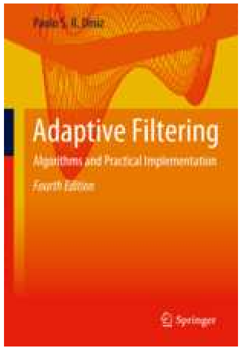
LMS-Based Algorithms

Paulo S. R. Diniz - *diniz@lps.ufrj.br*



SIGNAL PROCESSING LABORATORY

COPPE/Poli - Federal University of Rio de Janeiro

October 2012

slidec44ed.tex

LMS-Based Algorithms

- The Quantized-Error Algorithms
- The LMS-Newton Algorithm
- The Normalized LMS Algorithm
- The Transform-Domain LMS Algorithm
- The Affine Projection Algorithm
- Simulation Examples

Quantized-Error Algorithms

The quantized error algorithm updates the coefficients according to

$$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\mu Q[e(k)]\mathbf{x}(k) \tag{1}$$

where $Q[.]$ represents a quantization operation.

The quantization of the error implies a modification of the objective function to be minimized. In a gradient algorithm

$$\begin{aligned} \mathbf{w}(k+1) &= \mathbf{w}(k) - \mu \frac{\partial F(e(k))}{\partial \mathbf{w}(k)} \\ &= \mathbf{w}(k) - \mu \frac{\partial F(e(k))}{\partial e(k)} \frac{\partial e(k)}{\partial \mathbf{w}(k)} \end{aligned} \tag{2}$$

The LMS-Based algorithms

For a linear combiner the above equation can be rewritten as

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu \frac{\partial F(e(k))}{\partial e(k)} \mathbf{x}(k) \tag{3}$$

The objective function that is minimized is such that

$$\frac{\partial F(e(k))}{\partial e(k)} = 2Q[e(k)] \tag{4}$$

$F(e(k))$ is obtained by integrating $2Q[e(k)]$ with respect to $e(k)$. The chain rule applied in eq. (3) is not valid at the points of discontinuity of $Q[\dot{]}$.

Sign-Error Algorithm

The simplest form for the quantization is the sign (sgn)

$$\operatorname{sgn}[b] = \begin{cases} 1, & b > 1 \\ 0, & b = 0 \\ -1, & b < 0 \end{cases} \tag{5}$$

Sign-Error Algorithm

The coefficient vector updating is performed by

$$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\mu \ \mathrm{sgn}[e(k)] \ \mathbf{x}(k) \tag{6}$$

The objective function minimized is

$$F[e(k)] = 2|e(k)| \tag{7}$$

Sign-Error Algorithm

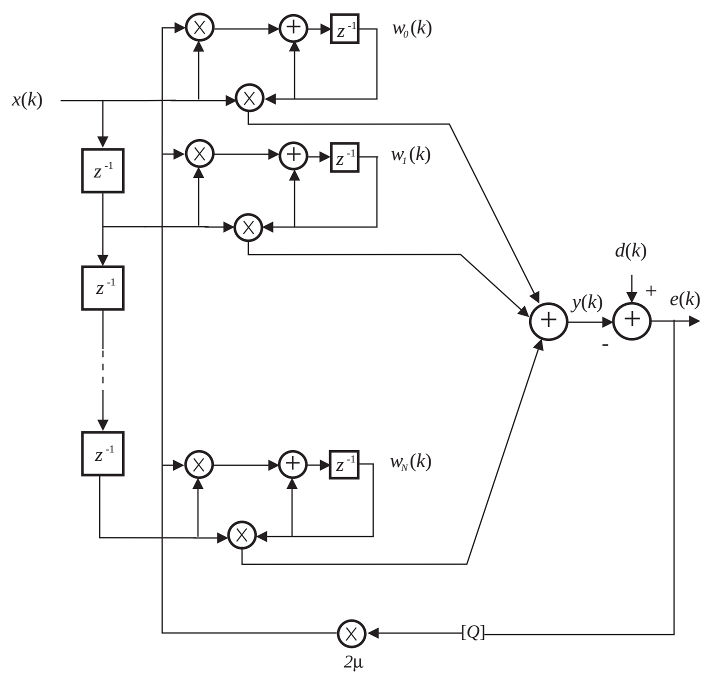


Figure 1: Sign-error adaptive FIR filter: $Q[e(k)] = \mathrm{sgn}[e(k)]$.

Sign-Error Algorithm

Algorithm 4.1

Sign-Error Algorithm

Initialization

$\mathbf{x}(0) = \mathbf{w}(0) = [0 \ldots 0]^T$

Do for $k \geq 0$

$e(k) = d(k) - \mathbf{x}^T(k)\mathbf{w}(k)$

$\rho = \text{sgn}[e(k)]$

$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\mu\rho\mathbf{x}(k)$

Quantized-Error Algorithms

Behavior of the Coefficient Vector

The sign error algorithm can be described by

$$\Delta\mathbf{w}(k+1) = \Delta\mathbf{w}(k) + 2\mu \, \text{sgn}(e(k)) \; \mathbf{x}(k) \tag{8}$$

where $\Delta\mathbf{w}(k) = \mathbf{w}(k) - \mathbf{w}_o$.

The expected value of the coefficient vector error is

$$E[\Delta\mathbf{w}(k+1)] = E[\Delta\mathbf{w}(k)] + 2\mu E[\text{sgn}(e(k)) \; \mathbf{x}(k)] \tag{9}$$

The convergence depends on the pdf of $n(k)$. Even if the error signal becomes very small, the coefficients will be updated due to the sign function.

Sign-Error Algorithm

If $d(k)$ and $\mathbf{x}(k)$ are zero mean and jointly Gaussian, and $n(k)$ is zero mean, Gaussian and independent of $\mathbf{x}(k)$ and $d(k)$, the error signal will be a zero mean Gaussian signal conditioned on $\Delta\mathbf{w}(k)$. In this case

$$E\{\text{sgn}(e(k))\ \mathbf{x}(k)\} \approx \sqrt{\frac{2}{\pi\xi(k)}} E[\mathbf{x}(k)e(k)] \tag{10}$$

where $\xi(k)$ is the variance of $e(k)$, and the approx. is valid for small values of μ. For large μ, $e(k)$ is dependent on $\Delta\mathbf{w}(k)$

Quantized-Error Algorithms

Applying (10) in (9) and replacing $e(k)$ by $e_o(k) - \Delta\mathbf{w}^T(k)\mathbf{x}(k)$

$$\begin{aligned} E[\Delta\mathbf{w}(k+1)] &= \left\{\mathbf{I} - 2\mu\sqrt{\frac{2}{\pi\xi(k)}}E[\mathbf{x}(k)\mathbf{x}^T(k)]\right\} E[\Delta\mathbf{w}(k)] \\ &+ 2\mu\sqrt{\frac{2}{\pi\xi(k)}}\ E[e_o(k)\mathbf{x}(k)] \qquad (11) \end{aligned}$$

Using orthogonality principle

$$E[\Delta\mathbf{w}(k+1)] = \left(\mathbf{I} - 2\mu\sqrt{\frac{2}{\pi\xi(k)}}\mathbf{R}\right) E[\Delta\mathbf{w}(k)] \qquad (12)$$

Sign-Error Algorithm

The coef. of the adaptive filter converge in the mean, if

$$0 < \mu < \frac{1}{\lambda_{max}} \sqrt{\frac{\pi \xi(k)}{2}} \tag{13}$$

where λ_{max} is the largest eigenvalue of $\mathbf{R}$. A more practical range for μ is given by

$$0 < \mu < \frac{1}{tr[\mathbf{R}]} \sqrt{\frac{\pi \xi(k)}{2}} \tag{14}$$

Quantized-Error Algorithms

Coefficient-Error Covariance Matrix

The difference equation for $\text{cov}[\Delta\mathbf{w}(k)]$ is

$$\begin{aligned}\text{cov}[\Delta\mathbf{w}(k+1)] &= \text{cov}[\Delta\mathbf{w}(k)] + 2\mu E[\text{sgn}(e(k))\mathbf{x}(k)\Delta\mathbf{w}^T(k)] \\ &+ 2\mu E[\text{sgn}(e(k))\Delta\mathbf{w}(k)\mathbf{x}^T(k)] + 4\mu^2\mathbf{R} \qquad (15)\end{aligned}$$

Quantized-Error Algorithms

The terms with expected values in the equation above can be expressed as

$$E[\mathrm{sgn}(e(k))\mathbf{x}(k)\Delta\mathbf{w}^T(k)] \approx -\sqrt{\frac{2}{\pi\xi(k)}}\mathbf{R}\mathrm{cov}[\Delta\mathbf{w}(k)] \qquad (16)$$

and

$$E[\mathrm{sgn}(e(k))\Delta\mathbf{w}(k)x^T(k)] \approx -\sqrt{\frac{2}{\pi\xi(k)}}\mathrm{cov}[\Delta\mathbf{w}(k)]\mathbf{R} \qquad (17)$$

Quantized-Error Algorithms

It then follows that

$$\mathbf{v}'(k+1) = \left(\mathbf{I} - 4\mu\sqrt{\frac{2}{\pi\xi(k)}}\,\mathbf{\Lambda}\right)\,\mathbf{v}'(k) + 4\mu^2\boldsymbol{\lambda} \tag{18}$$

The value of μ must be in the range

$$0 < \mu < \frac{1}{2\lambda_{max}}\sqrt{\frac{\pi\xi(k)}{2}} \tag{19}$$

Quantized-Error Algorithms

A more severe and practical range for μ is

$$0 < \mu < \frac{1}{2tr[\mathbf{R}]}\sqrt{\frac{\pi\xi(k)}{2}} \tag{20}$$

For $k \rightarrow \infty$ each element of $\mathbf{v}$'(k) tends to

$$v_i(\infty) = \mu\sqrt{\frac{\pi\xi(\infty)}{2}} \tag{21}$$

Quantized-Error Algorithms

Excess of MSE and Misadjustment

The excess of MSE can be expressed as

$$\Delta\xi(k) = \sum_{i=0}^{N} \lambda_i v_i(k) = \boldsymbol{\lambda}^T \mathbf{v}'(k) \tag{22}$$

Quantized-Error Algorithms

Substituting (21) in (22), it yields

$$\begin{aligned}\xi_{exc} &= \mu \sum_{i=0}^{N} \lambda_i \sqrt{\frac{\pi \xi(k)}{2}}, k \rightarrow \infty \\ &= \mu \sum_{i=0}^{N} \lambda_i \sqrt{\pi \, \frac{\xi_{min} + \xi_{exc}}{2}} \qquad (23)\end{aligned}$$

since $lim_{k\rightarrow\infty} \xi(k) = \xi_{min} + \xi_{exc}$.

