filters

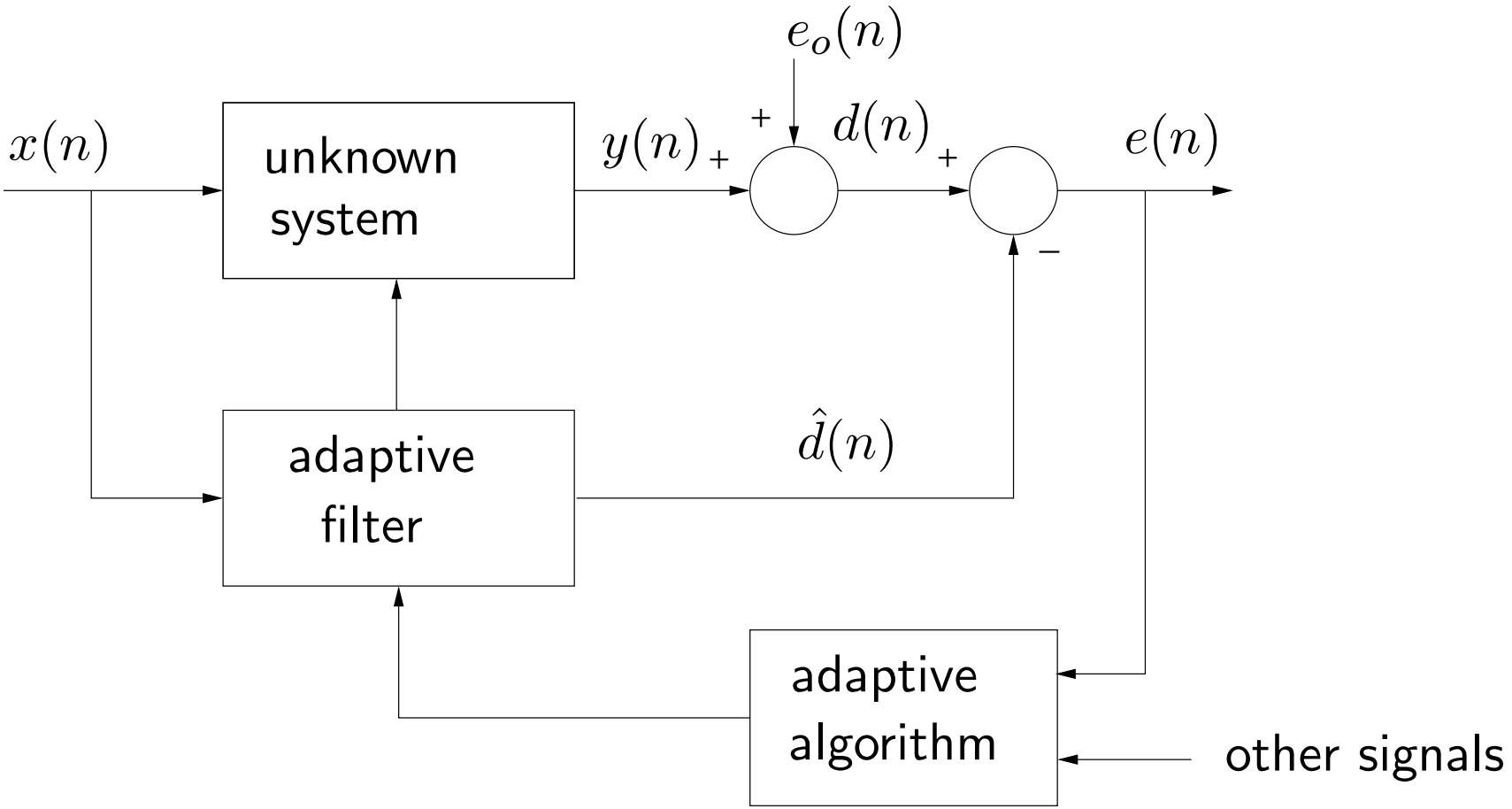
- They can operate satisfactorily in unknown and possibly time-varying environments without user intervention
- They improve their performance during operation by learning statistical characteristics from current signal observations
- They can track variations in the *signal operating environment* (SOE)

Adaptive Filtering Applications

Basic Classes of Adaptive Filtering Applications

- System Identification
- Inverse System Modeling
- Signal Prediction
- Interference Cancelation

System Identification



Applications – System Identification

Channel Estimation

- Communications systems
- Objective: model the channel to design distortion compensation
- $x(n)$: training sequence

Plant Identification

- Control systems
- Objective: model the plant to design a compensator
- $x(n)$: training sequence

Echo Cancellation

- Telephone systems and VoIP
- Echo caused by network impedance mismatches or acoustic environment
- Objective: model the echo path impulse response
- $x(n)$: transmitted signal
- $d(n)$: echo + noise

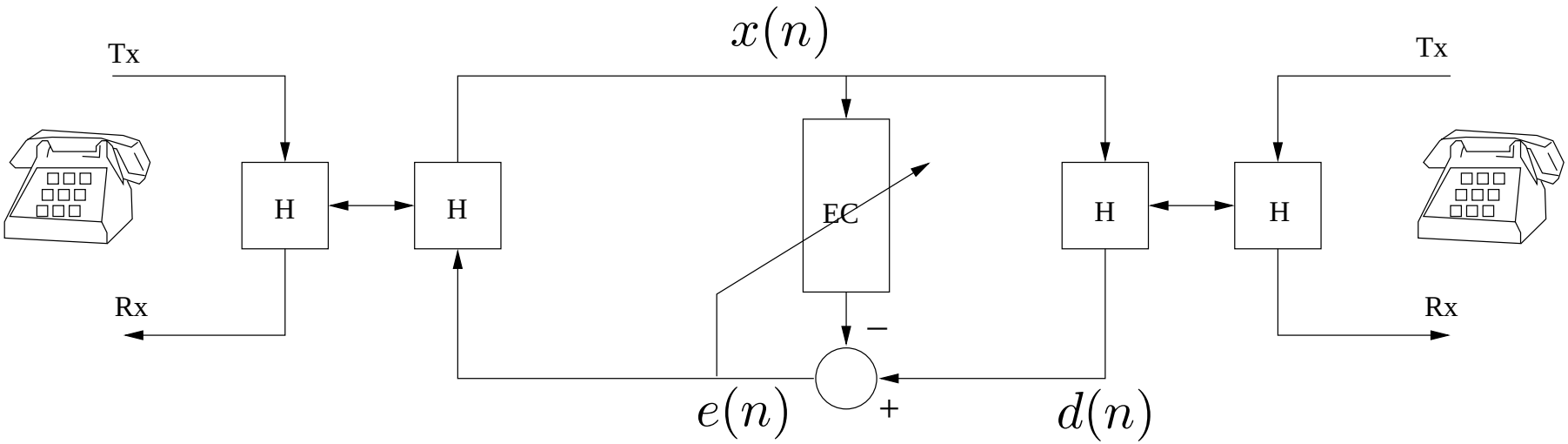
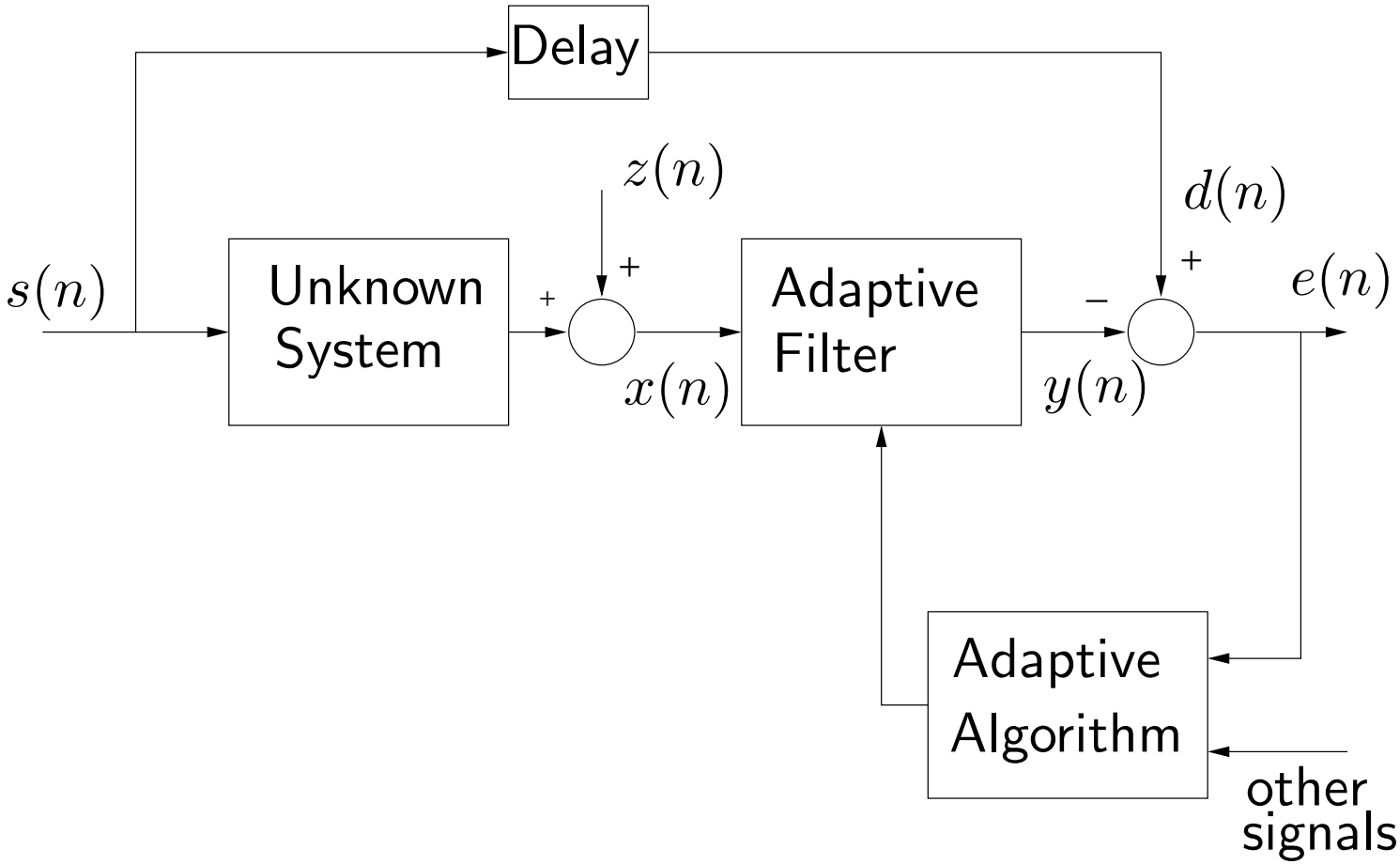


Figure: Network Echo Cancellation

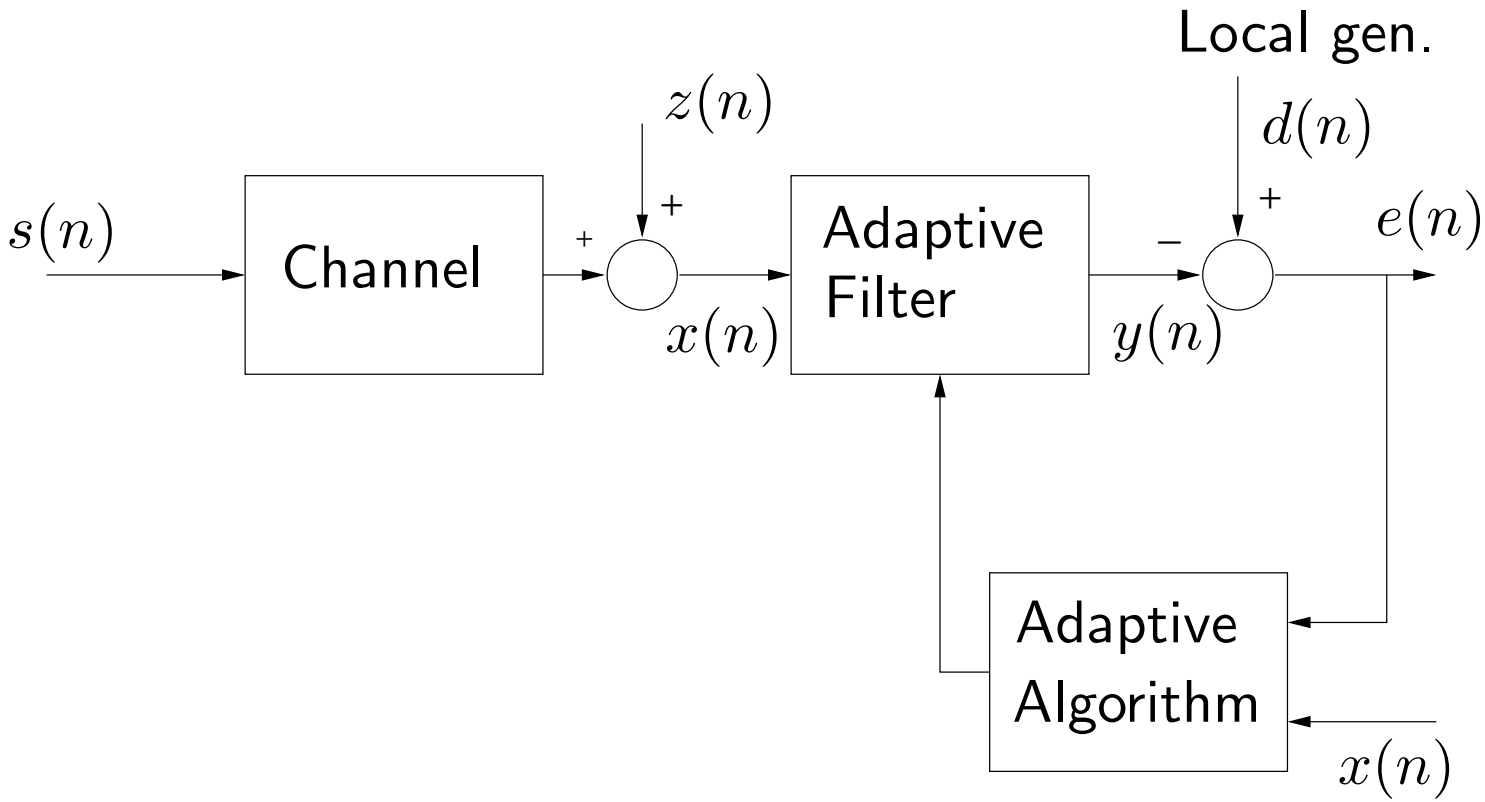
Inverse System Modeling

- Adaptive filter attempts to estimate unknown system's inverse
- Adaptive filter input usually corrupted by noise
- Desired response $d(n)$ may not be available