Therefore

$$\xi_{exc}^2 = \mu^2 \left(\sum_{i=0}^{N} \lambda_i \right)^2 \left(\frac{\pi \xi_{min}}{2} + \frac{\pi \xi_{exc}}{2} \right) \qquad (24)$$

Quantized-Error Algorithms

The solution for ξ_{exc}, when μ is small, is

$$\begin{aligned}\xi_{exc} &= \mu\sqrt{\frac{\pi\xi_{min}}{2}}\sum_{i=0}^{N}\lambda_i \\ &= \mu\sqrt{\frac{\pi\xi_{min}}{2}}\,tr[\mathbf{R}] \qquad (25)\end{aligned}$$

By comparing with the LMS, for the same excess of MSE

$$\mu = \mu_{LMS}\sqrt{\frac{2}{\pi}\xi_{min}^{-1}} \qquad (26)$$

Quantized-Error Algorithms

The misadjustment in the sign error algorithm is

$$M = \mu \sqrt{\frac{\pi}{2\xi_{min}}}\; tr[\mathbf{R}] \tag{27}$$

Note that when $\xi(k)$ is small $||E[\Delta\mathbf{w}(k+1)]||$ in the equation (11) can increase. In this case, from (8) we can conclude that

$$||\Delta\mathbf{w}(k+1)||^2 - ||\Delta\mathbf{w}(k)||^2 = -4\mu\ \mathrm{sgn}(e(k))\ e(k) + 4\mu^2||\mathbf{x}(k)||^2 \tag{28}$$

where a decrease in the norm of $\Delta\mathbf{w}(k)$ is obtained only while

$$|e(k)| > \mu||\mathbf{x}(k)||^2 \tag{29}$$

Quantized-Error Algorithms

For no additional noise, it follows that

$$\begin{aligned} E[e^2(k+1)] &= E[e^2(k)] - 4\mu E[|e(k)|] \; ||\mathbf{x}(k)||^2 \\ &+ 4\mu^2 E[||\mathbf{x}(k)||^4] \end{aligned} \tag{30}$$

such that

$$E[|e(k)|] = \mu E[||x(k)||^2], k \rightarrow \infty \tag{31}$$

Quantized-Error Algorithms

For zero-mean Gaussian $e(k)$, we conclude that

$$E[|e(k)|] \approx \sqrt{\frac{2}{\pi}}\sigma_e(k), k \rightarrow \infty \tag{32}$$

therefore, the expected variance of $e(k)$ is

$$\sigma_e^2(k) \approx \frac{\pi}{2}\mu^2 \; tr^2[\mathbf{R}], k \rightarrow \infty \tag{33}$$

If $n(k)$ has constantly large absolute value as compared to $-\Delta\mathbf{w}^T(k)\mathbf{x}(k)$ then $\text{sgn}(e(k)) = \text{sgn}(n(k))$ and the sign algorithm is fully controlled by the additional noise. In this case, the algorithm does not converge.

Quantized-Error Algorithms

Transient Behavior

The ratios r_{w_i} are given by

$$r_{w_i} = (1 - 2\mu\sqrt{\frac{2}{\pi\xi(k)}}\lambda_i) \tag{34}$$

for $i = 0, 1, \cdots N$. If μ is chosen in order to reach the same excess of MSE of the LMS algorithm, then

$$r_{w_i} = (1 - \frac{4}{\pi}\mu_{LMS}\sqrt{\frac{\xi_{min}}{\xi(k)}}\ \lambda_i) \tag{35}$$

Quantized-Error Algorithms

Recalling that r_{w_i} for the LMS is $(1 - 2\mu_{LMS}\lambda_i)$ since $\frac{2}{\pi}\sqrt{\frac{\xi_{min}}{\xi(k)}} < 1$, it is concluded that sign error algorithm is slower than the LMS for the same excess of MSE.

Quantized-Error Algorithms

Example 4.1

Suppose in an adaptive filtering environment that the input signal consists of

$$x(k) = e^{j\omega_0 k} + n(k)$$

and that the desired signal is given by

$$d(k) = e^{j\omega_0 (k-1)}$$

where $n(k)$ is a uniformly distributed white noise with variance $\sigma_n^2 = 0.1$ and $\omega_0 = \frac{2\pi}{M}$. In this case $M = 8$.

Compute the input-signal correlation matrix for a first-order adaptive filter. Calculate the value of μ_{max} for the sign-error algorithm.

Quantized-Error Algorithms

Solution:

The input-signal correlation matrix is:

$$\mathbf{R} = \begin{bmatrix} 1+\sigma_n^2 & e^{j\omega_0} \\ e^{-j\omega_0} & 1+\sigma_n^2 \end{bmatrix}$$

Since in this case $tr[\mathbf{R}] = 2.2$ and $\xi_{min} = 0.1$, we have

$$\xi_{exc} \approx \mu\sqrt{\frac{\pi\xi_{min}}{2}}\, tr[\mathbf{R}] = 0.87\mu$$

Quantized-Error Algorithms

The range of values of the convergence factor is given by

$$0 < \mu < \frac{1}{2tr[\mathbf{R}]}\sqrt{\frac{\pi(\xi_{min} + \xi_{exc})}{2}}$$

The upper bound for the convergence factor that is then given by

$$\mu_{max} \approx 0.132$$

Quantized-Error Algorithms

Dual Sign Algorithm

The quantization function for the dual sign algorithm is given by

$$\mathrm{ds}[a] = \begin{cases} \epsilon \ \mathrm{sgn}[a], & |a| > \rho \\ \mathrm{sgn}[a], & |a| \leq \rho \end{cases} \tag{36}$$

where γ is a power of two, greater than 1.

Quantized-Error Algorithms

The coefficient updating is performed as

$$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\mu \ \mathrm{ds}[e(k)]\mathbf{x}(k) \tag{37}$$

The objective function is given by

$$F[e(k)] = \begin{cases} 2\epsilon|e(k)| - 2\rho(\epsilon - 1), & |e(k)| > \rho \\ 2|e(k)|, & |e(k)| \leq \rho \end{cases} \tag{38}$$

Quantized-Error Algorithms

Power-of-Two Error Algorithm

The power-of-two error algorithm uses the quantization defined by

$$\mathrm{pe}[b] = \begin{cases} \mathrm{sgn}(b), & |b| \geq 1 \\ 2^{\mathrm{int}[log_2|b|]}\ \mathrm{sgn}(b), & 2^{-bd+1} \leq |b| < 1 \\ \tau \mathrm{sgn}(b), & |b| < 2^{-bd+1} \end{cases} \tag{39}$$

Quantized-Error Algorithms

bd is the data wordlength excl. the sign bit, and τ is usually 0 or 2^{-bd}.

The coefficient updating is

$$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\mu \ \mathrm{pe}[e(k)]\mathbf{x}(k) \tag{40}$$

Quantized-Error Algorithms

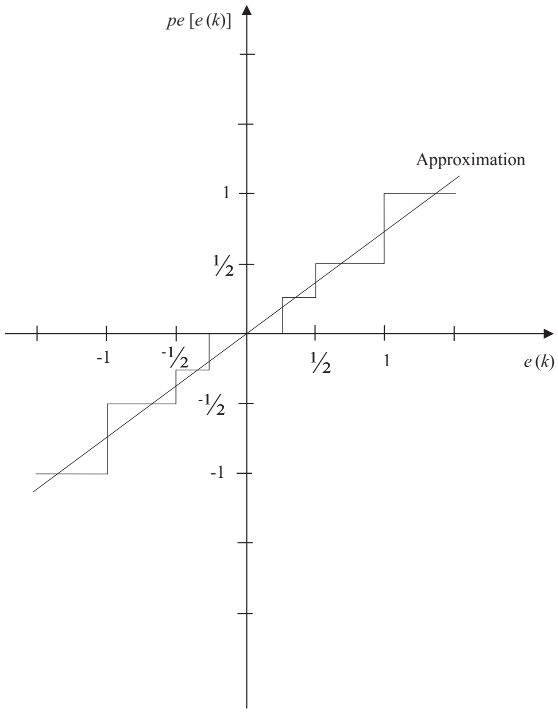
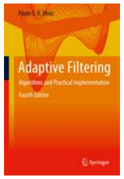


Figure 2: Transfer characteristic of a quantizer with 3 bits and $\tau = 0$.

Quantized-Error Algorithms



Sign Data Algorithm

The sign data algorithm

$$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\mu e(k)\ \mathrm{sgn}[\mathbf{x}(k)] \tag{41}$$

The sign-sign algorithm

$$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\mu\ \mathrm{sgn}[e(k)]\ \mathrm{sgn}[\mathbf{x}(k)] \tag{42}$$

The LMS-Newton Algorithm

For the direct form FIR structure, the MSE can be described by

$$\begin{aligned} \xi(k+1) &= \xi(k) + \mathbf{g_w}^T(k)(\mathbf{w}(k+1) - \mathbf{w}(k)) \\ &+ (\mathbf{w}(k+1) - \mathbf{w}(k))^T \mathbf{R} (\mathbf{w}(k+1) - \mathbf{w}(k)) \end{aligned} \tag{43}$$

The MSE is minimized at the instant $k+1$ if

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \frac{1}{2}\mathbf{R}^{-1}\mathbf{g_w}(k) \tag{44}$$

The LMS-Newton Algorithm

In practice, only estimates are available, so that

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \mu \hat{\mathbf{R}}^{-1}(k)\hat{\mathbf{g}}_{\mathbf{w}}(k) \tag{45}$$

and μ protects from divergence.

A consistent estimate of $\mathbf{R}$ is

$$\begin{aligned} \hat{\mathbf{R}}(k) &= \frac{1}{k+1}\sum_{i=0}^{k}\mathbf{x}(i)\mathbf{x}^T(i) \\ &= \frac{k}{k+1}\hat{\mathbf{R}}(k-1) + \frac{1}{k+1}\mathbf{x}(k)\mathbf{x}^T(k) \end{aligned} \tag{46}$$

The LMS-Newton Algorithm

since

$$E[\hat{\mathbf{R}}(k)] = \frac{1}{k+1}\sum_{i=0}^{k} E[\mathbf{x}(i)\mathbf{x}^T(i)] = \mathbf{R} \tag{47}$$

this is not a practical estimate for $\mathbf{R}$. An appropriate estimate is

$$\begin{aligned} \hat{\mathbf{R}}(k) &= \alpha\mathbf{x}(k)\mathbf{x}^T(k) + (1-\alpha)\hat{\mathbf{R}}(k-1) \\ &= \alpha\mathbf{x}(k)\mathbf{x}^T(k) + \alpha\sum_{i=0}^{k-1}(1-\alpha)^{k-i}\mathbf{x}(i)\mathbf{x}^T(i) \end{aligned} \tag{48}$$

α is small, chosen in the range $0 < \alpha \leq 0.1$.

The LMS-Newton Algorithm

For $k \rightarrow \infty$, it follows that

$$
\begin{aligned}
E[\hat{\mathbf{R}}(k)] &= \alpha \sum_{i=0}^{k-1} (1-\alpha)^{k-i} E[\mathbf{x}(i)\mathbf{x}^T(i)] \\
&= \mathbf{R} \qquad k \rightarrow \infty
\end{aligned} \tag{49}
$$

Therefore, the estimate of $\mathbf{R}$ is unbiased.

The LMS-Newton Algorithm

To avoid inverting $\hat{\mathbf{R}}(k)$, we can use the matrix inversion lemma

$$[\mathbf{A}+\mathbf{BCD}]^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}[\mathbf{DA}^{-1}\mathbf{B}+\mathbf{C}^{-1}]^{-1}\mathbf{DA}^{-1} \tag{50}$$

where $\mathbf{A}$, and $\mathbf{C}$ are square matrices.

The LMS-Newton Algorithm

Choose $\mathbf{A} = (1-\alpha)\,\hat{\mathbf{R}}(k-1)$, $\mathbf{B} = \mathbf{D}^T = \mathbf{x}(k)$ and $\mathbf{C} = \alpha$, it can be shown that

$$\hat{\mathbf{R}}^{-1}(k) = \frac{1}{1-\alpha}\left\{\hat{\mathbf{R}}^{-1}(k-1) - \frac{\hat{\mathbf{R}}^{-1}(k-1)\mathbf{x}(k)\mathbf{x}^T(k)\hat{\mathbf{R}}^{-1}(k-1)}{\frac{1-\alpha}{\alpha} + \mathbf{x}^T(k)\hat{\mathbf{R}}^{-1}(k-1)\mathbf{x}(k)}\right\} \tag{51}$$

$\hat{\mathbf{R}}^{-1}(k)$ is of order N^2 multiplications.