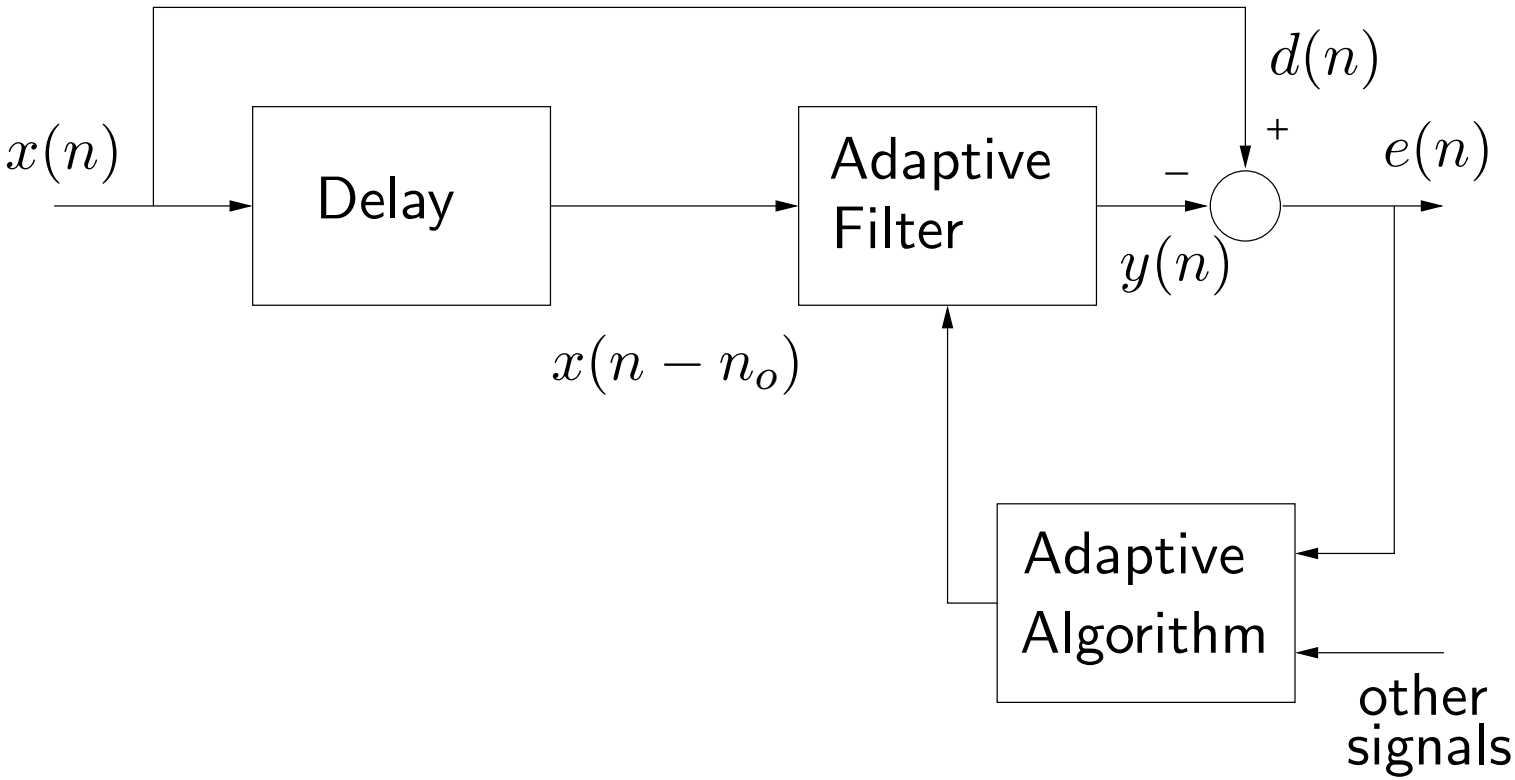
Applications – Inverse System Modeling

Channel Equalization



- Objective: reduce intersymbol interference
- Initially – training sequence in $d(n)$
- After training: $d(n)$ generated from previous decisions

Signal Prediction

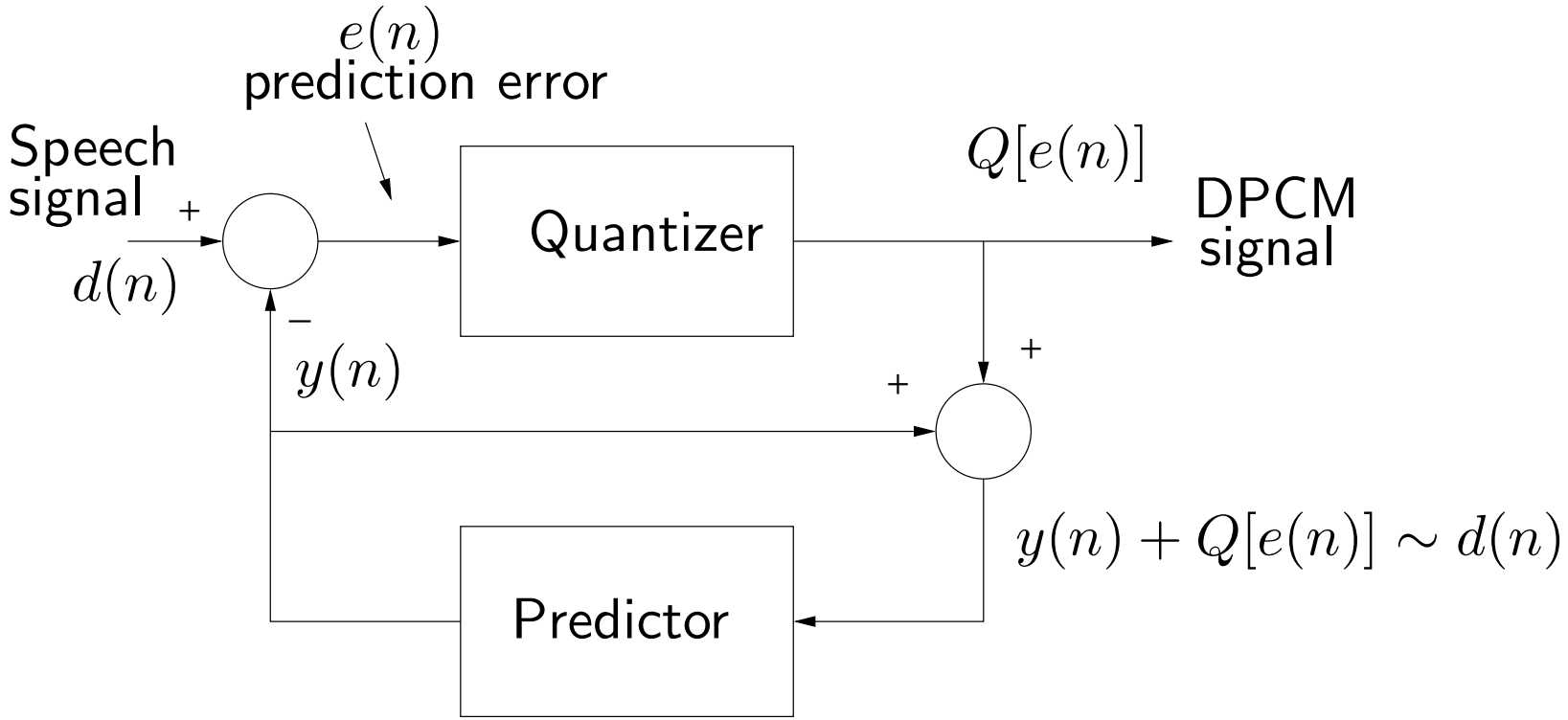


- most widely used case – forward prediction
- signal $x(n)$ to be predicted from samples $\{x(n - n_o), x(n - n_o - 1), \ldots, x(n - n_o - L)\}$

Application – Signal Prediction

DPCM Speech Quantizer - Linear Predictive Coding

- Objective: Reduce speech transmission bandwidth
- Signal transmitted all the time: quantization error
- Predictor coefficients are transmitted at low rate

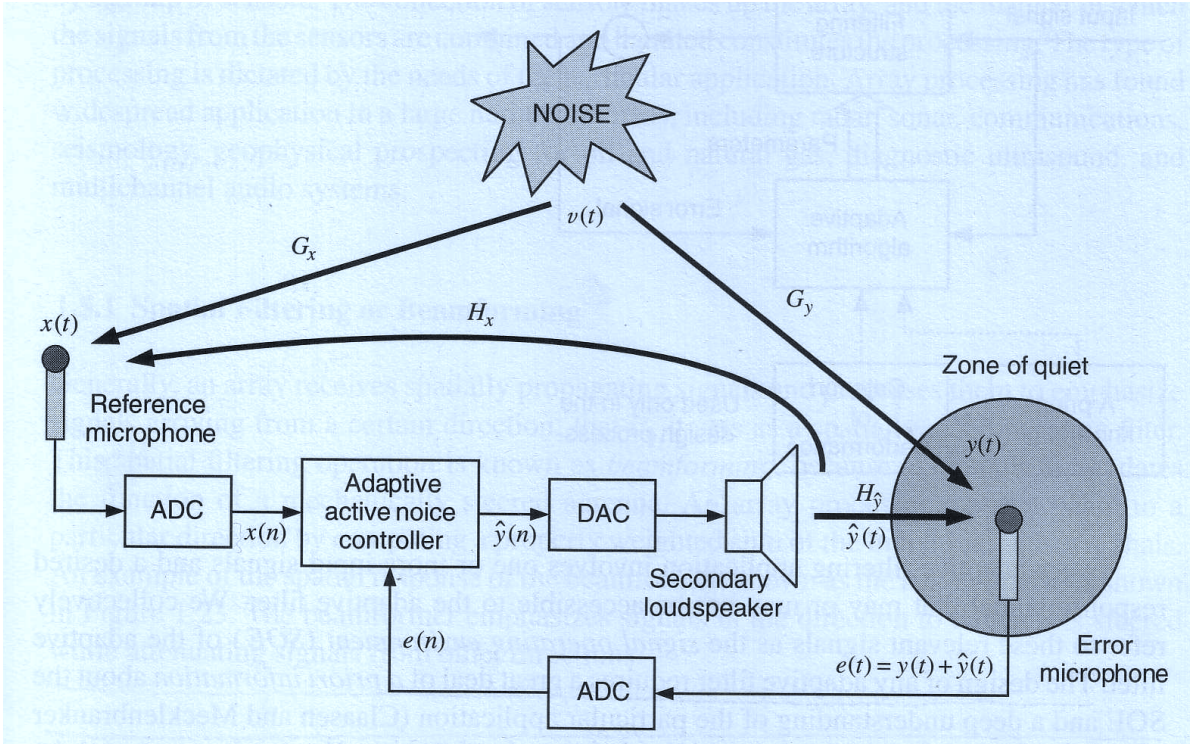


Interference Cancelation

- One or more sensor signals are used to remove interference and noise
- Reference signals correlated with the inteference should also be available
- Applications:
  - array processing for radar and communications
  - biomedical sensing systems
  - active noise control systems

Application – Interference Cancelation

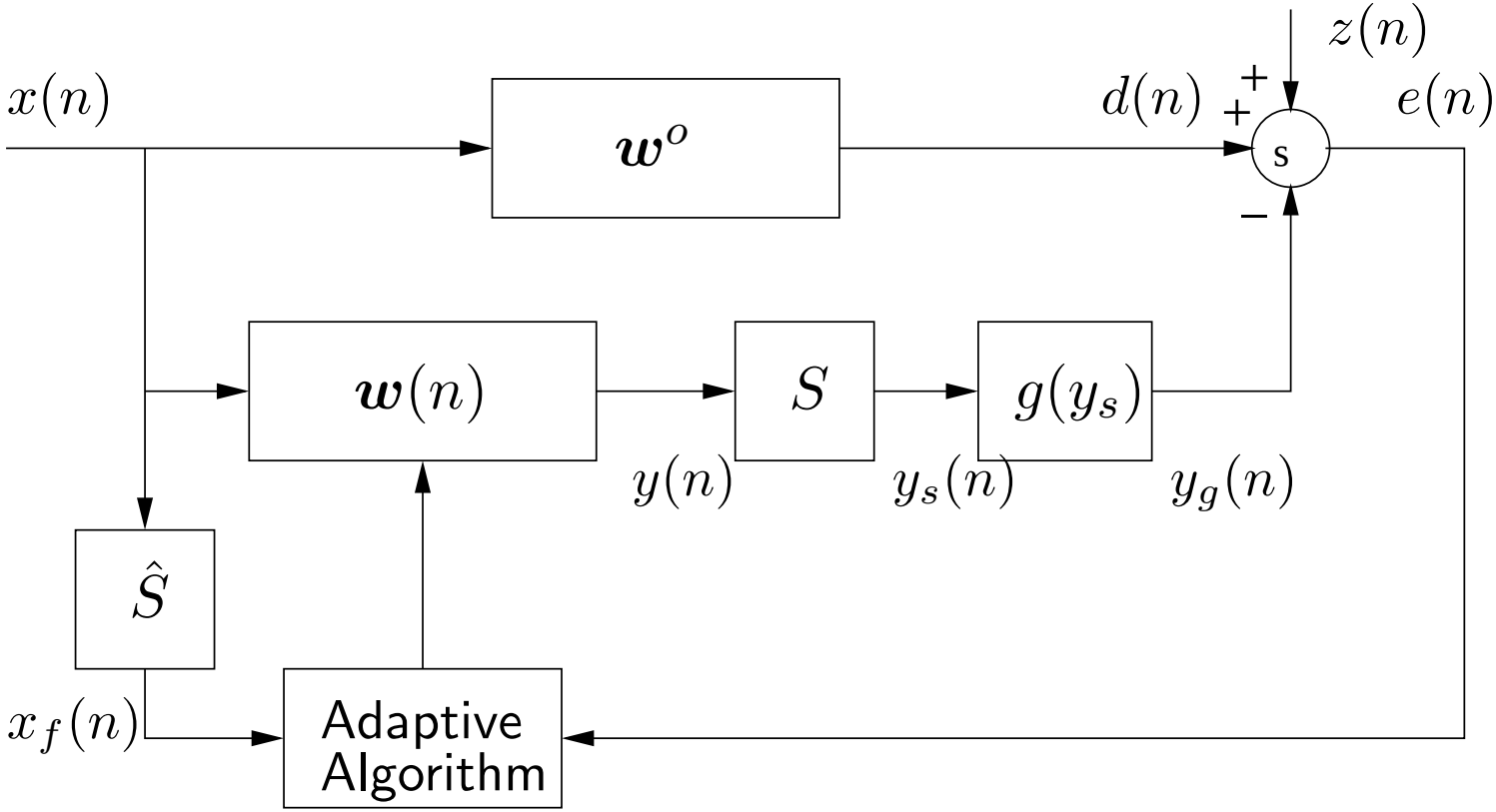
Active Noise Control



Ref: D.G. Manolakis, V.K. Ingle and S.M. Kogon, *Statistical and Adaptive Signal Processing, 2000.*

- Cancelation of acoustic noise using destructive interference
- Secondary system between the adaptive filter and the cancelation point is unavoidable
- Cancelation is performed in the acoustic environment

Active Noise Control – Block Diagram



Adaptive Filtering Principles

Adaptive Filter Features

Adaptive filters are composed of three basic modules:

- Filtering strucure
  - Determines the output of the filter given its input samples
  - Its weights are periodically adjusted by the adaptive algorithm
  - Can be linear or nonlinear, depending on the application
  - Linear filters can be FIR or IIR
- Performance criterion
  - Defined according to application and mathematical tractability
  - Is used to derive the adaptive algorithm
  - Its value at each iteration affects the adaptive weight updates
- Adaptive algorithm
  - Uses the performance criterion value and the current signals
  - Modifies the adaptive weights to improve performance
  - Its form and complexity are function of the structure and of the performance criterion

Signal Operating Environment (SOE)

Comprises all informations regarding the properties of the signals and systems

- Input signals
- Desired signal
- Unknown systems

If the SOE is nonstationary

- Aquisition or convergence mode: from start until close to best performance
- Tracking mode: readjustment following SOE's time variations

Adaptation can be

- Supervised – desired signal is available ➪ $e(n)$ can be evaluated
- Unsupervised – desired signal is unavailable