The LMS-Newton Algorithm

If the LMS direction is used, the updating formula is

$$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\,\mu\, e(k)\, \hat{\mathbf{R}}^{-1}(k)\mathbf{x}(k) \tag{52}$$

The LMS-Newton Algorithm

Algorithm 4.2
LMS-Newton Algorithm

Initialization

$\hat{\mathbf{R}}^{-1}(-1) = \delta \mathbf{I}$ (δ a small positive constant)

$\mathbf{w}(0) = \mathbf{x}(-1) = [0 \ldots 0]^T$

Do for $k \geq 0$

$e(k) = d(k) - \mathbf{x}^T(k)\mathbf{w}(k)$

$$\hat{\mathbf{R}}^{-1}(k) = \frac{1}{1-\alpha}\left[\hat{\mathbf{R}}^{-1}(k-1) - \frac{\hat{\mathbf{R}}^{-1}(k-1)\mathbf{x}(k)\mathbf{x}^T(k)\hat{\mathbf{R}}^{-1}(k-1)}{\frac{1-\alpha}{\alpha} + \mathbf{x}^T(k)\hat{\mathbf{R}}^{-1}(k-1)\mathbf{x}(k)}\right]$$

$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\,\mu\, e(k)\, \hat{\mathbf{R}}^{-1}(k)\mathbf{x}(k)$

The Normalized LMS Algorithm

The updating equation of the LMS algorithm can employ a variable convergence factor μ_k in order to improve the convergence rate. In this case

$$\begin{aligned} \mathbf{w}(k+1) &= \mathbf{w}(k) + 2\mu_k e(k)\mathbf{x}(k) \\ &= \mathbf{w}(k) + \Delta\tilde{\mathbf{w}}(k) \end{aligned} \qquad (53)$$

μ_k is chosen with the objective of achieving a faster convergence rate. A strategy is to reduce the instantaneous squared error

The Normalized LMS Algorithm

The instantaneous squared error is given by

$$e^2(k) = d^2(k) + \mathbf{w}^T(k)\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{w}(k) - 2d(k)\mathbf{w}^T(k)\mathbf{x}(k) \tag{54}$$

The Normalized LMS Algorithm

If $\tilde{\mathbf{w}}(k) = \mathbf{w}(k) + \Delta\tilde{\mathbf{w}}(k)$ is performed in the weight vector,

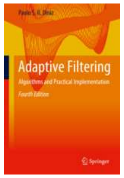
$$
\begin{aligned}
\tilde{e}^2(k) &= e^2(k) + 2\Delta\tilde{\mathbf{w}}^T(k)\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{w}(k) \\
&+ \Delta\tilde{\mathbf{w}}^T(k)\mathbf{x}(k)\mathbf{x}^T(k)\Delta\tilde{\mathbf{w}}(k) - 2d(k)\Delta\tilde{\mathbf{w}}^T(k)\mathbf{x}(k) \quad (55)
\end{aligned}
$$

The Normalized LMS Algorithm

It then follows that

$$\begin{aligned}\Delta e^2(k) &= \tilde{e}^2(k) - e^2(k) \\ &= -2\Delta\tilde{\mathbf{w}}^T(k)\mathbf{x}(k)e(k) + \Delta\tilde{\mathbf{w}}^T(k)\mathbf{x}(k)\mathbf{x}^T(k)\Delta\tilde{\mathbf{w}}(k)\end{aligned} \tag{56}$$

The Normalized LMS Algorithm



By replacing $\Delta\tilde{\mathbf{w}}(k)$ in the equation above

$$\Delta e^2(k) = -4\mu_k e^2(k)\mathbf{x}^T(k)\mathbf{x}(k) + 4\mu_k^2 e^2(k)[\mathbf{x}^T(k)\mathbf{x}(k)]^2 \tag{57}$$

The value of μ_k such that $\frac{\partial \Delta e^2(k)}{\partial \mu_k} = 0$ is given by

$$\mu_k = \frac{1}{2\mathbf{x}^T(k)\mathbf{x}(k)} \tag{58}$$

The Normalized LMS Algorithm

The updating equation for the LMS with var. conv. factor is

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{e(k)\mathbf{x}(k)}{\mathbf{x}^T(k)\mathbf{x}(k)} \tag{59}$$

A μ_n is included in the updating to control the misadjustment. Also a γ should be included to avoid large step sizes when $\mathbf{x}^T(k)\mathbf{x}(k)$ becomes small.

The range of values of μ_n to guarantee stability is

$$0 < \mu = \frac{\mu_n}{2\,tr[\mathbf{R}]} < \frac{1}{tr[\mathbf{R}]} \tag{60}$$

or $0 < \mu_n < 2$.

The Normalized LMS Algorithm

Algorithm 4.3

The Normalized LMS Algorithm

Initialization

$\mathbf{x}(0) = \mathbf{w}(0) = [0 \ldots 0]^T$

choose μ_n in the range $0 < \mu_n \leq 2$

$\gamma =$ small constant

Do for $k \geq 0$

$e(k) = d(k) - \mathbf{x}^T(k)\mathbf{w}(k)$

$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{\mu_n}{\gamma + \mathbf{x}^T(k)\mathbf{x}(k)}\, e(k)\, \mathbf{x}(k)$

Transform-Domain LMS Algorithm

In the transform domain LMS algorithm the input signal vector $\mathbf{x}(k)$ is transformed in a more convenient vector $\mathbf{s}(k)$, by applying an orthonormal (or unitary) i.e.

$$\mathbf{s}(k) = \mathbf{T}\mathbf{x}(k) \tag{61}$$

where $\mathbf{T}\mathbf{T}^T = \mathbf{I}$.

The MSE surface is

$$\xi(k) = \xi_{min} + \Delta\mathbf{w}^T(k)\mathbf{R}\Delta\mathbf{w}(k) \tag{62}$$

Transform-Domain LMS Algorithm

In the transform domain case the MSE surface becomes

$$\begin{aligned}\xi(k) &= \xi_{min} + \Delta\hat{\mathbf{w}}^T(k)E[\mathbf{s}(k)\mathbf{s}^T(k)]\Delta\hat{\mathbf{w}}(k) \\ &= \xi_{min} + \Delta\hat{\mathbf{w}}^T(k)\mathbf{T}\mathbf{R}\mathbf{T}^T\Delta\hat{\mathbf{w}}(k) \qquad (63)\end{aligned}$$

The autocorrelation matrix is

$$\mathbf{R}_s = \mathbf{T}\mathbf{R}\mathbf{T}^T \qquad (64)$$

Transform-Domain LMS Algorithm

To diagonalize $\mathbf{R}$, $\mathbf{T}^T$ corresponds to a matrix whose columns consists of the orthonormal eigenvectors of $\mathbf{R}$. This is the Karhunen-Loeve Transform (KLT).

Normalization of $\mathbf{s}(k)$ and application of the LMS algorithm leads to a solution independent of the input signal power. An equivalent alternative, without this limitation, is to apply in the updating formula

$$\hat{w}_i(k+1) = \hat{w}_i(k) + \frac{2\mu}{\gamma + \sigma_i^2(k)} e(k) s_i(k) \qquad (65)$$

where $\sigma_i^2(k) = \alpha s_i^2(k) + (1-\alpha)\sigma_i^2(k-1)$, α is a small in the range $0 < \alpha \leq 0.1$, and γ is small.

Transform-Domain LMS Algorithm

In matrix form the update is

$$\hat{\mathbf{w}}(k+1) = \hat{\mathbf{w}}(k) + 2\mu e(k)\boldsymbol{\sigma}^{-2}(k)\mathbf{s}(k) \tag{66}$$

where $\boldsymbol{\sigma}^{-2}(k)$ is a diagonal matrix.

Transform-Domain LMS Algorithm

The adaptive filter converges to

$$\hat{\mathbf{w}}_o = {\mathbf{R}_s}^{-1}\mathbf{p}_s \tag{67}$$

where $\mathbf{R}_s = \mathbf{T}\mathbf{R}\mathbf{T}^T$ and $\mathbf{p}_s = \mathbf{T}\mathbf{p}$.

As a consequence,

$$\hat{\mathbf{w}}_o = \left(\mathbf{T}\mathbf{R}\mathbf{T}^T\right)^{-1}\mathbf{T}\mathbf{p} = \mathbf{T}\mathbf{R}^{-1}\mathbf{p} = \mathbf{T}\mathbf{w}_o \tag{68}$$

Transform-Domain LMS Algorithm

The convergence speed of $\hat{\mathbf{w}}(k)$ is determined by the eigenvalue spread of $\boldsymbol{\sigma}^{-2}(k)\mathbf{R}_s$.

A number of real transforms such as DCT, discrete Hartley transform and others are available.

In particular, DCT is given by

$$s_0(k) = \frac{1}{\sqrt{N+1}} \sum_{l=0}^{N} x(k-l) \tag{69}$$

and

$$s_i(k) = \sqrt{\frac{2}{N+1}} \sum_{l=0}^{N} x(k-l) \cos[\pi i \frac{(2l+1)}{2(N+1)}]. \tag{70}$$

Transform-Domain LMS Algorithm

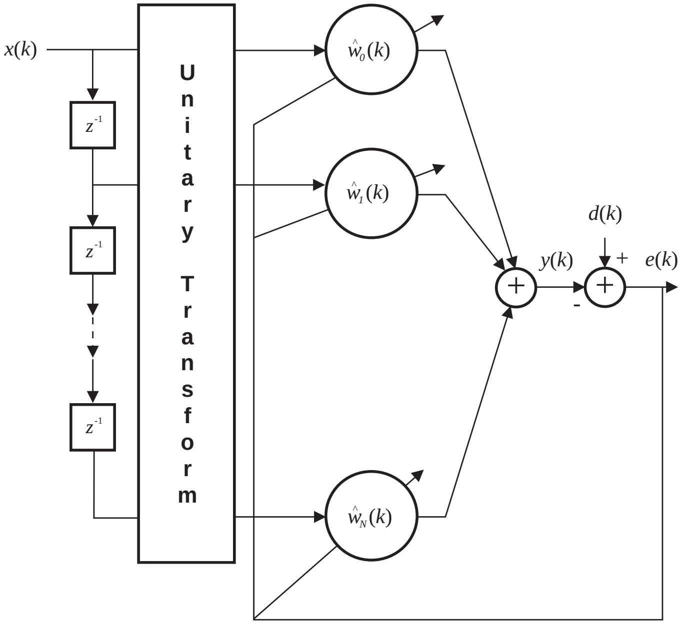
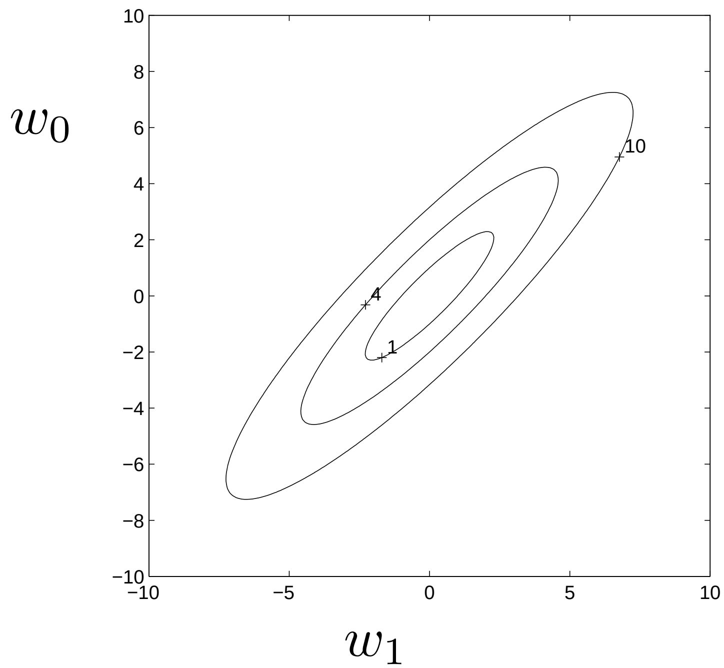
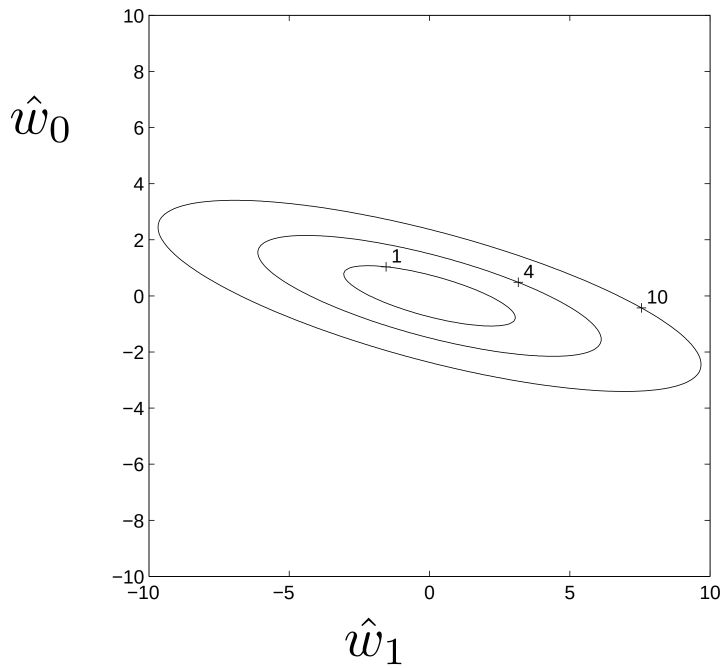