Performance Evaluation

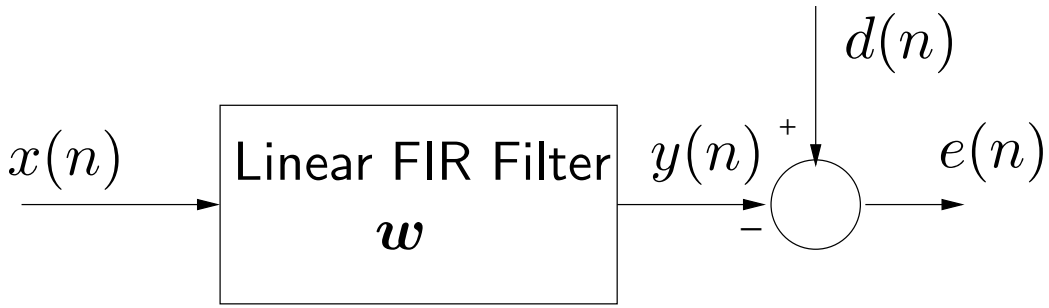
- Convergence rate
- Misadjustment
- Tracking
- Robustness (disturbances and numerical)
- Computational requirements (operations and memory)
- Structure
  - facility of implementation
  - performance surface
  - stability

Optimum versus Adaptive Filters in Linear Estimation

Conditions for this study

- Stationary SOE
- Filter structure is transversal FIR
- All signals are real valued
- Performance criterion: Mean-square error $E[e^2(n)]$

The Linear Estimation Problem



$$J_{ms} = E[e^2(n)]$$

The Linear Estimation Problem

$$\boldsymbol{x}(n) = [x(n), x(n-1), \cdots, x(n-N+1)]^T$$

$$y(n) = \boldsymbol{x}^T(n)\boldsymbol{w}$$

$$e(n) = d(n) - y(n) = d(n) - \boldsymbol{x}^T(n)\boldsymbol{w}$$

$$J_{ms} = E[e^2(n)] = \sigma_d^2 - 2\boldsymbol{p}^T\boldsymbol{w} + \boldsymbol{w}^T\boldsymbol{R}_{xx}\boldsymbol{w}$$

where

$$\boldsymbol{p} = E[\boldsymbol{x}(n)d(n)]; \quad \boldsymbol{R}_{xx} = E[\boldsymbol{x}(n)\boldsymbol{x}^T(n)]$$

Normal Equations

$$\boldsymbol{R}_{xx}\boldsymbol{w}^o = \boldsymbol{p} \quad \Rightarrow \quad \boldsymbol{w}^o = \boldsymbol{R}_{xx}^{-1}\boldsymbol{p} \quad \text{for } \boldsymbol{R}_{xx} > 0$$

$$J_{ms_{min}} = \sigma_d^2 - \boldsymbol{p}^T\boldsymbol{R}_{xx}^{-1}\boldsymbol{p}$$

What if $d(n)$ is nonstationary?

$$\boldsymbol{x}(n) = [x(n), x(n-1), \cdots, x(n-N+1)]^T$$

$$y(n) = \boldsymbol{x}^T(n)\boldsymbol{w}(n)$$

$$e(n) = d(n) - y(n) = d(n) - \boldsymbol{x}^T(n)\boldsymbol{w}(n)$$

$$J_{ms}(n) = E[e^2(n)] = \sigma_d^2(n) - 2\boldsymbol{p}(n)^T\boldsymbol{w}(n) + \boldsymbol{w}^T(n)\boldsymbol{R}_{xx}\boldsymbol{w}(n)$$

where

$$\boldsymbol{p}(n) = E[\boldsymbol{x}(n)d(n)]; \quad \boldsymbol{R}_{xx} = E[\boldsymbol{x}(n)\boldsymbol{x}^T(n)]$$

Normal Equations

$$\boldsymbol{R}_{xx}\boldsymbol{w}^o(n) = \boldsymbol{p}(n) \quad \Rightarrow \quad \boldsymbol{w}^o(n) = \boldsymbol{R}_{xx}^{-1}\boldsymbol{p}(n) \quad \text{for } \boldsymbol{R}_{xx} > 0$$

$$J_{ms_{min}}(n) = \sigma_d^2(n) - \boldsymbol{p}^T(n)\boldsymbol{R}_{xx}^{-1}\boldsymbol{p}(n)$$

Optimum Filters versus Adaptive Filters

Optimum Filters

- Compute $\boldsymbol{p}(n) = E[\boldsymbol{x}(n)d(n)]$
- Solve $\boldsymbol{R}_{xx}\boldsymbol{w}^o = \boldsymbol{p}(n)$
- Filter with $\boldsymbol{w}^o(n)$ ➪ $y(n) = \boldsymbol{x}^T(n)\boldsymbol{w}^o(n)$

Nonstationary SOE: Optimum filter determined for each value of n

Adaptive Filters

- Filtering: $y(n) = \boldsymbol{x}^T(n)\boldsymbol{w}(n)$
- Evaluate error: $e(n) = d(n) - y(n)$
- Adaptive algorithm:

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \Delta\boldsymbol{w}[\boldsymbol{x}(n), e(n)]$$

$\Delta\boldsymbol{w}(n)$ is chosen so that $\boldsymbol{w}(n)$ is close to $\boldsymbol{w}^o(n)$ for n large

Characteristics of Adaptive Filters

- Search for the optimum solution on the performance surface
- Follow principles of optimization techniques
- Implement a recursive optimization solution
- Convergence speed may depend on initialization
- Have stability regions
- Steady-state solution fluctuates about the optimum
- Can track time varying SOEs better than optimum filters
- Performance depends on the performance surface

Iterative Solutions for the Optimum Filtering Problem

Performance (Cost) Functions

- Mean-square error ➪ $E[e^2(n)]$ (Most popular)
  Adaptive algorithms: Least-Mean Square (LMS), Normalized LMS (NLMS), Affine Projection (AP), Recursive Least Squares (RLS), etc.
- Regularized MSE

$$J_{rms} = E[e^2(n)] + \alpha \|\boldsymbol{w}(n)\|^2$$

  Adaptive algorithm: leaky least-mean square (leaky LMS)
- ℓ_1 norm criterion

$$J_{\ell_1} = E[|e(n)|]$$

  Adaptive algorithm: Sign-Error

Performance (Cost) Functions – continued

- Least-mean fourth (LMF) criterion

$$J_{LMF} = E[e^4(n)]$$

  Adaptive algorithm: Least-Mean Fourth (LMF)

- Least-mean-mixed-norm (LMMN) criterion

$$J_{LMMN} = E[\alpha e^2(n) + \frac{1}{2}(1-\alpha)e^4(n)]$$

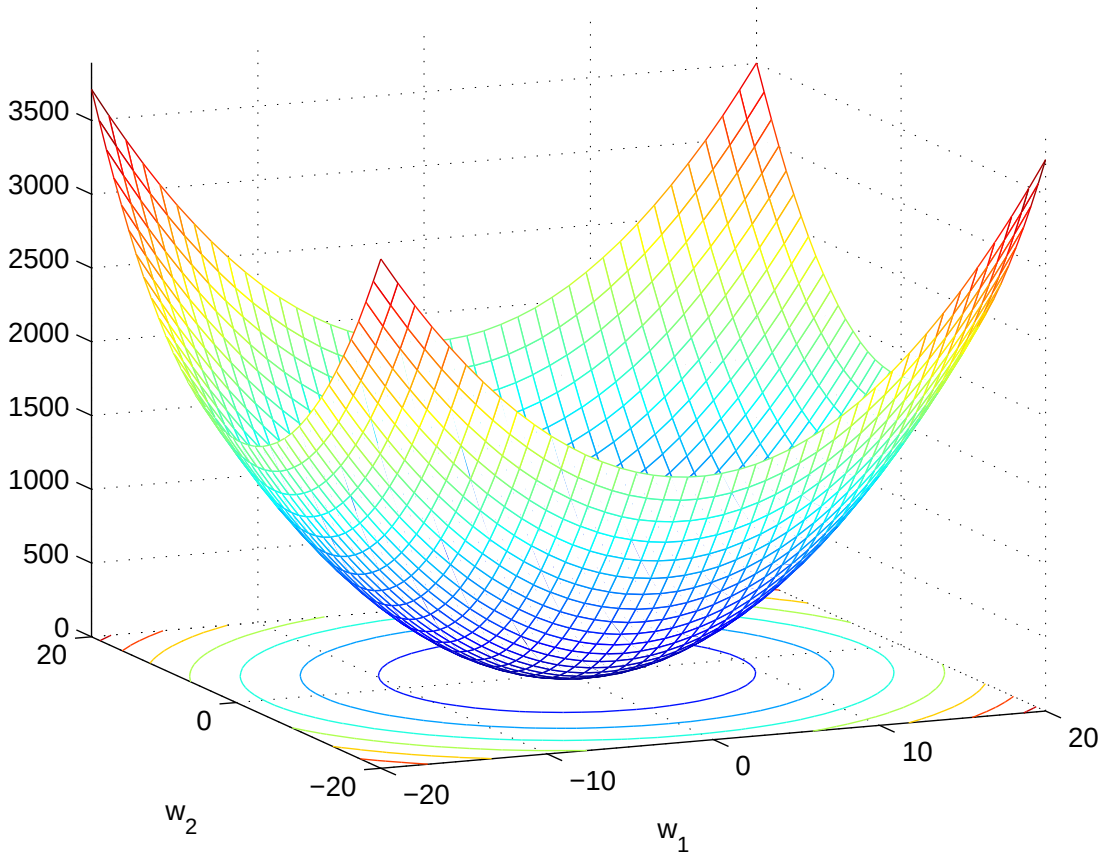
  Adaptive algorithm: Least-Mean-Mixed-Norm (LMMN)

- Constant-modulus criterion

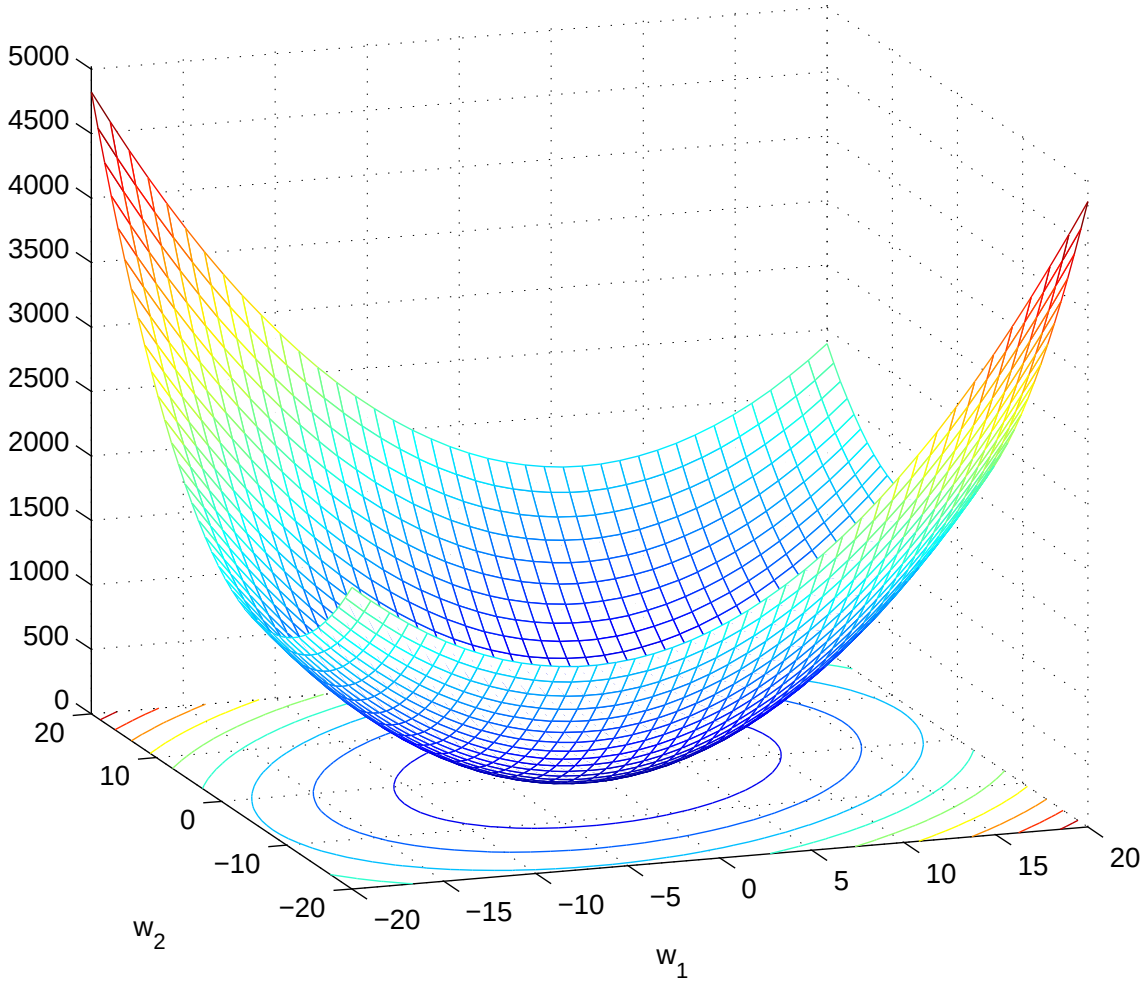
$$J_{CM} = E[\left(\gamma - |\boldsymbol{x}^T(n)\boldsymbol{w}(n)|^2\right)^2]$$

  Adaptive algorithm: Constant-Modulus (CM)

MSE Performance Surface – Small Input Correlation



MSE Performance Surface – Large Input Correlation



The Steepest Descent Algorithm – Stationary SOE

Cost Function

$$J_{ms}(n) = E[e^2(n)] = \sigma_d^2 - 2\boldsymbol{p}^T\boldsymbol{w}(n) + \boldsymbol{w}^T(n)\boldsymbol{R}_{xx}\boldsymbol{w}(n)$$

Weight Update Equation

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu\boldsymbol{c}(n)$$

μ: step-size
$\boldsymbol{c}(n)$: correction term (determines direction of $\Delta\boldsymbol{w}(n)$)
Steepest descent adjustment:

$$\boldsymbol{c}(n) = -\nabla J_{ms}(n) \quad \Rightarrow \quad J_{ms}(n+1) \leq J_{ms}(n)$$

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu[\boldsymbol{p} - \boldsymbol{R}_{xx}\boldsymbol{w}(n)]$$

Weight Update Equation About the Optimum Weights

Weight Error Update Equation

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu[\boldsymbol{p} - \boldsymbol{R}_{xx}\boldsymbol{w}(n)]$$

Using $\boldsymbol{p} = \boldsymbol{R}_{xx}\boldsymbol{w}^o$

$$\boldsymbol{w}(n+1) = (\boldsymbol{I} - \mu\boldsymbol{R}_{xx})\boldsymbol{w}(n) + \mu\boldsymbol{R}_{xx}\boldsymbol{w}^o$$

Weight error vector: $\boldsymbol{v}(n) = \boldsymbol{w}(n) - \boldsymbol{w}^o$

$$\boxed{\boldsymbol{v}(n+1) = (\boldsymbol{I} - \mu\boldsymbol{R}_{xx})\boldsymbol{v}(n)}$$

- Matrix $\boldsymbol{I} - \mu\boldsymbol{R}_{xx}$ must be *stable* for convergence ($|\lambda_i| < 1$)
- Assuming convergence, $\lim_{n\to\infty} \boldsymbol{v}(n) = \boldsymbol{0}$

Convergence Conditions

$$\boldsymbol{v}(n+1) = (\boldsymbol{I} - \mu \boldsymbol{R}_{xx})\boldsymbol{v}(n); \quad \boldsymbol{R}_{xx} \quad \text{positive definite}$$