Figure 3: Transform-domain adaptive filter.



$$\mathbf{R} = \begin{bmatrix} 1 & -0.9 \\ -0.9 & 1 \end{bmatrix}$$

Figure 4: Contours of the original MSE surface.



$$\mathbf{T} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \quad \theta = 60^\circ$$

Figure 5: Rotated contours of the MSE surface.

Transform-Domain LMS Algorithm

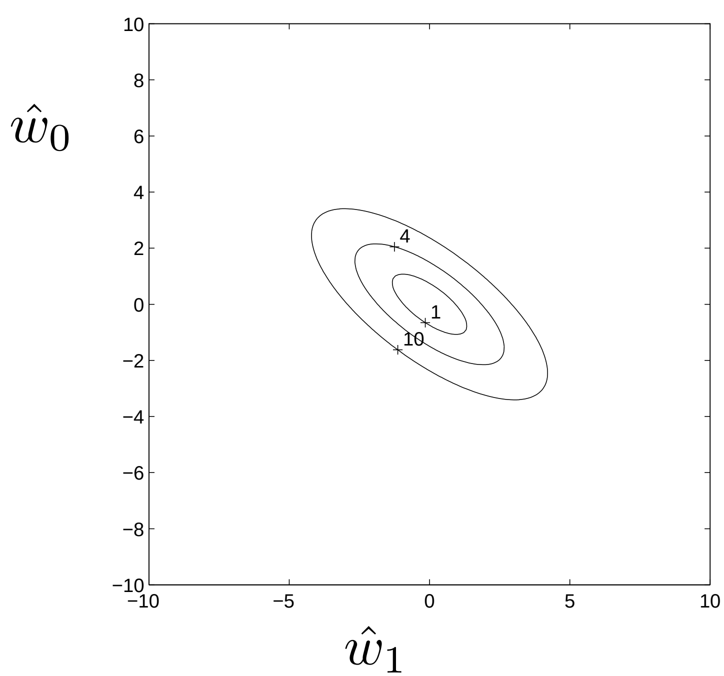
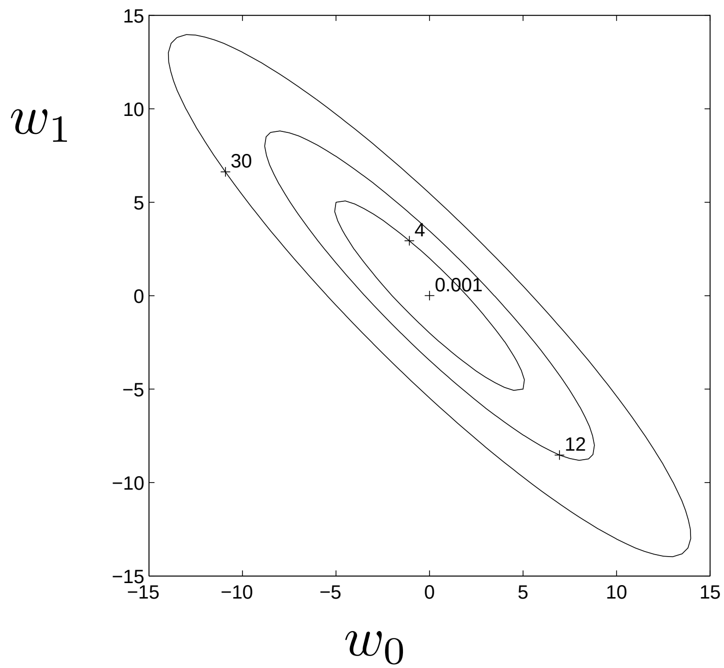
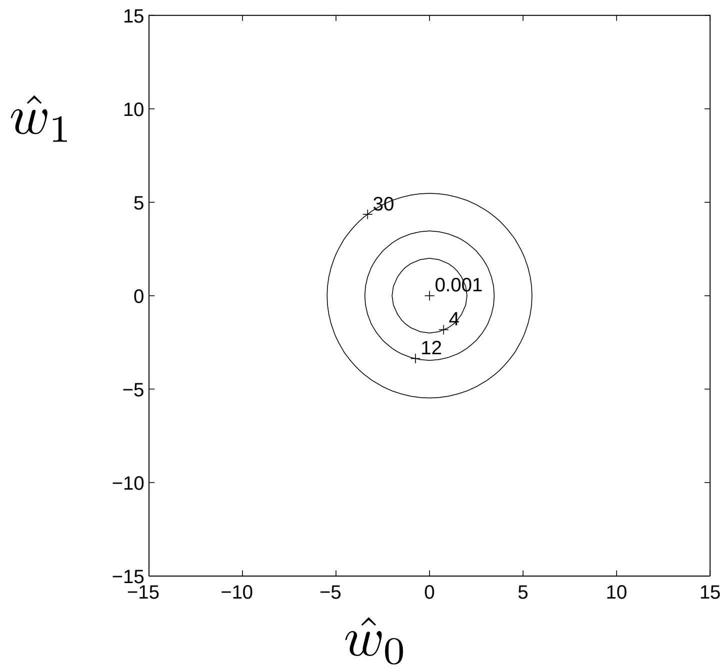


Figure 6: Contours of the power normalized MSE surface.



$$\mathbf{R} = \begin{bmatrix} 1 & 0.92 \\ 0.92 & 1 \end{bmatrix}$$

Figure 7: Contours of the original MSE surface.



$$\mathbf{TR} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Figure 8: Contours of the rotated and power normalized MSE surface.

Transform-Domain LMS Algorithm

Algorithm 4.4
The Transform-Domain LMS Algorithm

Initialization

$\mathbf{x}(0) = \hat{\mathbf{w}}(0) = [0 \ldots 0]^T$

$\gamma =$ small constant

$0 < \alpha \leq 0.1$

Do for each $x(k)$ and $d(k)$ given for $k \geq 0$

$\mathbf{s}(k) = \mathbf{T}\mathbf{x}(k)$

$e(k) = d(k) - \mathbf{s}^T(k)\hat{\mathbf{w}}(k)$

$\hat{\mathbf{w}}(k+1) = \hat{\mathbf{w}}(k) + 2\,\mu\, e(k)\, \boldsymbol{\sigma}^{-2}(k)\mathbf{s}(k)$

Transform-Domain LMS Algorithm

Example 4.2

Repeat the equalization problem of example 3.1 of the previous chapter using the transform-domain algorithm.

(a) Compute the Wiener solution.

(b) Choose an appropriate value for μ and plot the convergence path for the transform-domain algorithm on the MSE surface.

Transform-Domain LMS Algorithm

Solution:

(a) In this example, the correlation matrix of the adaptive filter input signal is given by

$$\mathbf{R} = \begin{bmatrix} 1.6873 & -0.7937 \\ -0.7937 & 1.6873 \end{bmatrix}$$

and the cross-correlation vector $\mathbf{p}$ is

$$\mathbf{p} = \begin{bmatrix} 0.9524 \\ 0.4762 \end{bmatrix}$$

Transform-Domain LMS Algorithm

For square matrix $\mathbf{R}$ of dimension 2, the transformation matrix corresponding to the cosine transform is given by

$$\mathbf{T} = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix}$$

For this filter order, the transformation matrix above coincides with the KLT.

Transform-Domain LMS Algorithm

The coefficients corresponding to the Wiener solution of the transform-domain filter are given by

$$\begin{aligned}
\hat{\mathbf{w}}_o &= (\mathbf{T}\mathbf{R}\mathbf{T}^T)^{-1}\mathbf{T}\mathbf{p} \\
&= \begin{bmatrix} \frac{1}{0.8936} & 0 \\ 0 & \frac{1}{2.4810} \end{bmatrix} \begin{bmatrix} 1.0102 \\ 0.3367 \end{bmatrix} \\
&= \begin{bmatrix} 1.1305 \\ 0.1357 \end{bmatrix}
\end{aligned}$$

Transform-Domain LMS Algorithm

(b) The transform-domain algorithm was applied to minimize the MSE using a small convergence factor $\mu = 1/300$.

The convergence path surface is depicted in Fig. 9.

Transform-Domain LMS Algorithm

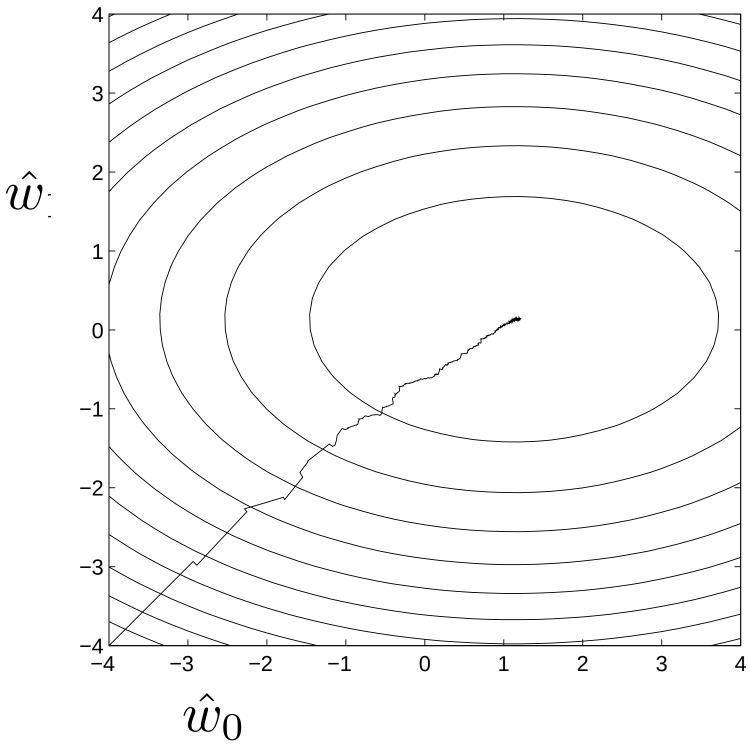


Figure 9: Convergence path of the transform-domain adaptive filter.