Eigen-decomposition of $\boldsymbol{R}_{xx}$

$$\boldsymbol{R}_{xx} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T$$

$$\boldsymbol{v}(n+1) = (\boldsymbol{I} - \mu \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T)\boldsymbol{v}(n)$$

$$\boldsymbol{Q}^T\boldsymbol{v}(n+1) = \boldsymbol{Q}^T\boldsymbol{v}(n) - \mu \boldsymbol{\Lambda}\boldsymbol{Q}^T\boldsymbol{v}(n)$$

Defining $\tilde{\boldsymbol{v}}(n+1) = \boldsymbol{Q}^T\boldsymbol{v}(n+1)$

$$\boxed{\tilde{\boldsymbol{v}}(n+1) = (\boldsymbol{I} - \mu \boldsymbol{\Lambda})\tilde{\boldsymbol{v}}(n)}$$

Convergence Properties

$$\tilde{\boldsymbol{v}}(n+1) = (\boldsymbol{I} - \mu\boldsymbol{\Lambda})\tilde{\boldsymbol{v}}(n)$$

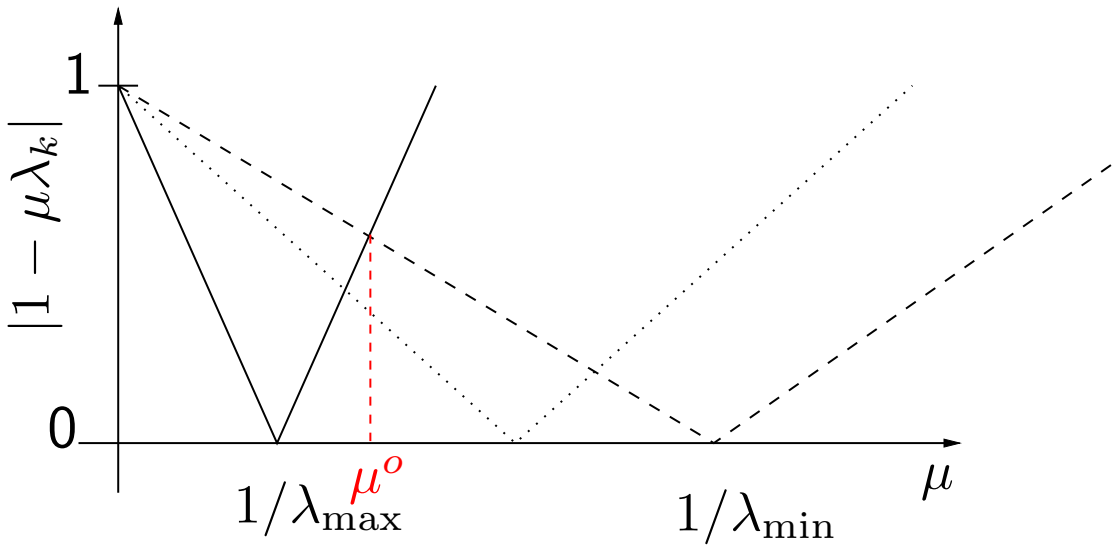
- $\tilde{v}_k(n+1) = (1-\mu\lambda_k)\tilde{v}_k(n),\ k = 1, \ldots, N$
- $\tilde{v}_k(n) = (1-\mu\lambda_k)^n \tilde{v}_k(0)$
- Convergence modes
  - monotonic if $\quad 0 < 1 - \mu\lambda_k < 1$
  - oscillatory if $\quad -1 < 1 - \mu\lambda_k < 0$
- Convergence if $|1 - \mu\lambda_k| < 1 \Rightarrow \boxed{0 < \mu < \frac{2}{\lambda_{\max}}}$

Optimal Step-Size

$$\tilde{v}_k(n) = (1 - \mu\lambda_k)^n \tilde{v}_k(0); \text{convergence modes: } 1 - \mu\lambda_k$$

- $\max |1 - \mu\lambda_k|$: slowest mode
- $\min |1 - \mu\lambda_k|$: fastest mode

Optimal Step-Size – continued



$$\min_{\mu} \max\{|1-\mu\lambda_k|\}$$

$$1-\mu^o\lambda_{\min} = -(1-\mu^o\lambda_{\max})$$

$$\boxed{\mu^o = \frac{2}{\lambda_{\max}+\lambda_{\min}}}$$

Optimal slowest modes: $\pm\frac{\rho-1}{\rho+1}$; $\rho = \frac{\lambda_{\max}}{\lambda_{\min}}$

The Learning Curve – $J_{ms}(n)$

$$J_{ms}(n) = J_{ms_{\min}} + \overbrace{\tilde{\boldsymbol{v}}^T(n)\boldsymbol{\Lambda}\tilde{\boldsymbol{v}}(n)}^{\text{Excess MSE}} = J_{ms_{\min}} + \sum_{k=1}^{N} \lambda_k \tilde{v}_k^2(n)$$

Since $\tilde{v}_k(n) = (1-\mu\lambda_k)^n \tilde{v}_k(0)$,

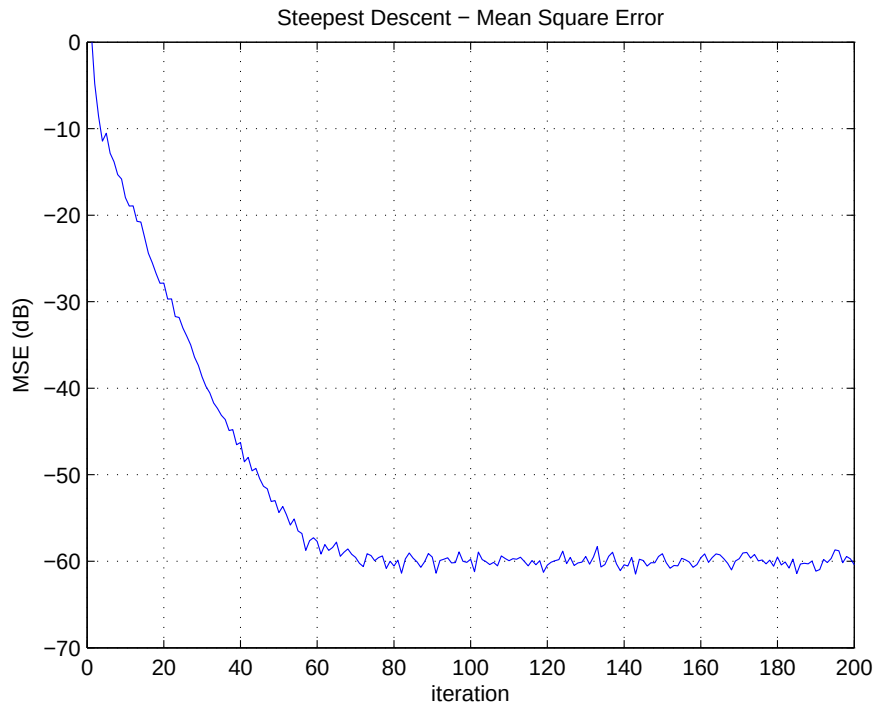
$$J_{ms}(n) = J_{ms_{\min}} + \sum_{k=1}^{N} \lambda_k (1-\mu\lambda_k)^{2n} \tilde{v}_k^2(0)$$

- $\lambda_k(1-\mu\lambda_k)^2 > 0$ ➪ monotonic convergence
- Stability limit is again $0 < \mu < \frac{2}{\lambda_{\max}}$
- $J_{ms}(n)$ converges faster than $\boldsymbol{w}(n)$
- Algorithm converges faster as $\rho = \lambda_{\max}/\lambda_{\min} \to 1$

Simulation Results

$$x(n) = \alpha\, x(n-1) + v(n)$$

Input: White noise ($\rho = 1$)



Input: AR(1), $\alpha = 0.7$ ($\rho = 5.7$)

- Linear system identification - FIR with 20 coefficients
- Step-size $\mu = 0.3$
- Noise power $\sigma_v^2 = 10^{-6}$

The Newton Algorithm

- Steepest descent: linear approx. of J_{ms} about the operating point
- Newton's method: Quadratic approximation of J_{ms}

Expanding $J_{ms}(\boldsymbol{w})$ in Taylor's series about $\boldsymbol{w}(n)$,

$$\begin{aligned} J_{ms}(\boldsymbol{w}) \sim & J_{ms}[\boldsymbol{w}(n)] + \nabla^T J_{ms}[\boldsymbol{w} - \boldsymbol{w}(n)] \\ & + \frac{1}{2}[\boldsymbol{w} - \boldsymbol{w}(n)]^T \boldsymbol{H}(n)[\boldsymbol{w} - \boldsymbol{w}(n)] \end{aligned}$$

Diferentiating w.r.t. $\boldsymbol{w}$ and equating to zero at $\boldsymbol{w} = \boldsymbol{w}(n+1)$,

$$\nabla J_{ms}[\boldsymbol{w}(n+1)] = \nabla J_{ms}[\boldsymbol{w}(n)] + \boldsymbol{H}[\boldsymbol{w}(n)][\boldsymbol{w}(n+1) - \boldsymbol{w}(n)] = \boldsymbol{0}$$

$$\boxed{\boldsymbol{w}(n+1) = \boldsymbol{w}(n) - \boldsymbol{H}^{-1}[\boldsymbol{w}(n)] \nabla J_{ms}[\boldsymbol{w}(n)]}$$

The Newton Algorithm – continued

- $\nabla J_{ms}[\boldsymbol{w}(n)] = -2\boldsymbol{p} + 2\boldsymbol{R}_{xx}\boldsymbol{w}(n)$
- $\boldsymbol{H}(\boldsymbol{w}(n)) = 2\boldsymbol{R}_{xx}$

Thus, adding a step-size control,

$$\boxed{\boldsymbol{w}(n+1) = \boldsymbol{w}(n) - \mu\boldsymbol{R}_{xx}^{-1}[-\boldsymbol{p} + \boldsymbol{R}_{xx}\boldsymbol{w}(n)]}$$

- Quadratic surface ➪ conv. in *one* iteration for $\mu = 1$ ☺
- Requires the determination of $\boldsymbol{R}_{xx}^{-1}$ ☹
- Can be used to derive simpler adaptive algorithms
- When $\boldsymbol{H}(n)$ is close to singular ➪ regularization

$$\tilde{\boldsymbol{H}}(n) = 2\boldsymbol{R}_{xx} + 2\varepsilon\boldsymbol{I}$$

Basic Adaptive Algorithms

Least Mean Squares (LMS) Algorithm

- Can be interpreted in different ways
- Each interpretation helps understanding the algorithm behavior
- Some of these interpretations are related to the steepest descent algorithm

LMS as a Stochastic Gradient Algorithm

- Suppose we use the estimate $J_{ms}(n) = E[e^2(n)] \simeq e^2(n)$
- The estimated gradient vector becomes

$$\hat{\nabla} J_{ms}(n) = \frac{\partial e^2(n)}{\partial \boldsymbol{w}(n)} = 2e(n)\frac{\partial e(n)}{\partial \boldsymbol{w}(n)}$$

- Since $e(n) = d(n) - \boldsymbol{x}^T(n)\boldsymbol{w}(n)$,

$$\hat{\nabla} J_{ms}(n) = -2e(n)\boldsymbol{x}(n) \qquad (\textit{stochastic gradient})$$

and, using the steepest descent weight update equation,

$$\boxed{\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu e(n)\boldsymbol{x}(n)} \qquad \text{(LMS weight update)}$$

LMS as a Stochastic Estimation Algorithm

- $\nabla J_{ms}(n) = -2\boldsymbol{p} + 2\boldsymbol{R}_{xx}\boldsymbol{w}(n)$
- Stochastic estimators

$$\hat{\boldsymbol{p}} = d(n)\boldsymbol{x}(n) \qquad \hat{\boldsymbol{R}}_{xx} = \boldsymbol{x}(n)\boldsymbol{x}^T(n)$$

Then,

$$\hat{\nabla} J_{ms}(n) = -2d(n)\boldsymbol{x}(n) + 2\boldsymbol{x}(n)\boldsymbol{x}^T(n)\boldsymbol{w}(n)$$

- Using $\hat{\nabla} J_{ms}(n)$ is the steepest descent weight update,

$$\boxed{\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu e(n)\boldsymbol{x}(n)}$$

LMS – A Solution to a Local Optimization

- Error expressions
  $e(n) = d(n) - \boldsymbol{x}^T(n)\boldsymbol{w}(n)$ (*a priori* error)
  $\epsilon(n) = d(n) - \boldsymbol{x}^T(n)\boldsymbol{w}(n+1)$ (*a posteriori* error)
- We want to maximize $|\epsilon(n) - e(n)|$ with $|\epsilon(n)| < |e(n)|$

$$\epsilon(n) - e(n) = -\boldsymbol{x}^T(n)\Delta\boldsymbol{w}(n)$$
$$\Delta\boldsymbol{w}(n) = \boldsymbol{w}(n+1) - \boldsymbol{w}(n)$$

- Expressing $\Delta\boldsymbol{w}(n)$ as $\Delta\boldsymbol{w}(n) = \Delta\tilde{\boldsymbol{w}}(n)e(n)$

$$\epsilon(n) - e(n) = -\boldsymbol{x}^T(n)\Delta\tilde{\boldsymbol{w}}(n)e(n)$$

- For max $|\epsilon(n) - e(n)|$ ➪ $\Delta\tilde{\boldsymbol{w}}(n)$ in the direction of $\boldsymbol{x}(n)$

- ➪ $\Delta\tilde{\boldsymbol{w}}(n) = \mu\boldsymbol{x}(n)$ and

$$\Delta\boldsymbol{w}(n) = \mu\boldsymbol{x}(n)e(n)$$

and

$$\boxed{\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu e(n)\boldsymbol{x}(n)}$$

- As

$$\epsilon(n) - e(n) = -\boldsymbol{x}^T(n)\Delta\boldsymbol{w}(n) = -\mu\boldsymbol{x}^T(n)\boldsymbol{x}(n)e(n)$$

$|\epsilon(n)| < |e(n)|$ requires $|1 - \mu\boldsymbol{x}^T(n)\boldsymbol{x}(n)| < 1$, or

$$\boxed{0 < \mu < \frac{2}{\|\boldsymbol{x}(n)\|^2}} \quad \textit{(stability region)}$$

Observations - LMS Algorithm

- LMS is a noisy approximation of the steepest descent algorithm
- The gradient estimate is unbiased
- The errors in the gradient estimate lead to $J_{ms_{ex}}(\infty) \neq 0$
- Vector $\boldsymbol{w}(n)$ is now random
- Steepest descent properties are no longer guaranteed
  ➪ *LMS analysis required*
- The instantaneous estimates allow tracking without redesign

Some Research Results

- J. C. M. Bermudez and N. J. Bershad, "A nonlinear analytical model for the quantized LMS algorithm - the arbitrary step size case," IEEE Transactions on Signal Processing, vol.44, No. 5, pp. 1175-1183, May 1996.
- J. C. M. Bermudez and N. J. Bershad, "Transient and tracking performance analysis of the quantized LMS algorithm for time-varying system identification," IEEE Transactions on Signal Processing, vol.44, No. 8, pp. 1990-1997, August 1996.
- N. J. Bershad and J. C. M. Bermudez, "A nonlinear analytical model for the quantized LMS algorithm - the power-of-two step size case," IEEE Transactions on Signal Processing, vol.44, No. 11, pp. 2895- 2900, November 1996.
- N. J. Bershad and J. C. M. Bermudez, "Sinusoidal interference rejection analysis of an LMS adaptive feedforward controller with a noisy periodic reference," IEEE Transactions on Signal Processing, vol.46, No. 5, pp. 1298-1313, May 1998.
- J. C. M. Bermudez and N. J. Bershad, "Non-Wiener behavior of the Filtered-X LMS algorithm," IEEE Trans. on Circuits and Systems II - Analog and Digital Signal Processing, vol.46, No. 8, pp. 1110-1114, Aug 1999.