The Affine Projection Algorithm

Reuse old data to improve the convergence.

The penalty to be paid by data reusing is increased algorithm misadjustment.

A convergence factor allows a tradeoff.

The Affine Projection Algorithm

The last $L+1$ input signal vectors

$$
\begin{aligned}
&\mathbf{X}_{\mathrm{ap}}(k) \\
=&\begin{bmatrix}
x(k) & x(k-1) & \cdots & x(k-L+1) & x(k-L) \\
x(k-1) & x(k-2) & \cdots & x(k-L) & x(k-L-1) \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
x(k-N) & x(k-N-1) & \cdots & x(k-L-N+1) & x(k-L-N)
\end{bmatrix} \\
=&[\mathbf{x}(k)\ \mathbf{x}(k-1) \ldots \mathbf{x}(k-L)]
\end{aligned}
\tag{71}
$$

Define the partial reusing results at a given iteration k:

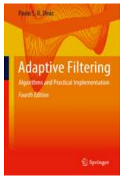
$$\mathbf{y}_{\text{ap}}(k) = \mathbf{X}_{\text{ap}}^{T}(k)\mathbf{w}(k) = \begin{bmatrix} y_{\text{ap},0}(k) \\ y_{\text{ap},1}(k) \\ \vdots \\ y_{\text{ap},L}(k) \end{bmatrix} \tag{72}$$

$$\mathbf{d}_{\text{ap}}(k) = \begin{bmatrix} d(k) \\ d(k-1) \\ \vdots \\ d(k-L) \end{bmatrix} \tag{73}$$

The Affine Projection Algorithm

$$
\mathbf{e}_{\mathrm{ap}}(k) = \begin{bmatrix} e_{\mathrm{ap},0}(k) \\ e_{\mathrm{ap},1}(k) \\ \vdots \\ e_{\mathrm{ap},L}(k) \end{bmatrix} = \begin{bmatrix} d(k) - y_{\mathrm{ap},0}(k) \\ d(k-1) - y_{\mathrm{ap},1}(k) \\ \vdots \\ d(k-L) - y_{\mathrm{ap},L}(k) \end{bmatrix} = \mathbf{d}_{\mathrm{ap}}(k) - \mathbf{y}_{\mathrm{ap}}(k) \tag{74}
$$

The Affine Projection Algorithm



The objective of the affine projection algorithm is to minimize

$$\begin{aligned}
& \frac{1}{2}\|\mathbf{w}(k+1)-\mathbf{w}(k)\|^2 \\
& \text{subject to :} \\
& \mathbf{d}_{\text{ap}}(k)-\mathbf{X}_{\text{ap}}^T(k)\mathbf{w}(k+1)=\mathbf{0}
\end{aligned} \tag{75}$$

The AP algorithm maintains the next coefficient vector $\mathbf{w}(k+1)$ as close as possible to $\mathbf{w}(k)$, while forcing the *a posteriori* error to be zero.

The Affine Projection Algorithm

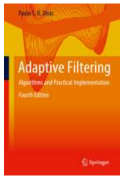
Using the method of Lagrange multipliers:

$$\begin{aligned} F[\mathbf{w}(k+1)] = \tfrac{1}{2}\|\mathbf{w}(k+1) - \mathbf{w}(k)\|^2 \\ +\boldsymbol{\lambda}_{\mathrm{ap}}^T(k)[\mathbf{d}(k) - \mathbf{X}_{\mathrm{ap}}^T(k)\mathbf{w}(k+1)] \end{aligned} \tag{76}$$

where $\boldsymbol{\lambda}_{\mathrm{ap}}(k)$ is an $(L+1) \times 1$ vector of Lagrange multipliers. Then

$$\begin{aligned} F[\mathbf{w}(k+1)] &= \frac{1}{2}\left[\mathbf{w}(k+1) - \mathbf{w}(k)\right]^T\left[\mathbf{w}(k+1) - \mathbf{w}(k)\right] \\ &\quad + \left[\mathbf{d}^T(k) - \mathbf{w}^T(k+1)\mathbf{X}_{\mathrm{ap}}(k)\right]\boldsymbol{\lambda}_{\mathrm{ap}}(k) \end{aligned} \tag{77}$$

The Affine Projection Algorithm



The gradient of $F[\mathbf{w}(k+1)]$ with respect to $\mathbf{w}(k+1)$ is given by

$$\mathbf{g_w}\left\{F[\mathbf{w}(k+1)]\right\} = \frac{1}{2}\left[2\mathbf{w}(k+1) - 2\mathbf{w}(k)\right] - \mathbf{X}_{\rm ap}(k)\boldsymbol{\lambda}_{\rm ap}(k) \qquad (78)$$

After setting $\mathbf{g_w}\left\{F[\mathbf{w}(k+1)]\right\} = \mathbf{0}$

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mathbf{X}_{\rm ap}(k)\boldsymbol{\lambda}_{\rm ap}(k) \qquad (79)$$

The Affine Projection Algorithm

If we substitute equation (79) in the constraint relation of equation (75)

$$\mathbf{X}_{\mathrm{ap}}^{T}(k)\mathbf{X}_{\mathrm{ap}}(k)\boldsymbol{\lambda}_{\mathrm{ap}}(k) = \mathbf{d}_{\mathrm{ap}}(k) - \mathbf{X}_{\mathrm{ap}}^{T}(k)\mathbf{w}(k) = \mathbf{e}_{\mathrm{ap}}(k) \tag{80}$$

The update equation is given by equation (79) with $\boldsymbol{\lambda}_{\mathrm{ap}}(k)$ being the solution of (80), i.e.,

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mathbf{X}_{\mathrm{ap}}(k)\left(\mathbf{X}_{\mathrm{ap}}^{T}(k)\mathbf{X}_{\mathrm{ap}}(k)\right)^{-1}\mathbf{e}_{\mathrm{ap}}(k) \tag{81}$$

The Affine Projection Algorithm

A trade-off between final misadjustment and convergence speed is achieved through a convergence factor

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu \mathbf{X}_{\mathrm{ap}}(k) \left(\mathbf{X}_{\mathrm{ap}}^T(k) \mathbf{X}_{\mathrm{ap}}(k) \right)^{-1} \mathbf{e}_{\mathrm{ap}}(k) \tag{82}$$

The Affine Projection Algorithm

Algorithm 6.5
The Affine Projection Algorithm

Initialization

$\mathbf{x}(0) = \mathbf{w}(0) = [0\,0 \ldots 0]^T$

choose μ in the range $0 < \mu \leq 2$

γ = small constant

Do for $k \geq 0$

$\mathbf{e}_{\mathrm{ap}}(k) = \mathbf{d}_{\mathrm{ap}}(k) - \mathbf{X}_{\mathrm{ap}}^T(k)\mathbf{w}(k)$

$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu \mathbf{X}_{\mathrm{ap}}(k) \left(\mathbf{X}_{\mathrm{ap}}^T(k)\mathbf{X}_{\mathrm{ap}}(k) + \gamma \mathbf{I} \right)^{-1} \mathbf{e}_{\mathrm{ap}}(k)$

The Affine Projection Algorithm

Let's define the hyperplane $\mathcal{S}(k)$ as follows

$$\mathcal{S}(k) = \{\mathbf{w}(k+1) \in \mathcal{R}^{N+1} : d(k) - \mathbf{w}^T(k+1)\mathbf{x}(k) = 0\} \quad (83)$$

The *a posteriori* error over this hyperplane is zero.

The LMS takes a step towards $\mathcal{S}(k)$ so that $\mathbf{w}(k+1)$ is anywhere between points 1 and 3 in Fig. 10, closer to $\mathcal{S}(k)$ than $\tilde{\mathbf{w}}$. The NLMS with unit convergence factor performs a line search in the direction of $\mathbf{x}(k)$ to yield in a single step the solution $\mathbf{w}(k+1)$, point 3 in Fig. 10, which belongs to $\mathcal{S}(k)$. A single reuse of the previous data using NLMS would lead to point 4. The binormalized LMS solution belongs to $\mathcal{S}(k-1)$ and $\mathcal{S}(k)$, repres. by point 5 in Fig. 10.

The Affine Projection Algorithm

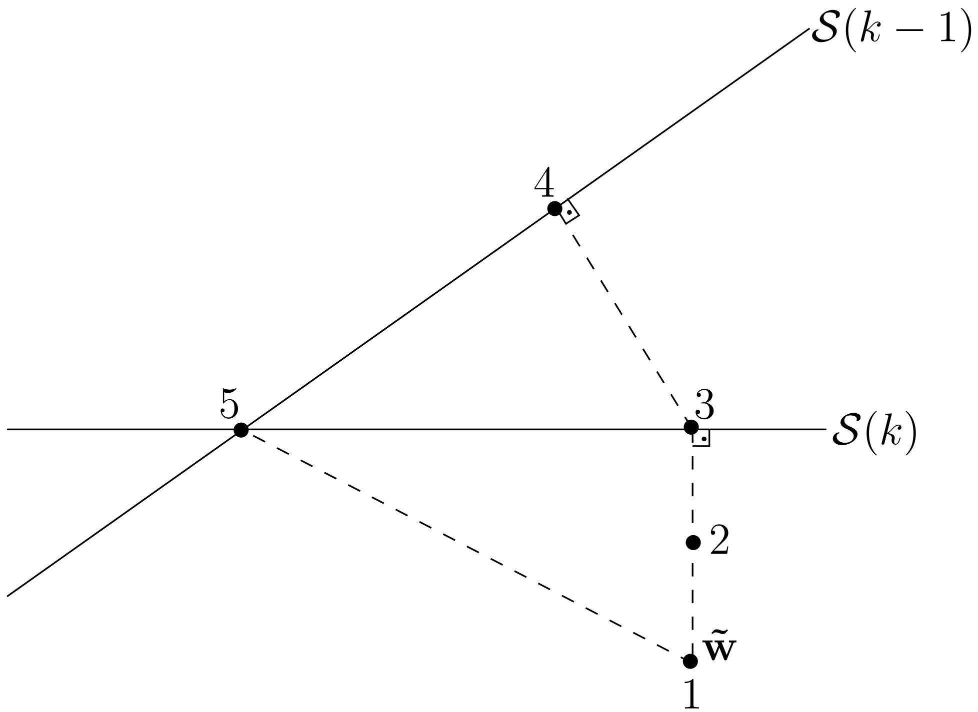


Figure 10: Coefficient vector updating for the normalized LMS algorithm and binormalized LMS algorithm.

The Affine Projection Algorithm

It is possible to observe in Fig. 11 that by repeatedly re-utilizing the data vectors $\mathbf{x}(k)$ and $\mathbf{x}(k-1)$ to update the coefficients with the normalized LMS algorithm would reach point 5 in a zig-zag pattern after an infinite number of iterations. This approach is known as Kaczmarz method.

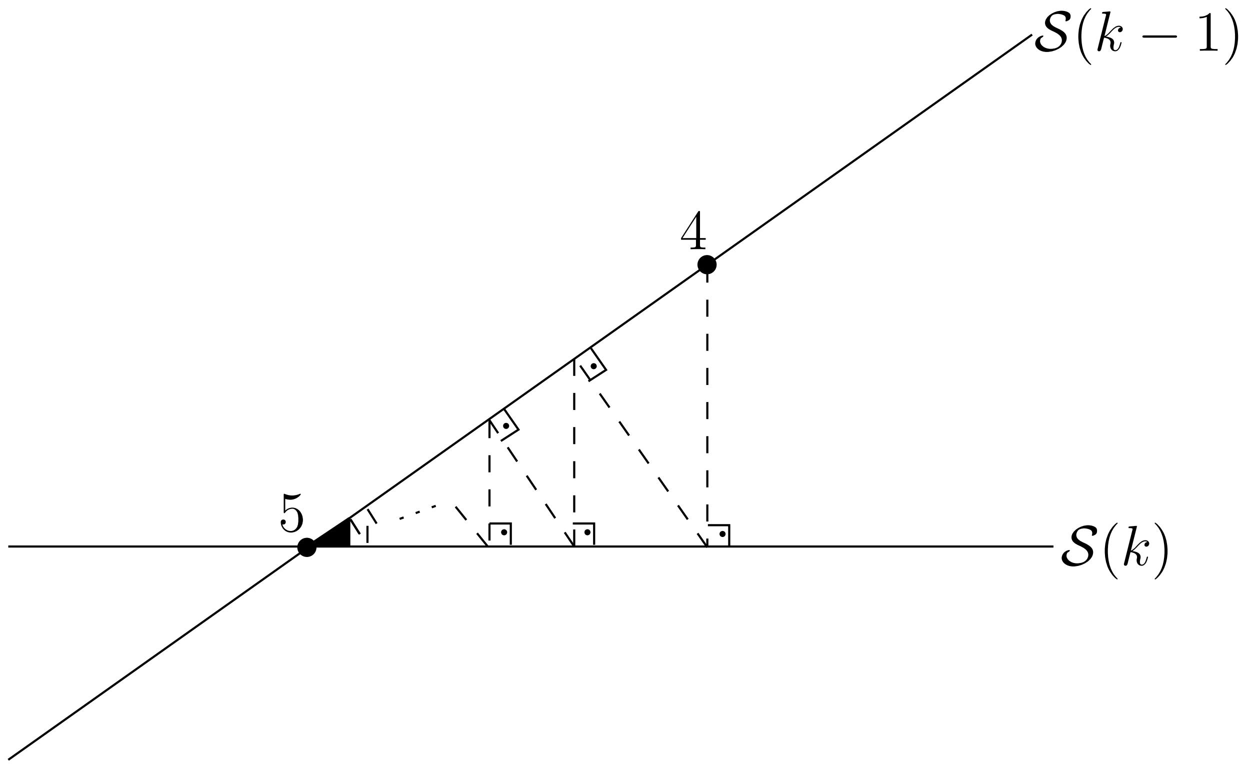


Figure 11: Multiple data reuse for the normalized LMS algorithm.

The Affine Projection Algorithm

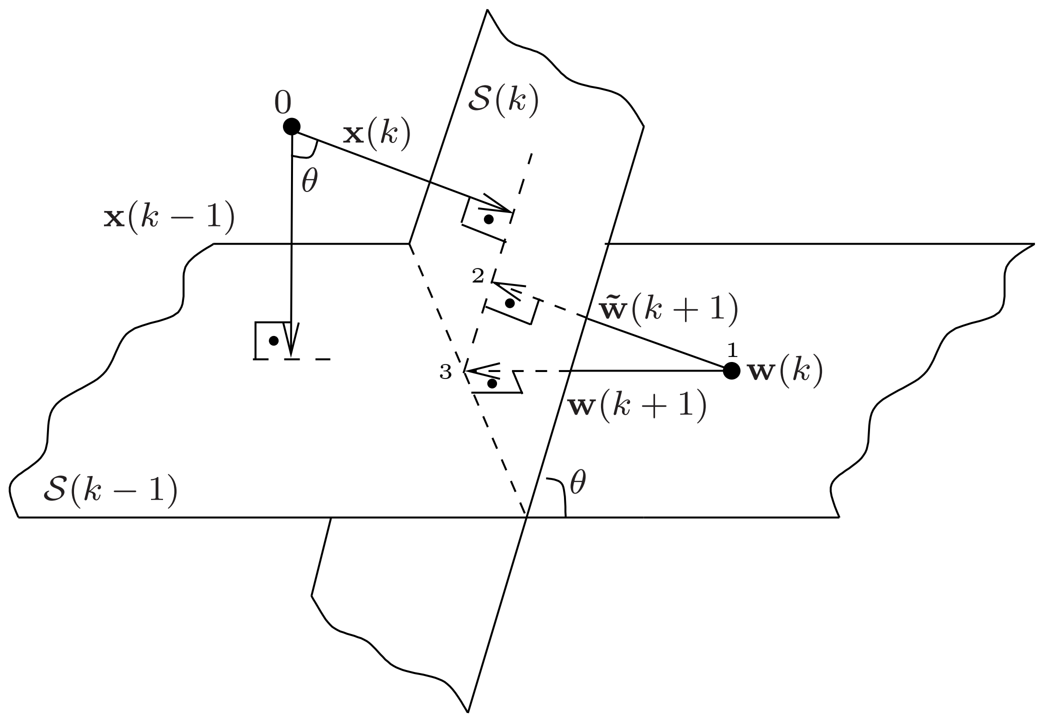
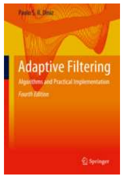


Figure 12: Three-dimensional coefficient vector updating for the normalized LMS algorithm and binormalized LMS algorithm.

Misadjustment in the AP Algorithm



A general AF algorithm has coefficient updating

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \mu \mathbf{F}_{\mathbf{X}}(k)\mathbf{f}_{\mathbf{e}}(k) \tag{84}$$

where $\mathbf{F}_{\mathbf{X}}(k)$ is a matrix whose elements are functions of the input data and $\mathbf{f}_{\mathbf{e}}(k)$ is a vector whose elements are functions of the error. Assuming that the desired signal is given by

$$d(k) = \mathbf{w}_o^T \mathbf{x}(k) + n(k) \tag{85}$$

the underlying updating equation can be alternatively described by

$$\Delta\mathbf{w}(k+1) = \Delta\mathbf{w}(k) - \mu \mathbf{F}_{\mathbf{X}}(k)\mathbf{f}_{\mathbf{e}}(k) \tag{86}$$

where $\Delta\mathbf{w}(k) = \mathbf{w}(k) - \mathbf{w}_o$.

Misadjustment in the AP Algorithm

In the case of the affine projection algorithm

$$\mathbf{f}_{\mathbf{e}}(k) = -\mathbf{e}_{\mathrm{ap}}(k) \tag{87}$$

By premultiplying equation (86) by the input vector matrix

$$\begin{aligned} \mathbf{X}_{\mathrm{ap}}^{T}(k)\Delta\mathbf{w}(k+1) &= \mathbf{X}_{\mathrm{ap}}^{T}(k)\Delta\mathbf{w}(k) + \mu\mathbf{X}_{\mathrm{ap}}^{T}(k)\mathbf{F}_{\mathbf{X}}(k)\mathbf{e}_{\mathrm{ap}}(k) \\ -\tilde{\boldsymbol{\varepsilon}}_{\mathrm{ap}}(k) &= -\tilde{\mathbf{e}}_{\mathrm{ap}}(k) + \mu\mathbf{X}_{\mathrm{ap}}^{T}(k)\mathbf{F}_{\mathbf{X}}(k)\mathbf{e}_{\mathrm{ap}}(k) \end{aligned} \tag{88}$$

Misadjustment in the AP Algorithm

$$\tilde{\boldsymbol{\varepsilon}}_{\mathrm{ap}}(k) = -\mathbf{X}_{\mathrm{ap}}^{T}(k)\Delta\mathbf{w}(k+1) \tag{89}$$

is the noiseless *a posteriori* error vector and

$$\tilde{\mathbf{e}}_{\mathrm{ap}}(k) = -\mathbf{X}_{\mathrm{ap}}^{T}(k)\Delta\mathbf{w}(k) = \mathbf{e}_{\mathrm{ap}}(k) - \mathbf{n}_{\mathrm{ap}}(k) \tag{90}$$

is the noiseless *a priori* error vector with

$$\mathbf{n}_{\mathrm{ap}}(k) = \begin{bmatrix} n(k) \\ n(k-1) \\ \vdots \\ n(k-L) \end{bmatrix}$$

Misadjustment in the AP Algorithm

For the regularized affine projection algorithm

$$\mathbf{F}_{\mathbf{X}}(k) = \mathbf{X}_{\rm ap}(k) \left(\mathbf{X}_{\rm ap}^{T}(k) \mathbf{X}_{\rm ap}(k) + \gamma \mathbf{I} \right)^{-1}$$

where the matrix $\gamma\mathbf{I}$ is added in order to avoid numerical problems.

By solving equation (88), we get

$$\frac{1}{\mu} \left(\mathbf{X}_{\rm ap}^{T}(k) \mathbf{X}_{\rm ap}(k) \right)^{-1} \left(\tilde{\mathbf{e}}_{\rm ap}(k) - \tilde{\boldsymbol{\varepsilon}}_{\rm ap}(k) \right) =$$

$$\left(\mathbf{X}_{\rm ap}^{T}(k) \mathbf{X}_{\rm ap}(k) + \gamma \mathbf{I} \right)^{-1} \mathbf{e}_{\rm ap}(k)$$

Misadjustment in the AP Algorithm

If we replace the equation above in

$$\Delta \mathbf{w}(k+1) = \Delta \mathbf{w}(k) + \mu \mathbf{X}_{\mathrm{ap}}(k) \left(\mathbf{X}_{\mathrm{ap}}^{T}(k) \mathbf{X}_{\mathrm{ap}}(k) + \gamma \mathbf{I} \right)^{-1} \mathbf{e}_{\mathrm{ap}}(k) \quad (91)$$

It is possible to deduce that

$$\Delta \mathbf{w}(k+1) - \mathbf{X}_{\mathrm{ap}}(k) \left(\mathbf{X}_{\mathrm{ap}}^{T}(k) \mathbf{X}_{\mathrm{ap}}(k) \right)^{-1} \tilde{\mathbf{e}}_{\mathrm{ap}}(k) =$$

$$\Delta \mathbf{w}(k) - \mathbf{X}_{\mathrm{ap}}(k) \left(\mathbf{X}_{\mathrm{ap}}^{T}(k) \mathbf{X}_{\mathrm{ap}}(k) \right)^{-1} \tilde{\boldsymbol{\varepsilon}}_{\mathrm{ap}}(k) \quad (92)$$

Misadjustment in the AP Algorithm

From the equation above it is possible to prove that

$$E\left[\|\Delta\mathbf{w}(k+1)\|^2\right] + E\left[\tilde{\mathbf{e}}_{\text{ap}}^T(k)\left(\mathbf{X}_{\text{ap}}^T(k)\mathbf{X}_{\text{ap}}(k)\right)^{-1}\tilde{\mathbf{e}}_{\text{ap}}(k)\right] =$$

$$E\left[\|\Delta\mathbf{w}(k)\|^2\right] + E\left[\tilde{\boldsymbol{\varepsilon}}_{\text{ap}}^T(k)\left(\mathbf{X}_{\text{ap}}^T(k)\mathbf{X}_{\text{ap}}(k)\right)^{-1}\tilde{\boldsymbol{\varepsilon}}_{\text{ap}}(k)\right] \qquad (93)$$