Some Research Results – continued

- O. J. Tobias, J. C. M. Bermudez and N. J. Bershad, "Mean weight behavior of the Filtered-X LMS algorithm," IEEE Transactions on Signal Processing, vol. 48, No. 4, pp. 1061-1075, April 2000.
- M. H. Costa, J. C. M. Bermudez and N. J. Bershad, "Stochastic analysis of the LMS algorithm with a saturation nonlinearity following the adaptive filter output," IEEE Transactions on Signal Processing, vol. 49, No. 7, pp. 1370-1387, July 2001.
- M. H. Costa, J. C, M. Bermudez and N. J. Bershad, "Stochastic analysis of the Filtered-X LMS algorithm in systems with nonlinear secondary paths," IEEE Transactions on Signal Processing, vol. 50, No. 6, pp. 1327-1342, June 2002.
- M. H. Costa, J. C. M. Bermudez and N. J. Bershad, "The performance surface in filtered nonlinear mean square estimation," IEEE Transactions on Circuits and Systems I, vol. 50, No. 3, p. 445-447, March 2003.
- G. Barrault, J. C. M. Bermudez and A. Lenzi, "New Analytical Model for the Filtered-x Least Mean Squares Algorithm Verified Through Active Noise Control Experiment," Mechanical Systems and Signal Processing, v. 21, p. 1839-1852, 2007.

Some Research Results – continued

- N. J. Bershad, J. C. M. Bermudez and J. Y. Tourneret, "Stochastic Analysis of the LMS Algorithm for System Identification with Subspace Inputs," IEEE Trans. on Signal Process., v. 56, p. 1018-1027, 2008.
- J. C. M. Bermudez, N. J. Bershad and J. Y. Tourneret, "An Affine Combination of Two LMS Adaptive Filters Transient Mean-Square Analysis," IEEE Trans. on Signal Process., v. 56, p. 1853-1864, 2008.
- M. H. Costa, L. R. Ximenes and J. C. M. Bermudez, "Statistical Analysis of the LMS Adaptive Algorithm Subjected to a Symmetric Dead-Zone Nonlinearity at the Adaptive Filter Output," Signal Processing, v. 88, p. 1485-1495, 2008.
- M. H. Costa and J. C. M. Bermudez, "A Noise Resilient Variable Step-Size LMS Algorithm," Signal Processing, v. 88, no. 3, p. 733-748, March 2008.
- P. Honeine, C. Richard, J. C. M. Bermudez, J. Chen and H. Snoussi, "A Decentralized Approach for Nonlinear Prediction of Time Series Data in Sensor Networks," EURASIP Journal on Wireless Communications and Networking, v. 2010, p. 1-13, 2010. (KNLMS)

The Normalized LMS Algorithm – NLMS

- Most Employed adaptive algorithm in real-time applications
- Like LMS has different interpretations (even more)
- Alleviates a drawback of the LMS algorithm

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu e(n)\boldsymbol{x}(n)$$

  - If amplitude of $\boldsymbol{x}(n)$ is large ➪ Gradient noise amplification
  - Sub-optimal performance when σ_x^2 varies with time (for instance, speech)

NLMS – A Solution to a Local Optimization Problem

- Error expressions
  $e(n) = d(n) - \boldsymbol{x}^T(n)\boldsymbol{w}(n)$ (*a priori* error)
  $\epsilon(n) = d(n) - \boldsymbol{x}^T(n)\boldsymbol{w}(n+1)$ (*a posteriori* error)
- We want to maximize $|\epsilon(n) - e(n)|$ with $|\epsilon(n)| < |e(n)|$

$$\epsilon(n) - e(n) = -\boldsymbol{x}^T(n)\Delta\boldsymbol{w}(n) \qquad (A)$$
$$\Delta\boldsymbol{w}(n) = \boldsymbol{w}(n+1) - \boldsymbol{w}(n)$$

- For $|\epsilon(n)| < |e(n)|$ we impose the restriction

$$\epsilon(n) = (1-\mu)e(n), \quad |1-\mu| < 1$$

$$\Rightarrow \quad \epsilon(n) - e(n) = -\mu e(n) \qquad (B)$$

- For max $|\epsilon(n) - e(n)| \Rightarrow \Delta\boldsymbol{w}(n)$ in the direction of $\boldsymbol{x}(n)$

$$\Rightarrow \Delta\boldsymbol{w}(n) = k\boldsymbol{x}(n) \qquad (C)$$

- Using (A), (B) and (C),

$$k = \mu \frac{e(n)}{\boldsymbol{x}^T(n)\boldsymbol{x}(n)}$$

and

$$\boxed{\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu \frac{e(n)\boldsymbol{x}(n)}{\boldsymbol{x}^T(n)\boldsymbol{x}(n)}} \quad \text{(NLMS weight update)}$$

NLMS – Solution to a Constrained Optimization Problem

- Error sequence

$$y(n) = \boldsymbol{x}^T(n)\boldsymbol{w}(n) \quad \text{(estimate of } d(n)\text{)}$$
$$e(n) = d(n) - y(n) = d(n) - \boldsymbol{x}^T(n)\boldsymbol{w}(n) \quad \text{(estimation error)}$$

- Optimization (principle of minimal disturbance)

$$\begin{aligned} \text{Minimize} \quad & \|\Delta\boldsymbol{w}(n)\|^2 = \|\boldsymbol{w}(n+1) - \boldsymbol{w}(n)\|^2 \\ \text{subject to:} \quad & \boldsymbol{x}^T(n)\boldsymbol{w}(n+1) = d(n) \end{aligned}$$

This problem can be solved using the method of Lagrange multipliers

- Using Lagrange multipliers, we minimize

$$f[\boldsymbol{w}(n+1)] = \|\boldsymbol{w}(n+1) - \boldsymbol{w}(n)\|^2 + \lambda[d(n) - \boldsymbol{x}^T(n)\boldsymbol{w}(n+1)]$$

- Differentiating w.r.t. $\boldsymbol{w}(n+1)$ and equating the result to zero,

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \frac{1}{2}\lambda\boldsymbol{x}(n) \qquad (*)$$

Using this result in $\boldsymbol{x}^T(n)\boldsymbol{w}(n+1) = d(n)$ yields

$$\lambda = \frac{2e(n)}{\boldsymbol{x}^T(n)\boldsymbol{x}(n)}$$

- Using this result in (*) yields

$$\boxed{\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu\frac{e(n)\boldsymbol{x}(n)}{\boldsymbol{x}^T(n)\boldsymbol{x}(n)}}$$

NLMS as an Orthogonalization Process

- Conditions for the analysis
  - $e(n) = d(n) - \boldsymbol{x}^T(n)\boldsymbol{w}(n)$
  - $\boldsymbol{w}^o$ is the optimal solution (Wiener solution)
  - $d(n) = \boldsymbol{x}^T(n)\boldsymbol{w}^o$ (no noise)
  - $\boldsymbol{v}(n) = \boldsymbol{w}(n) - \boldsymbol{w}^o$ (weight error vector)
- Error signal

$$\begin{aligned} e(n) &= d(n) - \boldsymbol{x}^T(n)[\boldsymbol{v}(n) + \boldsymbol{w}^o] \\ &= \boldsymbol{x}^T(n)\boldsymbol{w}^o - \boldsymbol{x}^T(n)[\boldsymbol{v}(n) + \boldsymbol{w}^o] \\ &= -\boldsymbol{x}^T(n)\boldsymbol{v}(n) \end{aligned}$$

$$e(n) = -\boldsymbol{x}^T(n)\boldsymbol{v}(n)$$

- *Interpretation:* To minimize the error $\boldsymbol{v}(n)$ should be orthogonal to all input vectors
  Restriction: We have only one vector $\boldsymbol{x}(n)$
- *Iterative solution:*
  We can subtract from $\boldsymbol{v}(n)$ its component in the direction of $\boldsymbol{x}(n)$ at each iteration
- If there are N adaptive coefficients, $\boldsymbol{v}(n)$ could be reduced to zero after N orthogonal input vectors
- Iterative projection extraction
  ➪ Gram-Schmidt orthogonalization

- Recursive orthogonalization

$$\begin{aligned} n=0: &\quad \boldsymbol{v}(1) = \boldsymbol{v}(0) - \text{proj. of } \boldsymbol{v}(0) \text{ onto } \boldsymbol{x}(0) \\ n=1: &\quad \boldsymbol{v}(2) = \boldsymbol{v}(1) - \text{proj. of } \boldsymbol{v}(1) \text{ onto } \boldsymbol{x}(1) \\ \vdots: &\quad \vdots \\ n+1: &\quad \boldsymbol{v}(n+1) = \boldsymbol{v}(n) - \text{proj. of } \boldsymbol{v}(n) \text{ onto } \boldsymbol{x}(n) \end{aligned}$$

- Projection of $\boldsymbol{v}(n)$ onto $\boldsymbol{x}(n)$

$$\mathcal{P}_{\boldsymbol{x}(n)}[\boldsymbol{v}(n)] = \left\{\boldsymbol{x}(n)[\boldsymbol{x}^T(n)\boldsymbol{x}(n)]^{-1}\boldsymbol{x}^T(n)\right\}\boldsymbol{v}(n)$$

- Weight update equation

$$\begin{aligned} \boldsymbol{v}(n+1) &= \boldsymbol{v}(n) - \mu \overbrace{\underbrace{[\boldsymbol{x}^T(n)\boldsymbol{x}(n)]^{-1}}_{\text{scalar}}\boldsymbol{x}^T(n)\boldsymbol{v}(n)}^{-e(n)}\boldsymbol{x}(n) \\ &= \boldsymbol{v}(n) + \mu\frac{e(n)\boldsymbol{x}(n)}{\boldsymbol{x}^T(n)\boldsymbol{x}(n)} \end{aligned}$$

NLMS has its owns problems!

- NLMS solves the LMS gradient error amplification problem, but ...
- What happens if $\|\boldsymbol{x}(n)\|^2$ gets too small?
- One needs to add some regularization

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu \frac{e(n)\boldsymbol{x}(n)}{\boldsymbol{x}^T(n)\boldsymbol{x}(n) + \varepsilon} \quad (\varepsilon\text{−NLMS})$$

ε-NLMS – Stochastic Approximation of Regularized Newton

- Regularized Newton Algorithm

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu[\varepsilon \boldsymbol{I} + \boldsymbol{R}_{xx}]^{-1}[\boldsymbol{p} - \boldsymbol{R}_{xx}\boldsymbol{w}(n)]$$

- Instantaneous estimates

$$\hat{\boldsymbol{p}} = \boldsymbol{x}(n)d(n)$$
$$\hat{\boldsymbol{R}}_{xx} = \boldsymbol{x}(n)\boldsymbol{x}^T(n)$$

- Using these estimates

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu[\varepsilon \boldsymbol{I} + \boldsymbol{x}(n)\boldsymbol{x}^T(n)]^{-1}\boldsymbol{x}(n) \overbrace{[d(n) - \boldsymbol{x}^T(n)\boldsymbol{w}(n)]}^{e(n)}$$

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu[\varepsilon \boldsymbol{I} + \boldsymbol{x}(n)\boldsymbol{x}^T(n)]^{-1}\boldsymbol{x}(n)e(n)$$

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu[\varepsilon \boldsymbol{I} + \boldsymbol{x}(n)\boldsymbol{x}^T(n)]^{-1}\boldsymbol{x}(n)e(n)$$

- Inversion of $\varepsilon \boldsymbol{I} + \boldsymbol{x}(n)\boldsymbol{x}^T(n)$?
- Matrix Inversion Formula

$$[\boldsymbol{A} + \boldsymbol{BCD}]^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{B}[\boldsymbol{C}^{-1} + \boldsymbol{DA}^{-1}\boldsymbol{B}]^{-1}\boldsymbol{DA}^{-1}$$

- Thus,

$$[\varepsilon \boldsymbol{I} + \boldsymbol{x}(n)\boldsymbol{x}^T(n)]^{-1} = \varepsilon^{-1}\boldsymbol{I} - \frac{\varepsilon^{-2}}{1 + \varepsilon^{-1}\boldsymbol{x}^T(n)\boldsymbol{x}(n)}\boldsymbol{x}(n)\boldsymbol{x}^T(n)$$

- Post-multiplying both sides by $\boldsymbol{x}(n)$ and rearranging

$$[\varepsilon \boldsymbol{I} + \boldsymbol{x}(n)\boldsymbol{x}^T(n)]^{-1}\boldsymbol{x}(n) = \frac{\boldsymbol{x}(n)}{\varepsilon + \boldsymbol{x}^T(n)\boldsymbol{x}(n)}$$

and

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu\frac{e(n)\boldsymbol{x}(n)}{\boldsymbol{x}^T(n)\boldsymbol{x}(n) + \varepsilon}$$

Some Research Results

- M. H. Costa and J. C. M. Bermudez, "An improved model for the normalized LMS algorithm with Gaussian inputs and large number of coefficients," in Proc. of the 2002 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP-2002), Orlando, Florida, pp. II- 1385-1388, May 13-17, 2002.
- J. C. M. Bermudez and M. H. Costa, "A Statistical Analysis of the ε-NLMS and NLMS Algorithms for Correlated Gaussian Signals," Journal of the Brazilian Telecommunications Society, v. 20, n. 2, p. 7-13, 2005.
- G. Barrault, M. H. Costa, J. C. M. Bermudez and A. Lenzi, "A new analytical model for the NLMS algorithm," in Proc. 2005 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP 2005) Pennsylvania, USA, vol. IV, p. 41-44, 2005.
- J. C. M. Bermudez, N. J. Bershad and J.-Y. Tourneret, "An Affine Combination of Two NLMS Adaptive Filters - Transient Mean-Square Analysis," In Proc. Forty-Second Asilomar Conference on Asilomar Conference on Signals, Systems & Computers, Pacific Grove, CA, USA, 2008.
- P. Honeine, C. Richard, J. C. M. Bermudez and H. Snoussi, "Distributed prediction of time series data with kernels and adaptive filtering techniques in sensor networks," In Proc. Forty-Second Asilomar Conference on Asilomar Conference on Signals, Systems & Computers, Pacific Grove, CA, USA, 2008.

The Affine Projection Algorithm

- Consider the NLMS algorithm but using $\{\boldsymbol{x}(n), \boldsymbol{x}(n-1), \ldots, \boldsymbol{x}(n-P)\}$

$$\text{Minimize} \quad \|\Delta \boldsymbol{w}(n)\|^2 = \|\boldsymbol{w}(n+1) - \boldsymbol{w}(n)\|^2$$

$$\text{subject to:} \begin{cases} \boldsymbol{x}^T(n)\boldsymbol{w}(n+1) = d(n) \\ \boldsymbol{x}^T(n-1)\boldsymbol{w}(n+1) = d(n-1) \\ \quad \vdots \\ \boldsymbol{x}^T(n-P)\boldsymbol{w}(n+1) = d(n-P) \end{cases}$$

- Observation matrix

$$\boldsymbol{X}(n) = [\boldsymbol{x}(n), \boldsymbol{x}(n-1), \cdots, \boldsymbol{x}(n-P)]$$

- Desired signal vector

$$\boldsymbol{d}(n) = [d(n), d(n-1), \ldots, d(n-P)]$$

- Error vector

$$\boldsymbol{e}(n) = [e(n), e(n-1), \ldots, e(n-P)] = \boldsymbol{d}(n) - \boldsymbol{X}^T(n)\boldsymbol{w}(n)$$

- Vector of the constraint errors

$$\boldsymbol{e}_c(n) = \boldsymbol{d}(n) - \boldsymbol{X}^T(n)\boldsymbol{w}(n+1)$$

- Using Lagrange multipliers, we minimize

$$f[\boldsymbol{w}(n+1)] = \|\boldsymbol{w}(n+1) - \boldsymbol{w}(n)\|^2 + \boldsymbol{\lambda}^T[\boldsymbol{d}(n) - \boldsymbol{X}^T(n)\boldsymbol{w}(n+1)]$$

- Differentiating w.r.t. $\boldsymbol{w}(n+1)$ and equating the result to zero,

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \frac{1}{2}\boldsymbol{X}(n)\boldsymbol{\lambda} \qquad (*)$$

Using this result in $\boldsymbol{X}^T(n)\boldsymbol{w}(n+1) = \boldsymbol{d}(n)$ yields

$$\boldsymbol{\lambda} = 2[\boldsymbol{X}^T(n)\boldsymbol{X}(n)]^{-1}\boldsymbol{e}(n)$$

- Using this result in (*) yields

$$\boxed{\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu\boldsymbol{X}(n)[\boldsymbol{X}^T(n)\boldsymbol{X}(n)]^{-1}\boldsymbol{e}(n)}$$

Affine Projection
Solution to the Underdetermined Least-Squares Problem

- We want minimize

$$\|\boldsymbol{e}_c(n)\|^2 = \|\boldsymbol{d}(n) - \boldsymbol{X}^T(n)\boldsymbol{w}(n+1)\|^2$$

  where $\boldsymbol{X}^T(n)$ is $(P+1) \times N$ with $(P+1) < N$
- Thus, we look for the least-squares solution of the underdetermined system

$$\boldsymbol{X}^T(n)\boldsymbol{w}(n+1) = \boldsymbol{d}(n)$$

- The solution is

$$\boldsymbol{w}(n+1) = \left[\boldsymbol{X}^T(n)\right]^+ \boldsymbol{d}(n) = \boldsymbol{X}(n)[\boldsymbol{X}^T(n)\boldsymbol{X}(n)]^{-1}\boldsymbol{d}(n)$$

- Using $\boldsymbol{d}(n) = \boldsymbol{e}(n) + \boldsymbol{X}^T(n)\boldsymbol{w}(n)$ yields

$$\boxed{\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu\boldsymbol{X}(n)[\boldsymbol{X}^T(n)\boldsymbol{X}(n)]^{-1}\boldsymbol{e}(n)}$$

Observations:

- Order of the AP algorithm: $P+1$ ($P=0 \rightarrow$ NLMS)
- Convergence speed increases with P (but not linearly)
- Computational complexity increases with P (not linearly)
- If $\boldsymbol{X}^T(n)\boldsymbol{X}(n)$ is close to singular, use $\boldsymbol{X}^T(n)\boldsymbol{X}(n)+\varepsilon\boldsymbol{I}$
- The scalar error of NLMS becomes a vector error in AP (except for $\mu=1$)

Affine Projection – Stochastic Approximation of Regularized Newton

- Regularized Newton Algorithm

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu[\tilde{\varepsilon}\boldsymbol{I} + \boldsymbol{R}_{xx}]^{-1}[\boldsymbol{p} - \boldsymbol{R}_{xx}\boldsymbol{w}(n)]$$

- Estimates using time window statistics

$$\hat{\boldsymbol{p}} = \frac{1}{P+1}\sum_{k=n-P}^{n}\boldsymbol{X}(k)d(k) = \frac{1}{P+1}\boldsymbol{X}(n)\boldsymbol{d}(n)$$

$$\hat{\boldsymbol{R}}_{xx} = \frac{1}{P+1}\sum_{k=n-P}^{n}\boldsymbol{X}(k)\boldsymbol{X}^T(k) = \frac{1}{P+1}\boldsymbol{X}(n)\boldsymbol{X}^T(n)$$

- Using these estimates with $\tilde{\varepsilon} = \varepsilon/(P+1)$,

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu[\varepsilon\boldsymbol{I} + \boldsymbol{X}(n)\boldsymbol{X}^T(n)]^{-1}\boldsymbol{X}(n)\overbrace{[\boldsymbol{d}(n) - \boldsymbol{X}^T(n)\boldsymbol{w}(n)]}^{\boldsymbol{e}(n)}$$

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu[\varepsilon\boldsymbol{I} + \boldsymbol{X}(n)\boldsymbol{X}^T(n)]^{-1}\boldsymbol{X}(n)\boldsymbol{e}(n)$$

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu[\varepsilon \boldsymbol{I} + \boldsymbol{X}(n)\boldsymbol{X}^T(n)]^{-1}\boldsymbol{X}(n)\boldsymbol{e}(n)$$

- Using the Matrix Inversion Formula

$$[\varepsilon \boldsymbol{I} + \boldsymbol{X}(n)\boldsymbol{X}^T(n)]^{-1} = \boldsymbol{X}(n)[\varepsilon \boldsymbol{I} + \boldsymbol{X}^T(n)\boldsymbol{X}(n)]^{-1}$$

and

$$\boxed{\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu \boldsymbol{X}(n)[\varepsilon \boldsymbol{I} + \boldsymbol{X}^T(n)\boldsymbol{X}(n)]^{-1}\boldsymbol{e}(n)}$$

Affine Projection Algorithm as a Projection onto an Affine Subspace

- Conditions for the analysis
  - $\boldsymbol{e}(n) = \boldsymbol{d}(n) - \boldsymbol{X}^T(n)\boldsymbol{w}(n)$
  - $\boldsymbol{d}(n) = \boldsymbol{X}^T(n)\boldsymbol{w}^o$ defines the optimal solution in the least-squares sense
  - $\boldsymbol{v}(n) = \boldsymbol{w}(n) - \boldsymbol{w}^o$ (weight error vector)
- Error vector

$$\begin{aligned}\boldsymbol{e}(n) &= \boldsymbol{d}(n) - \boldsymbol{X}^T(n)[\boldsymbol{v}(n) + \boldsymbol{w}(n+1)] \\ &= \boldsymbol{X}^T(n)\boldsymbol{w}(n+1) - \boldsymbol{X}^T(n)[\boldsymbol{v}(n) + \boldsymbol{w}(n+1)] \\ &= -\boldsymbol{X}^T(n)\boldsymbol{v}(n)\end{aligned}$$

$$\boldsymbol{e}(n) = -\boldsymbol{X}^T(n)\boldsymbol{v}(n)$$

- *Interpretation:* To minimize the error $\boldsymbol{v}(n)$ should be orthogonal to all input vectors
  Restriction: We are going to use only $\{\boldsymbol{x}(n), \boldsymbol{x}(n-1), \ldots, \boldsymbol{x}(n-P)\}$
- *Iterative solution:*
  We can subtract from $\boldsymbol{v}(n)$ its projection onto the range of $\boldsymbol{X}(n)$ at each iteration

$$\boldsymbol{v}(n+1) = \boldsymbol{v}(n) - \boldsymbol{P}_{\boldsymbol{X}(n)}\boldsymbol{v}(n) \quad \text{(Proj. onto an affine subspace)}$$

- Using $\boldsymbol{X}^T(n)\boldsymbol{v}(n) = -\boldsymbol{e}(n)$

$$\begin{aligned}\boldsymbol{v}(n+1) &= \boldsymbol{v}(n) - \left\{\boldsymbol{X}(n)[\boldsymbol{X}^T(n)\boldsymbol{X}(n)]^{-1}\boldsymbol{X}^T(n)\right\}\boldsymbol{v}(n) \\ &= \boldsymbol{v}(n) + \mu\boldsymbol{X}(n)[\boldsymbol{X}^T(n)\boldsymbol{X}(n)]^{-1}\boldsymbol{e}(n) \quad \text{(AP algorithm)}\end{aligned}$$

Pseudo Affine Projection Algorithm

- Major problem with the AP algorithm
  For $\mu \neq 1$, $e(n) \rightarrow \boldsymbol{e}(n)$ ➪ large computational complexity
- The Pseudo-AP algorithm replaces the input $x(n)$ with its P-th order autoregressive prediction

$$\hat{\boldsymbol{x}}(n) = \sum_{k=1}^{P} a_k \boldsymbol{x}(n-k)$$

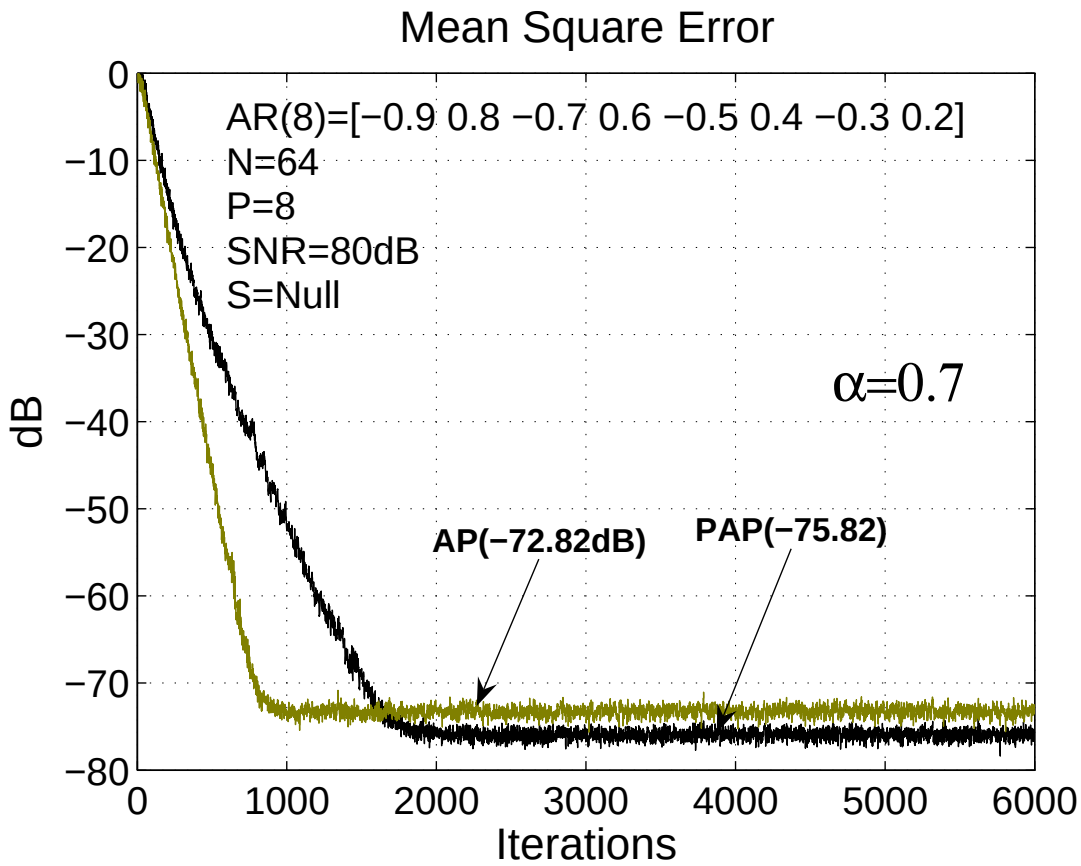
- Using the least-squares solution for $\boldsymbol{a} = [a_1, a_2, \ldots, a_P]^T$,

$$\boldsymbol{a} = [\boldsymbol{X}_p^T(n)\boldsymbol{X}_p(n)]^{-1}\boldsymbol{X}_p^T(n)\boldsymbol{x}(n), \quad \boldsymbol{X}_p(n) = [\boldsymbol{x}(n-1), \ldots, \boldsymbol{x}(n-P)]$$

- Now, we subtract from $\boldsymbol{x}(n)$ its projections onto the last P input vectors

$$\begin{aligned}\boldsymbol{\phi}(n) &= \boldsymbol{x}(n) - \boldsymbol{X}_p(n)\boldsymbol{a}(n) \\ &= \boldsymbol{x}(n) - \left\{\boldsymbol{X}_p(n)[\boldsymbol{X}_p^T(n)\boldsymbol{X}_p(n)]^{-1}\boldsymbol{X}_p^T(n)\right\}\boldsymbol{x}(n) \\ &= (\boldsymbol{I} - \boldsymbol{P}_P)\boldsymbol{x}(n); \qquad (\boldsymbol{P}_P\text{: projection matrix})\end{aligned}$$

Example – Pseudo-AP *versus* AP



- Using ϕ as the new input vector for the NLMS algorithm,

$$\boxed{\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu \frac{\boldsymbol{\phi}(n)}{\boldsymbol{\phi}^T(n)\boldsymbol{\phi}(n)} e(n)}$$

- It can be shown that Pseudo-AP is identical to AP for an AR input and $\mu = 1$
- Otherwise, Pseudo-AP is different from AP
- Simpler to implement than AP for $\mu \neq 1$
- Can lead even to better steady-state results than AP for AR inputs
- For AR inputs: NLMS with input "orthogonalization"

Some Research Results

- S. J. M. de Almeida, J. C. M. Bermudez, N. J. Bershad and M. H. Costa, "A statistical analysis of the affine projection algorithm for unity step size and autoregressive inputs," IEEE Transactions on Circuits and Systems - I, vol. 52, pp. 1394-1405, July 2005.
- S. J. M. Almeida, J. C. M. Bermudez and N. J. Bershad, "A Stochastic Model for a Pseudo Affine Projection Algorithm," IEEE Transactions on Signal Processing, v. 57, p. 107-118, 2009.
- S. J. M. Almeida, M. H. Costa and J. C. M. Bermudez, "A Stochastic Model for the Deficient Length Pseudo Affine Projection Adaptive Algorithm," In Proc. 17th European Signal Processing Conference (EUSIPCO), Aug. 24-28, Glasgow, Scotland, 2009.
- S. J. M. Almeida, M. H. Costa and J. C. M. Bermudez, "A stochastic model for the deficient order affine projection algorithm," in Proc. ISSPA 2010, Kuala Lumpur, Malaysia, May 2010.
- C. Richard, J. C. M. Bermudez and P. Honeine, "Online Prediction of Time Series Data With Kernels," IEEE Transactions on Signal Processing, v. 57, p. 1058-1067, 2009.
- P. Honeine, C. Richard, J. C. M. Bermudez, H. Snoussi, M. Essoloh and F. Vincent, "Functional estimation in Hilbert space for distributed learning in wireless sensor networks," In Proc. IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), Taipei, Taiwan, p. 2861-2864, 2009.