Proof: One can calculate the Euclidean norm of both sides of eq. (92)

$$
\begin{aligned}
& \left[\Delta \mathbf{w}(k+1) - \mathbf{X}_{\mathrm{ap}}(k)\left(\mathbf{X}_{\mathrm{ap}}^{T}(k)\mathbf{X}_{\mathrm{ap}}(k)\right)^{-1}\tilde{\mathbf{e}}_{\mathrm{ap}}(k)\right]^{T} \\
\times \quad & \left[\Delta \mathbf{w}(k+1) - \mathbf{X}_{\mathrm{ap}}(k)\left(\mathbf{X}_{\mathrm{ap}}^{T}(k)\mathbf{X}_{\mathrm{ap}}(k)\right)^{-1}\tilde{\mathbf{e}}_{\mathrm{ap}}(k)\right] = \\
& \left[\Delta \mathbf{w}(k) - \mathbf{X}_{\mathrm{ap}}(k)\left(\mathbf{X}_{\mathrm{ap}}^{T}(k)\mathbf{X}_{\mathrm{ap}}(k)\right)^{-1}\tilde{\boldsymbol{\varepsilon}}_{\mathrm{ap}}(k)\right]^{T} \\
\times \quad & \left[\Delta \mathbf{w}(k) - \mathbf{X}_{\mathrm{ap}}(k)\left(\mathbf{X}_{\mathrm{ap}}^{T}(k)\mathbf{X}_{\mathrm{ap}}(k)\right)^{-1}\tilde{\boldsymbol{\varepsilon}}_{\mathrm{ap}}(k)\right]
\end{aligned}
$$

By performing the inner products one by one, the equation above becomes

$$
\begin{aligned}
&\Delta\mathbf{w}^T(k+1)\Delta\mathbf{w}(k+1) - \Delta\mathbf{w}^T(k+1)\mathbf{X}_{\mathrm{ap}}(k)\left(\mathbf{X}_{\mathrm{ap}}^T(k)\mathbf{X}_{\mathrm{ap}}(k)\right)^{-1}\tilde{\mathbf{e}}_{\mathrm{ap}}(k) \\
&- \left[\mathbf{X}_{\mathrm{ap}}(k)\left(\mathbf{X}_{\mathrm{ap}}^T(k)\mathbf{X}_{\mathrm{ap}}(k)\right)^{-1}\tilde{\mathbf{e}}_{\mathrm{ap}}(k)\right]^T\Delta\mathbf{w}(k+1) \\
&+ \left[\mathbf{X}_{\mathrm{ap}}(k)\left(\mathbf{X}_{\mathrm{ap}}^T(k)\mathbf{X}_{\mathrm{ap}}(k)\right)^{-1}\tilde{\mathbf{e}}_{\mathrm{ap}}(k)\right]^T\left[\mathbf{X}_{\mathrm{ap}}(k)\left(\mathbf{X}_{\mathrm{ap}}^T(k)\mathbf{X}_{\mathrm{ap}}(k)\right)^{-1}\tilde{\mathbf{e}}_{\mathrm{ap}}(k)\right] = \\
&\Delta\mathbf{w}^T(k)\Delta\mathbf{w}(k) - \Delta\mathbf{w}^T(k)\mathbf{X}_{\mathrm{ap}}(k)\left(\mathbf{X}_{\mathrm{ap}}^T(k)\mathbf{X}_{\mathrm{ap}}(k)\right)^{-1}\tilde{\boldsymbol{\varepsilon}}_{\mathrm{ap}}(k) \\
&- \left[\mathbf{X}_{\mathrm{ap}}(k)\left(\mathbf{X}_{\mathrm{ap}}^T(k)\mathbf{X}_{\mathrm{ap}}(k)\right)^{-1}\tilde{\boldsymbol{\varepsilon}}_{\mathrm{ap}}(k)\right]^T\Delta\mathbf{w}(k) \\
&+ \left[\mathbf{X}_{\mathrm{ap}}(k)\left(\mathbf{X}_{\mathrm{ap}}^T(k)\mathbf{X}_{\mathrm{ap}}(k)\right)^{-1}\tilde{\boldsymbol{\varepsilon}}_{\mathrm{ap}}(k)\right]^T\left[\mathbf{X}_{\mathrm{ap}}(k)\left(\mathbf{X}_{\mathrm{ap}}^T(k)\mathbf{X}_{\mathrm{ap}}(k)\right)^{-1}\tilde{\boldsymbol{\varepsilon}}_{\mathrm{ap}}(k)\right]
\end{aligned}
$$

Misadjustment in the AP Algorithm

Since $\tilde{\boldsymbol{\varepsilon}}_{\text{ap}}(k) = -\mathbf{X}_{\text{ap}}^T(k)\Delta\mathbf{w}(k+1)$ and $\tilde{\mathbf{e}}_{\text{ap}}(k) = -\mathbf{X}_{\text{ap}}^T(k)\Delta\mathbf{w}(k)$

$$
\begin{aligned}
& \|\Delta\mathbf{w}(k+1)\|^2 + \tilde{\boldsymbol{\varepsilon}}_{\text{ap}}^T(k)\left(\mathbf{X}_{\text{ap}}^T(k)\mathbf{X}_{\text{ap}}(k)\right)^{-1}\tilde{\mathbf{e}}_{\text{ap}}(k) \\
+ \quad & \tilde{\mathbf{e}}_{\text{ap}}^T(k)\left(\mathbf{X}_{\text{ap}}^T(k)\mathbf{X}_{\text{ap}}(k)\right)^{-1}\tilde{\boldsymbol{\varepsilon}}_{\text{ap}}(k) + \tilde{\mathbf{e}}_{\text{ap}}^T(k)\left(\mathbf{X}_{\text{ap}}^T(k)\mathbf{X}_{\text{ap}}(k)\right)^{-1}\tilde{\mathbf{e}}_{\text{ap}}(k) = \\
& \|\Delta\mathbf{w}(k)\|^2 + \tilde{\mathbf{e}}_{\text{ap}}^T(k)\left(\mathbf{X}_{\text{ap}}^T(k)\mathbf{X}_{\text{ap}}(k)\right)^{-1}\tilde{\boldsymbol{\varepsilon}}_{\text{ap}}(k) \\
+ \quad & \tilde{\boldsymbol{\varepsilon}}_{\text{ap}}^T(k)\left(\mathbf{X}_{\text{ap}}^T(k)\mathbf{X}_{\text{ap}}(k)\right)^{-1}\tilde{\mathbf{e}}_{\text{ap}}(k) + \tilde{\boldsymbol{\varepsilon}}_{\text{ap}}^T(k)\left(\mathbf{X}_{\text{ap}}^T(k)\mathbf{X}_{\text{ap}}(k)\right)^{-1}\tilde{\boldsymbol{\varepsilon}}_{\text{ap}}(k)
\end{aligned}
$$

Misadjustment in the AP Algorithm

By removing the equal terms on both sides of the last equation the following equality holds

$$\|\Delta\mathbf{w}(k+1)\|^2 + \tilde{\mathbf{e}}_{\rm ap}^T(k)\left(\mathbf{X}_{\rm ap}^T(k)\mathbf{X}_{\rm ap}(k)\right)^{-1}\tilde{\mathbf{e}}_{\rm ap}(k) =$$

$$\|\Delta\mathbf{w}(k)\|^2 + \tilde{\boldsymbol{\varepsilon}}_{\rm ap}^T(k)\left(\mathbf{X}_{\rm ap}^T(k)\mathbf{X}_{\rm ap}(k)\right)^{-1}\tilde{\boldsymbol{\varepsilon}}_{\rm ap}(k) \qquad (94)$$

As can be observed no approximations were utilized so far. Now by applying the expected value operation on both sides of the equation above, the expression of equation

Misadjustment in the AP Algorithm

If it is assumed that the algorithm has converged, then $E\left[\|\Delta\mathbf{w}(k+1)\|^2\right] = E\left[\|\Delta\mathbf{w}(k)\|^2\right]$. As a result the following equality holds in the steady state.

$$
\begin{aligned}
E\left[\tilde{\mathbf{e}}_{\text{ap}}^T(k)\left(\mathbf{X}_{\text{ap}}^T(k)\mathbf{X}_{\text{ap}}(k)\right)^{-1}\tilde{\mathbf{e}}_{\text{ap}}(k)\right] &= \\
E\left[\tilde{\boldsymbol{\varepsilon}}_{\text{ap}}^T(k)\left(\mathbf{X}_{\text{ap}}^T(k)\mathbf{X}_{\text{ap}}(k)\right)^{-1}\tilde{\boldsymbol{\varepsilon}}_{\text{ap}}(k)\right] & \qquad (95)
\end{aligned}
$$

This is useful to remove the *a posteriori* error, by applying equation (88) to the AP algorithm case.

$$
\tilde{\boldsymbol{\varepsilon}}_{\text{ap}}(k) = \tilde{\mathbf{e}}_{\text{ap}}(k) - \mu\mathbf{X}_{\text{ap}}^T(k)\mathbf{X}_{\text{ap}}(k)\left(\mathbf{X}_{\text{ap}}^T(k)\mathbf{X}_{\text{ap}}(k) + \gamma\mathbf{I}\right)^{-1}\mathbf{e}_{\text{ap}}(k) \quad (96)
$$

By substituting equation (96) in equation (95) we get

$$
\begin{aligned}
& E\left[\tilde{\mathbf{e}}_{\mathrm{ap}}^{T}(k)\left(\mathbf{X}_{\mathrm{ap}}^{T}(k)\mathbf{X}_{\mathrm{ap}}(k)\right)^{-1}\tilde{\mathbf{e}}_{\mathrm{ap}}(k)\right] = \\
& E\left[\tilde{\mathbf{e}}_{\mathrm{ap}}^{T}(k)\left(\mathbf{X}_{\mathrm{ap}}^{T}(k)\mathbf{X}_{\mathrm{ap}}(k)\right)^{-1}\tilde{\mathbf{e}}_{\mathrm{ap}}(k)\right. \\
- \quad & \mu\tilde{\mathbf{e}}_{\mathrm{ap}}^{T}(k)\left(\mathbf{X}_{\mathrm{ap}}^{T}(k)\mathbf{X}_{\mathrm{ap}}(k)+\gamma\mathbf{I}\right)^{-1}\mathbf{e}_{\mathrm{ap}}(k) \\
- \quad & \mu\mathbf{e}_{\mathrm{ap}}^{T}(k)\left(\mathbf{X}_{\mathrm{ap}}^{T}(k)\mathbf{X}_{\mathrm{ap}}(k)+\gamma\mathbf{I}\right)^{-1}\tilde{\mathbf{e}}_{\mathrm{ap}}(k) \\
+ \quad & \mu^{2}\mathbf{e}_{\mathrm{ap}}^{T}(k)\left(\mathbf{X}_{\mathrm{ap}}^{T}(k)\mathbf{X}_{\mathrm{ap}}(k)+\gamma\mathbf{I}\right)^{-1}\mathbf{X}_{\mathrm{ap}}^{T}(k)\mathbf{X}_{\mathrm{ap}}(k) \\
& \left.\left(\mathbf{X}_{\mathrm{ap}}^{T}(k)\mathbf{X}_{\mathrm{ap}}(k)+\gamma\mathbf{I}\right)^{-1}\mathbf{e}_{\mathrm{ap}}(k)\right]
\end{aligned}
\tag{97}
$$