Deterministic Algorithms

Recursive Least Squares Algorithm (RLS)

- Based on a deterministic philosophy
- Designed for the present realization of the input signals
- Least squares method adapted for real time processing of temporal series
- Convergence speed is not strongly dependent on the input statistics

RLS – Problem Definition

- Define

$$\boldsymbol{x}_o(n) = [x(n), x(n-1), \ldots, x(0)]^T$$
$$\boldsymbol{x}(n) = [x(n), x(n-1), \ldots, x(n-N+1)]^T \quad \text{input to the } N\text{-tap filter}$$

- Desired signal $d(n)$
- Estimator of $d(n)$: $y(n) = \boldsymbol{x}^T(n)\boldsymbol{w}(n)$
- Cost function – Squared Error

$$J_{ls}(n) = \sum_{k=0}^{n} e^2(k) = \sum_{k=0}^{n} [d(k) - \boldsymbol{x}^T(k)\boldsymbol{w}(n)]^2$$
$$= \boldsymbol{e}^T(n)\boldsymbol{e}(n), \quad \boldsymbol{e}(n) = [e(n), e(n-1), \ldots, e(0)]^T$$

In vector form

$$\boldsymbol{d}(n) = [d(n), d(n-1), \ldots, d(0)]^T$$

$$\boldsymbol{y}(n) = [y(n), y(n-1), \ldots, y(0)]^T, \quad y(k) = \boldsymbol{x}^T(k)\boldsymbol{w}(n)$$

$$\boldsymbol{y}(n) = \sum_{k=0}^{N-1} w_k(n)\boldsymbol{x}_o(n-k) = \boldsymbol{\Theta}(n)\boldsymbol{w}(n)$$

$$\boldsymbol{\Theta}(n) = \begin{bmatrix} x(n) & x(n-1) & \cdots & x(n-N+1) \\ x(n-1) & x(n-2) & \cdots & x(n-N) \\ \vdots & \vdots & \vdots & \vdots \\ x(0) & x(-1) & \cdots & x(-N+1) \end{bmatrix}$$

$$= [\boldsymbol{x}_o(n), \boldsymbol{x}_o(n-1), \ldots \boldsymbol{x}_o(n-N+1)]$$

$$\boxed{\boldsymbol{e}(n) = \boldsymbol{d}(n) - \boldsymbol{\Theta}(n)\boldsymbol{w}(n)}$$

$$\boldsymbol{e}(n) = \boldsymbol{d}(n) - \boldsymbol{\Theta}(n)\boldsymbol{w}(n)$$

and we minimize

$$J_{ls}(n) = \|\boldsymbol{e}(n)\|^2$$

- Normal Equations

$$\boldsymbol{\Theta}^T(n)\boldsymbol{\Theta}(n)\boldsymbol{w}(n) = \boldsymbol{\Theta}^T(n)\boldsymbol{d}(n)$$

$$\tilde{\boldsymbol{R}}(n)\boldsymbol{w}(n) = \tilde{\boldsymbol{p}}(n)$$

- Alternative representation for $\tilde{\boldsymbol{R}}$ and $\tilde{\boldsymbol{p}}$

$$\tilde{\boldsymbol{R}}(n) = \boldsymbol{\Theta}^T(n)\boldsymbol{\Theta}(n) = \sum_{k=0}^{n} \boldsymbol{x}(k)\boldsymbol{x}^T(k)$$

$$\tilde{\boldsymbol{p}}(n) = \boldsymbol{\Theta}^T(n)\boldsymbol{d}(n) = \sum_{k=0}^{n} \boldsymbol{x}(k)d(k)$$

RLS – Observations

- $\boldsymbol{w}(n)$ remains fixed for $0 \leq k \leq n$ to determine $J_{ls}(n)$
- Optimum vector $\boldsymbol{w}(n)$ is $\boldsymbol{w}^o(n) = \tilde{\boldsymbol{R}}^{-1}(n)\tilde{\boldsymbol{p}}$
- The algorithm has growing memory
- In an adaptive implementation,

$$\boldsymbol{w}^o(n) = \tilde{\boldsymbol{R}}^{-1}(n)\tilde{\boldsymbol{p}}$$

  must be solved for each iteration n
- Problems for an adaptive implementation
  - Dificulties with nonstationary signals (infinite data window)
  - Computational complexity

RLS - Forgetting the Past

- Modified cost function

$$J_{ls}(n) = \sum_{k=0}^{n} \lambda^{n-k} e^2(k) = \sum_{k=0}^{n} \lambda^{n-k} [d(k) - \boldsymbol{x}^T(k)\boldsymbol{w}(n)]^2$$
$$= \boldsymbol{e}^T(n)\boldsymbol{\Lambda}\boldsymbol{e}(n), \quad \boldsymbol{\Lambda} = \text{diag}[1, \lambda, \lambda^2, \ldots, \lambda^n]$$

- Modified Normal Equations

$$\boldsymbol{\Theta}^T(n)\boldsymbol{\Lambda}\boldsymbol{\Theta}(n)\boldsymbol{w}(n) = \boldsymbol{\Theta}^T(n)\boldsymbol{\Lambda}\boldsymbol{d}(n)$$

$$\hat{\boldsymbol{R}}(n)\boldsymbol{w}(n) = \hat{\boldsymbol{p}}(n)$$

where

$$\hat{\boldsymbol{R}}(n) = \boldsymbol{\Theta}^T(n)\boldsymbol{\Lambda}\boldsymbol{\Theta}(n) = \sum_{k=0}^{n} \lambda^{n-k} \boldsymbol{x}(k)\boldsymbol{x}^T(k)$$

$$\hat{\boldsymbol{p}}(n) = \boldsymbol{\Theta}^T(n)\boldsymbol{\Lambda}\boldsymbol{d}(n) = \sum_{k=0}^{n} \lambda^{n-k} \boldsymbol{x}(k)d(k)$$

RLS – Recursive Updating

- Correlation Matrix

$$\begin{aligned}\hat{\boldsymbol{R}}(n) &= \sum_{k=0}^{n} \lambda^{n-k} \boldsymbol{x}(k)\boldsymbol{x}^T(k) \\ &= \sum_{k=0}^{n-1} \lambda^{n-k} \boldsymbol{x}(k)\boldsymbol{x}^T(k) + \boldsymbol{x}(n)\boldsymbol{x}^T(n) \\ &= \lambda \hat{\boldsymbol{R}}(n-1) + \boldsymbol{x}(n)\boldsymbol{x}^T(n)\end{aligned}$$

- Cross-correlation vector

$$\begin{aligned}\hat{\boldsymbol{p}}(n) &= \sum_{k=0}^{n} \lambda^{n-k} \boldsymbol{x}(k)d(k) \\ &= \sum_{k=0}^{n-1} \lambda^{n-k} \boldsymbol{x}(k)d(k) + \boldsymbol{x}(n)d(n) \\ &= \lambda \hat{\boldsymbol{p}}(n-1) + \boldsymbol{x}(n)d(n)\end{aligned}$$

- We have recursive updating expressions for $\hat{\boldsymbol{R}}(n)$ and $\hat{\boldsymbol{p}}(n)$
- However, we need a recursive updating for $\hat{\boldsymbol{R}}^{-1}(n)$, as

$$\boldsymbol{w}^o(n) = \hat{\boldsymbol{R}}^{-1}(n)\hat{\boldsymbol{p}}(n)$$

- Applying the matrix inversion lemma to

$$\hat{\boldsymbol{R}}(n) = \lambda\hat{\boldsymbol{R}}(n-1) + \boldsymbol{x}(n)\boldsymbol{x}^T(n)$$

and defining $\boldsymbol{P}(n) = \hat{\boldsymbol{R}}^{-1}(n)$ yields

$$\boldsymbol{P}(n) = \lambda^{-1}\boldsymbol{P}(n-1) - \frac{\lambda^{-2}\boldsymbol{P}(n-1)\boldsymbol{x}(n)\boldsymbol{x}^T(n)\boldsymbol{P}(n-1)}{1+\lambda^{-1}\boldsymbol{x}^T(n)\boldsymbol{P}(n-1)\boldsymbol{x}(n)}$$

The Gain Vector

- Definition

$$\boldsymbol{k}(n) = \frac{\lambda^{-1}\boldsymbol{P}(n-1)\boldsymbol{x}(n)}{1+\lambda^{-1}\boldsymbol{x}^T(n)\boldsymbol{P}(n-1)\boldsymbol{x}(n)}$$

- Using this definition

$$\boldsymbol{P}(n) = \lambda^{-1}\boldsymbol{P}(n-1) - \lambda^{-1}\boldsymbol{k}(n)\boldsymbol{x}^T(n)\boldsymbol{P}(n-1)$$

- $\boldsymbol{k}(n)$ can be written as

$$\begin{aligned}\boldsymbol{k}(n) &= \overbrace{\left[\lambda^{-1}\boldsymbol{P}(n-1) - \lambda^{-1}\boldsymbol{k}(n)\boldsymbol{x}^T(n)\boldsymbol{P}(n-1)\right]}^{\boldsymbol{P}(n)}\boldsymbol{x}(n) \\ &= \hat{\boldsymbol{R}}^{-1}(n)\boldsymbol{x}(n)\end{aligned}$$

➪ $\boldsymbol{k}(n)$ is the repres. of $\boldsymbol{x}(n)$ on the column space of $\hat{\boldsymbol{R}}(n)$

RLS – Recursive Weight Update

- $\boldsymbol{w}(n+1) = \boldsymbol{w}^o(n)$
- Using

$$\begin{aligned}
\boldsymbol{w}(n+1) &= \hat{\boldsymbol{R}}^{-1}(n)\hat{\boldsymbol{p}}(n) = \boldsymbol{P}(n)\hat{\boldsymbol{p}}(n) \\
\hat{\boldsymbol{p}}(n) &= \lambda\hat{\boldsymbol{p}}(n-1) + \boldsymbol{x}(n)d(n) \\
\boldsymbol{P}(n) &= \lambda^{-1}\boldsymbol{P}(n-1) - \lambda^{-1}\boldsymbol{k}(n)\boldsymbol{x}^T(n)\boldsymbol{P}(n-1) \\
\boldsymbol{k}(n) &= \boldsymbol{P}(n)\boldsymbol{x}(n) \\
e(n) &= d(n) - \boldsymbol{x}^T(n)\boldsymbol{w}(n)
\end{aligned}$$

yields

$$\boxed{\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \boldsymbol{k}(n)e(n)}$$ RLS weight update

RLS – Observations

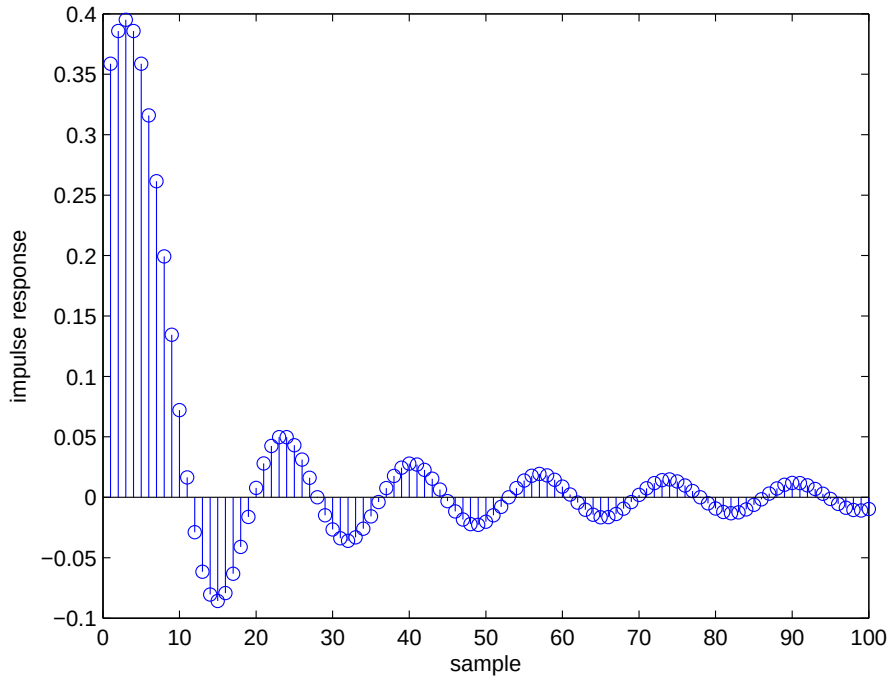
- The RLS convergence speed is not affected by the eigenvalues of $\hat{\boldsymbol{R}}(n)$
- Initialization: $\hat{\boldsymbol{p}}(0) = \boldsymbol{0}$, $\hat{\boldsymbol{R}}(0) = \delta \boldsymbol{I}$, $\delta \simeq (1-\lambda)\sigma_x^2$
- This results in $\hat{\boldsymbol{R}}(n)$ changed to $\lambda^n \delta \boldsymbol{I} + \hat{\boldsymbol{R}}(n)$
  ➪ Biased estimate of $\boldsymbol{w}^o(n)$ for small n
  (No problem for $\lambda < 1$ and large n)
- Numerical problems in finite precision
  - Unstable for $\lambda = 1$
  - Loss of symmetry in $\boldsymbol{P}(n)$
    - ★ Evaluate only upper (lower) part and diagonal of $\boldsymbol{P}(n)$
    - ★ Replace $\boldsymbol{P}(n)$ with $[\boldsymbol{P}(n) + \boldsymbol{P}^T(n)]/2$ after updating from $P(n-1)$
- Numerical problems when $\boldsymbol{x}(n) \to \boldsymbol{0}$ and $\lambda < 1$

Some Research Results

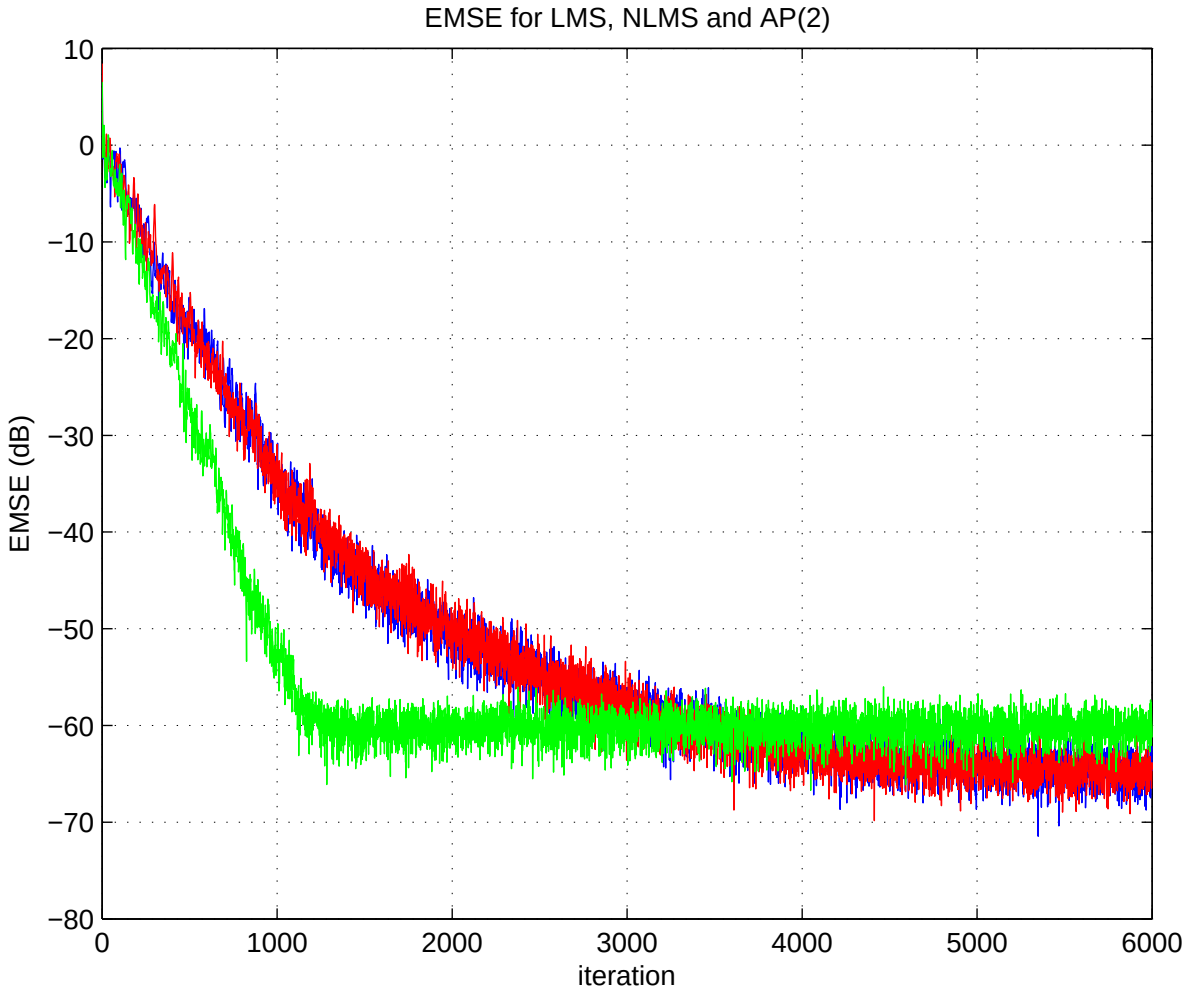
- C. Ludovico and J. C. M. Bermudez, "A recursive least squares algorithm robust to low- power excitation," in. Proc. 2004 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP-2004), Montreal, Canada, vol. II, p. 673-677, 2004.
- C. Ludovico and J. C. M. Bermudez, "An improved recursive least squares algorithm robust to input power variation," in Proc. IEEE Statistical Signal Processing Workshop, Bordeaux, France, 2005.

Performance Comparison

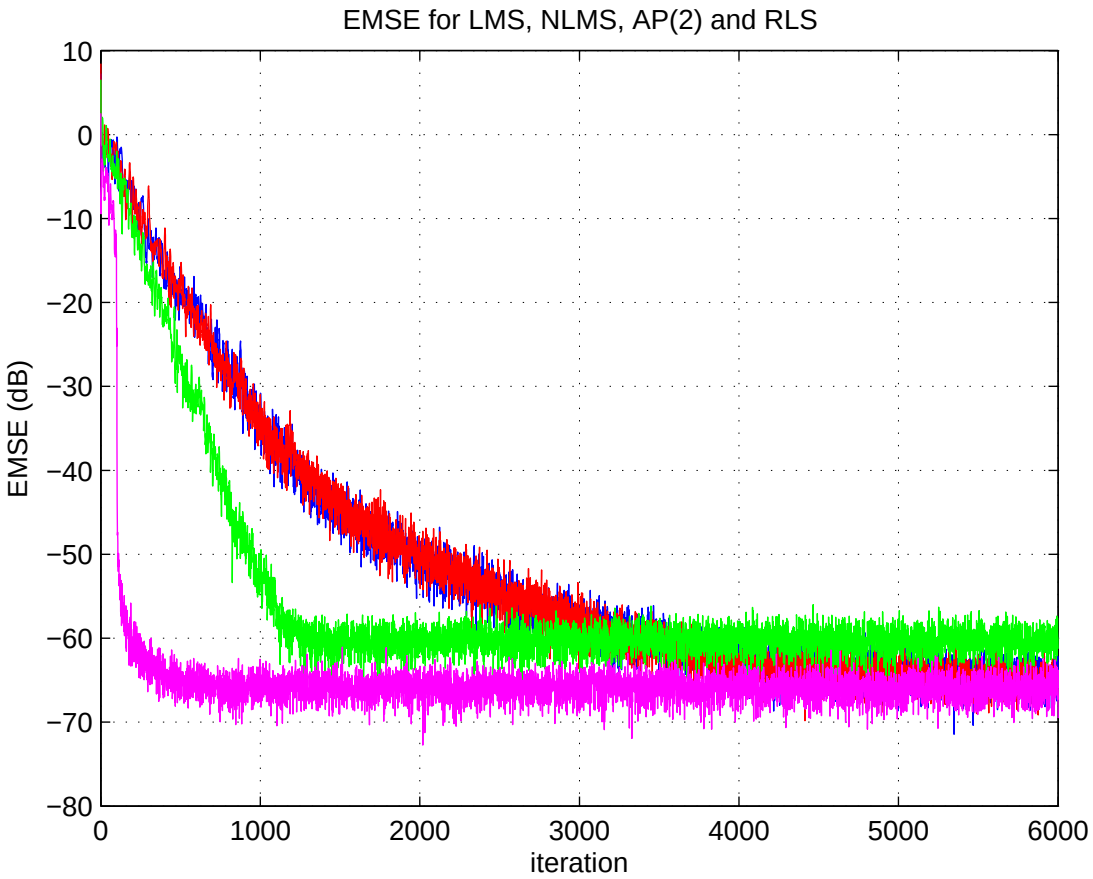
- System identification, N=100
- Input – AR(1) $\quad x(n) = 0.9x(n-1) + v(n)$
- AP algorithm with $P = 2$
- Step sizes designed for equal LMS and NLMS performances
- Impulse response to be estimated



Excess Mean Square Error - LMS, NLMS and AP



Excess Mean Square Error - LMS, NLMS, AP and RLS



Performance Comparisons
Computational Complexity

- Adaptive filter with N real coefficients and real signals
- For the AP algorithm, $K = P + 1$

Algorithm	$\times$	$+$	$/$
LMS	$2N + 1$	$2N$	
NLMS	$3N + 1$	$3N$	1
AP	$(K^2 + 2K)N + K^3 + K$	$(K^2 + 2K)N + K^3 + K^2$	
RLS	$N^2 + 5N + 1$	$N^2 + 3N$	1

For $N = 100$, $P = 2$

Algorithm	$\times$	$+$	$/$	$\simeq$ factor
LMS	201	200		1
NLMS	301	300	1	1.5
AP	1,530	1,536		7.5
RLS	10,501	10,300	1	52.5

Typical values for acoustic echo cancellation ($N = 1024,\ P = 2$)

Algorithm	$\times$	$+$	$/$	$\simeq$ factor
LMS	$2,049$	$2,048$		1
NLMS	$3,073$	$3,072$	1	1.5
AP	$15,390$	$15,396$		7.5
RLS	$1,053,697$	$1,051,648$	1	514

How to Deal with Computational Complexity?

- Not an easy task!!!
- There are "fast" versions for some algorithms (especially RLS)
- What is usually not said is that ... speed can bring
  - Instability
  - Increased need for memory

  ➪ *Most applications rely on simple solutions*

Understanding the Adaptive Filter Behavior

What are Adaptive Filters?

Adaptive filters are, by design, systems that are

- Time-variant ($\boldsymbol{w}(n)$ is time-variant)
- Nonlinear ($y(n)$ is a nonlinear function of $\boldsymbol{x}(n)$)
- Stochastic ($\boldsymbol{w}(n)$ is random)

LMS:

$$\begin{aligned}\boldsymbol{w}(n+1) &= \boldsymbol{w}(n) + \mu e(n)\boldsymbol{x}(n) \\ &= \left[\boldsymbol{I} - \mu\, \boldsymbol{x}(n)\boldsymbol{x}^T(n)\right]\boldsymbol{x}(n) + \mu\, d(n)\boldsymbol{x}(n) \\ y(n) &= \boldsymbol{x}^T(n)\boldsymbol{w}(n)\end{aligned}$$

- They are difficult to understand
- Different applications and signals require different analyses
- Simplifying assumptions are necessary

➪ *Good design requires good analytical models.*

LMS Analysis

Basic equations:

$$
\begin{aligned}
d(n) &= \boldsymbol{x}^T(n)\boldsymbol{w}^o + z(n) \\
e(n) &= d(n) - \boldsymbol{x}^T(n)\boldsymbol{w}(n) \\
\boldsymbol{w}(n+1) &= \boldsymbol{w}(n) + \mu\, e(n)\boldsymbol{x}(n) \\
\boldsymbol{v}(n) &= \boldsymbol{w}(n) - \boldsymbol{w}^o \quad \text{(study about the optimum weight vector)}
\end{aligned}
$$

Mean Weight Behavior:

Neglecting dependence between $\boldsymbol{v}(n)$ *and* $\boldsymbol{x}(n)\boldsymbol{x}^T(n)$,

$$E[\boldsymbol{v}(n+1)] = (\boldsymbol{I} - \mu\boldsymbol{R}_{xx})E[\boldsymbol{v}(n)]$$

- Follows the steepest descent trajectory
- Convergence condition: $0 < \mu < 2/\lambda_{\max}$
- $\lim_{n\to\infty} E[\boldsymbol{w}(n)] = \boldsymbol{w}^o$

However: $\boldsymbol{w}(n)$ is random

➪ *We need to study its fluctuations about* $E[\boldsymbol{w}(n)] \rightarrow$ MSE

The Mean-Square Estimation Error

$$e(n) = e_o(n) - \boldsymbol{x}^T(n)\boldsymbol{v}(n), \quad e_o(n) = d(n) - \boldsymbol{x}^T(n)\boldsymbol{w}^o$$

- Square the error equation
- Take the expected value
- Neglect the statistical dep. between $\boldsymbol{v}(n)$ and $\boldsymbol{x}(n)\boldsymbol{x}^T(n)$

$$J_{ms}(n) = E[e_o^2(n)] + Tr\{\boldsymbol{R}_{xx}\boldsymbol{K}(n)\}, \quad \boldsymbol{K}(n) = E[\boldsymbol{v}(n)\boldsymbol{v}^T(n)]$$

- This expression is independent of the adaptive algorithm
- Effect of the alg. on $J_{ms}(n)$ is determined by $\boldsymbol{K}(n)$
- $J_{ms_{\min}} = E[e_o^2(n)]$
- $J_{ms_{\text{ex}}} = Tr\{\boldsymbol{R}_{xx}\boldsymbol{K}(n)\}$

The Behavior of $\boldsymbol{K}(n)$ for LMS

- Using the basic equations

$$\boldsymbol{v}(n+1) = \left[\boldsymbol{I} + \mu\, \boldsymbol{x}(n)\boldsymbol{x}^T(n)\right] \boldsymbol{v}(n) + \mu\, e_o(n)\boldsymbol{x}(n)$$

- Post-multiply by $\boldsymbol{v}^T(n+1)$
- Take the expected value assuming $\boldsymbol{v}(n)$ and $\boldsymbol{x}(n)\boldsymbol{x}^T(n)$ independent
- Evaluate the expectations
  - Higher order moments of $x(n)$ require input pdf
- Assuming Gaussian inputs

$$\begin{aligned}\boldsymbol{K}(n+1) =& \boldsymbol{K}(n) - \mu\, [\boldsymbol{R}_{xx}\boldsymbol{K}(n) + \boldsymbol{K}(n)\boldsymbol{R}_{xx}] \\ &+ \mu^2\, \{\boldsymbol{R}_{xx}\mathsf{Tr}[\boldsymbol{R}_{xx}\boldsymbol{K}(n)] + 2\boldsymbol{R}_{xx}\boldsymbol{K}(n)\boldsymbol{R}_{xx}\} \\ &+ \mu^2\, \boldsymbol{R}_{xx} J_{ms_{\min}}\end{aligned}$$

Design Guidelines Extrated from the Model

- Stability

$$0 < \mu < \frac{m}{\mathsf{Tr}[\boldsymbol{R}_{xx}]}; \quad m = 2 \text{ or } m = 2/3$$

- $J_{ms}(n)$ converges as a function of

$$\left[(1-\mu\lambda_i)^2 + 2\mu^2\lambda_i^2\right]^n, \quad i = 1, \ldots, N$$

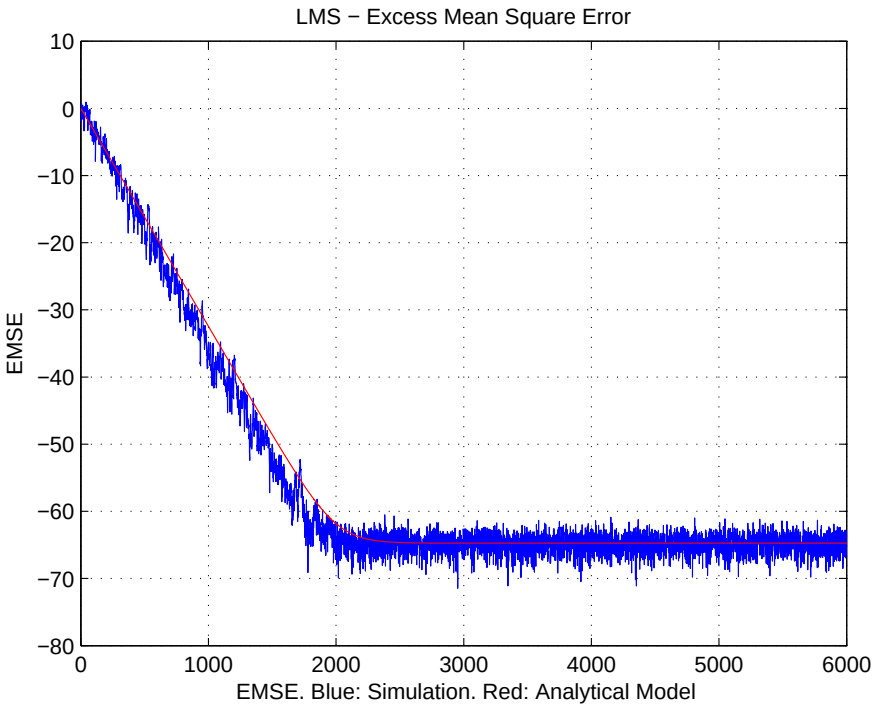
- Steady-State

$$J_{ms_{\text{ex}}}(\infty) \simeq \frac{\mu}{2}\mathsf{Tr}[\boldsymbol{R}_{xx}]J_{ms_{\min}}$$

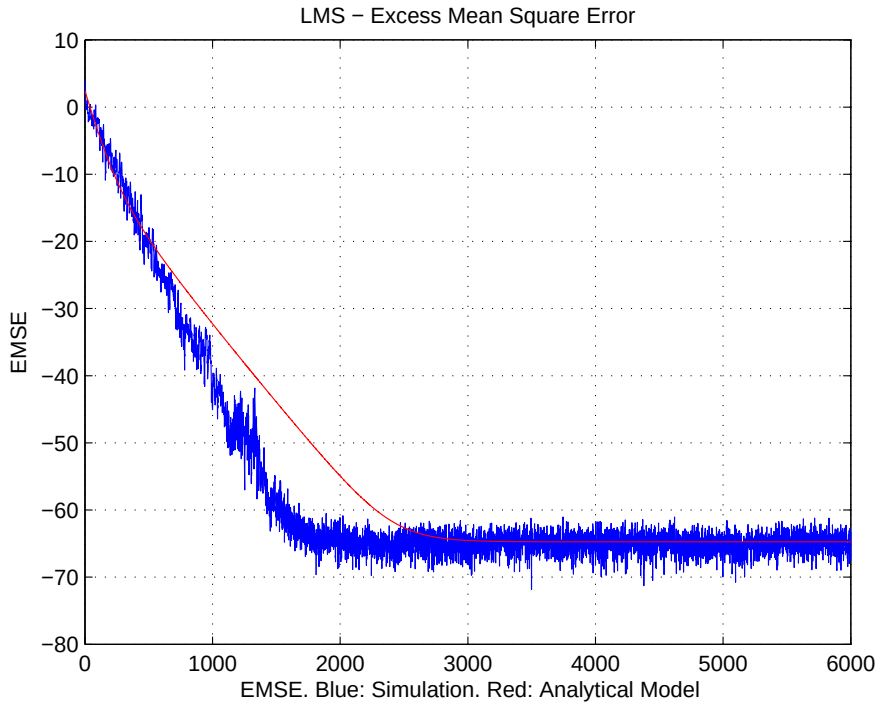
$$\mathcal{M}(\infty) = \frac{J_{ms_{\text{ex}}}(\infty)}{J_{ms_{\min}}} \quad \text{(MSE Misadjustment)}$$

Model Evaluation

$$x(n) = \alpha\, x(n-1) + v(n)$$



Input: White noise



Input: AR(1), $\alpha = 0.8$

- Linear system identification - FIR with 100 coefficients
- Step-size $\mu = 0.05$
- Noise power $\sigma_v^2 = 10^{-6}$

Merci pour votre attention!!!