Misadjustment in the AP Algorithm

The expression above can be simplified as

$$
\begin{aligned}
& \mu^2 E\Big[\mathbf{e}_{\mathrm{ap}}^T(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\hat{\mathbf{R}}_{\mathrm{ap}}(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\mathbf{e}_{\mathrm{ap}}(k)\Big] \\
= \; & \mu E\Big[\tilde{\mathbf{e}}_{\mathrm{ap}}^T(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\mathbf{e}_{\mathrm{ap}}(k) + \mathbf{e}_{\mathrm{ap}}^T(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\tilde{\mathbf{e}}_{\mathrm{ap}}(k)\Big]
\end{aligned}
\tag{98}
$$

where the following definitions are employed to simplify the discussion

$$
\begin{aligned}
\hat{\mathbf{R}}_{\mathrm{ap}}(k) &= \mathbf{X}_{\mathrm{ap}}^T(k)\mathbf{X}_{\mathrm{ap}}(k) \\
\hat{\mathbf{S}}_{\mathrm{ap}}(k) &= \Big(\mathbf{X}_{\mathrm{ap}}^T(k)\mathbf{X}_{\mathrm{ap}}(k) + \gamma\mathbf{I}\Big)^{-1}
\end{aligned}
\tag{99}
$$

Misadjustment in the AP Algorithm

Using the error squared we obtain

$$\begin{aligned}\xi(k) \;&=\; E[e^2(k)] = E[n^2(k)] - 2E[n(k)\Delta\mathbf{w}^T(k)\mathbf{x}(k)] \\ &\quad + E[\Delta\mathbf{w}^T(k)\mathbf{x}(k)\mathbf{x}^T(k)\Delta\mathbf{w}(k)]\end{aligned} \tag{100}$$

If the coefficients have weak dependency of the additional noise and applying the orthogonality principle,

$$\begin{aligned}\xi(k) &= \sigma_n^2 + E[\Delta\mathbf{w}^T(k)\mathbf{x}(k)\mathbf{x}^T(k)\Delta\mathbf{w}(k)] \\ &= \sigma_n^2 + E[\tilde{e}^2_{\mathrm{ap},1}(k)]\end{aligned} \tag{101}$$

where $\tilde{e}_{\mathrm{ap},1}(k)$ is the first element of vector $\tilde{\mathbf{e}}_{\mathrm{ap}}(k)$.

Misadjustment in the AP Algorithm

To compute the excess of mean-square error we can remove the value of $E[\tilde{e}^2_{\text{ap},1}(k)]$ from equation (98). Since our aim is to compute $E[\tilde{e}^2_{\text{ap},1}(k)]$, we can substitute equation (90) in equation (98) in order to get rid of $\mathbf{e}_{\text{ap}}(k)$. The resulting expression is given by

$$E\left[\mu(\tilde{\mathbf{e}}_{\text{ap}}(k)+\mathbf{n}_{\text{ap}}(k))^T\hat{\mathbf{S}}_{\text{ap}}(k)\hat{\mathbf{R}}_{\text{ap}}(k)\hat{\mathbf{S}}_{\text{ap}}(k)(\tilde{\mathbf{e}}_{\text{ap}}(k)+\mathbf{n}_{\text{ap}}(k))\right]=$$
$$E\left[\tilde{\mathbf{e}}^T_{\text{ap}}(k)\hat{\mathbf{S}}_{\text{ap}}(k)(\tilde{\mathbf{e}}_{\text{ap}}(k)+\mathbf{n}_{\text{ap}}(k))+(\tilde{\mathbf{e}}_{\text{ap}}(k)+\mathbf{n}_{\text{ap}}(k))^T\hat{\mathbf{S}}_{\text{ap}}(k)\tilde{\mathbf{e}}_{\text{ap}}(k)\right] \tag{102}$$

Misadjustment in the AP Algorithm

By considering the noise white and statistically independent of the input signal, then

$$\mu E\left[\tilde{\mathbf{e}}_{\mathrm{ap}}^{T}(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\hat{\mathbf{R}}_{\mathrm{ap}}(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\tilde{\mathbf{e}}_{\mathrm{ap}}(k)\right.$$
$$\left.+\mathbf{n}_{\mathrm{ap}}^{T}(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\hat{\mathbf{R}}_{\mathrm{ap}}(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\mathbf{n}_{\mathrm{ap}}(k)\right]=$$
$$2E\left[\tilde{\mathbf{e}}_{\mathrm{ap}}^{T}(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\tilde{\mathbf{e}}_{\mathrm{ap}}(k)\right] \tag{103}$$

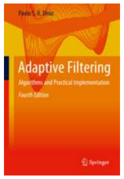
Misadjustment in the AP Algorithm

The expression above, after some rearrangements, can be rewritten as

$$
\begin{aligned}
&2E\left\{\mathrm{tr}[\tilde{\mathbf{e}}_{\mathrm{ap}}(k)\tilde{\mathbf{e}}_{\mathrm{ap}}^{T}(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k)]\right\} \\
&-\mu E\left\{\mathrm{tr}[\tilde{\mathbf{e}}_{\mathrm{ap}}(k)\tilde{\mathbf{e}}_{\mathrm{ap}}^{T}(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\hat{\mathbf{R}}_{\mathrm{ap}}(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k)]\right\} = \\
&\mu E\left\{\mathrm{tr}[\mathbf{n}_{\mathrm{ap}}(k)\mathbf{n}_{\mathrm{ap}}^{T}(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\hat{\mathbf{R}}_{\mathrm{ap}}(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\mathbf{n}_{\mathrm{ap}}(k)]\right\} \qquad (104)
\end{aligned}
$$

where we used the property $\mathrm{tr}[\mathbf{A}\cdot\mathbf{B}] = \mathrm{tr}[\mathbf{B}\cdot\mathbf{A}]$.

Misadjustment in the AP Algorithm



In addition, if matrix $\hat{\mathbf{R}}_{\text{ap}}(k)$ is invertible it can be noticed that

$$\begin{aligned}
\hat{\mathbf{S}}_{\text{ap}}(k) &= \left[\hat{\mathbf{R}}_{\text{ap}}(k) + \gamma\mathbf{I}\right]^{-1} \\
&= \hat{\mathbf{R}}_{\text{ap}}^{-1}(k)\left[\mathbf{I} - \gamma\hat{\mathbf{R}}_{\text{ap}}^{-1}(k) + \gamma^2\hat{\mathbf{R}}_{\text{ap}}^{-2}(k) - \gamma^3\hat{\mathbf{R}}_{\text{ap}}^{-3}(k) + \cdots\right] \\
&\approx \hat{\mathbf{R}}_{\text{ap}}^{-1}(k)\left[\mathbf{I} - \gamma\hat{\mathbf{R}}_{\text{ap}}^{-1}(k)\right] \qquad (105)
\end{aligned}$$

where the last relation is valid for $\gamma \ll 1$.

Misadjustment in the AP Algorithm

By assuming that the matrix $\hat{\mathbf{S}}_{\rm ap}(k)$ is statistically independent of the noiseless *a priori* error after convergence, and of the noise, the equation (104) can be rewritten as

$$
\begin{aligned}
&2\mathrm{tr}\left\{E[\tilde{\mathbf{e}}_{\rm ap}(k)\tilde{\mathbf{e}}^T_{\rm ap}(k)]E[\hat{\mathbf{S}}_{\rm ap}(k)]\right\} - \mu\mathrm{tr}\left\{E[\tilde{\mathbf{e}}_{\rm ap}(k)\tilde{\mathbf{e}}^T_{\rm ap}(k)]E[\hat{\mathbf{S}}_{\rm ap}(k)]\right\} \\
&+\gamma\mu\mathrm{tr}\left\{E[\tilde{\mathbf{e}}_{\rm ap}(k)\tilde{\mathbf{e}}^T_{\rm ap}(k)]\right\} = \\
&\quad \mu\mathrm{tr}\left\{E[\mathbf{n}_{\rm ap}(k)\mathbf{n}^T_{\rm ap}(k)]E[\hat{\mathbf{S}}_{\rm ap}(k)]\right\} - \gamma\mu\mathrm{tr}\left\{E[\mathbf{n}_{\rm ap}(k)\mathbf{n}^T_{\rm ap}(k)]\right\}
\end{aligned}
\tag{106}
$$

Misadjustment in the AP Algorithm

This equation can be further simplified by assuming the noise is white and γ is small leading to the following expression

$$(2-\mu)\mathrm{tr}\{E[\tilde{\mathbf{e}}_{\mathrm{ap}}(k)\tilde{\mathbf{e}}_{\mathrm{ap}}^T(k)]E[\hat{\mathbf{S}}_{\mathrm{ap}}(k)]\} = \mu\sigma_n^2\mathrm{tr}\{E[\hat{\mathbf{S}}_{\mathrm{ap}}(k)]\} \qquad (107)$$

Our task now is to compute $E[\tilde{\mathbf{e}}_{\mathrm{ap}}(k)\tilde{\mathbf{e}}_{\mathrm{ap}}^T(k)]$ where we will assume in the process that this matrix is diagonal dominant

$$E[\tilde{\mathbf{e}}_{\mathrm{ap}}(k)\tilde{\mathbf{e}}_{\mathrm{ap}}^T(k)] = \mathbf{A}E[\tilde{e}_{\mathrm{ap},1}^2(k)] + \mu^2\mathbf{B}\sigma_n^2$$

Misadjustment in the AP Algorithm

Proof:

The i-th rows of equations (89) and (90) are given by

$$\tilde{\varepsilon}_{\mathrm{ap},i}(k) = -\mathbf{x}^T(k-i+1)\Delta\mathbf{w}(k+1) \tag{108}$$

and

$$\tilde{e}_{\mathrm{ap},i}(k) = -\mathbf{x}^T(k-i+1)\Delta\mathbf{w}(k) = e_{\mathrm{ap},i}(k) - n(k-i) \tag{109}$$

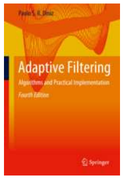
for $i = 1, \ldots, L+1$.

Misadjustment in the AP Algorithm

Using in equation (88) the fact that $\mathbf{X}_{\mathrm{ap}}^{T}(k)\mathbf{F}_{\mathbf{X}}(k) \approx \mathbf{I}$ for small γ, then

$$-\tilde{\boldsymbol{\varepsilon}}_{\mathrm{ap}}(k) = -\tilde{\mathbf{e}}_{\mathrm{ap}}(k) + \mu \mathbf{e}_{\mathrm{ap}}(k) \tag{110}$$

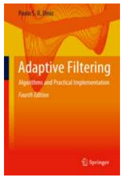
Misadjustment in the AP Algorithm



By properly utilizing in equations (108) and (109) the i-th row of (88), we obtain

$$\begin{aligned}
\tilde{\varepsilon}_{\mathrm{ap},i}(k) &= \mathbf{x}^T(k-i+1)\Delta\mathbf{w}(k+1) \\
&= (1-\mu)\tilde{e}_{\mathrm{ap},i}(k) - \mu n(k-i) \\
&= -(1-\mu)\mathbf{x}^T(k-i)\Delta\mathbf{w}(k) - \mu n(k-i) \qquad (111)
\end{aligned}$$

Misadjustment in the AP Algorithm



Squaring the equation above, assuming the coefficients are weakly dependent on the noise which is in turn white noise, we get

$$E[(\mathbf{x}^T(k-i+1)\Delta\mathbf{w}(k+1))^2] = (1-\mu)^2 E[(\mathbf{x}^T(k-i+1)\Delta\mathbf{w}(k))^2] + \mu^2\sigma_n^2 \tag{112}$$

Misadjustment in the AP Algorithm

The same kind of relation holds for the previous time instant, that is

$$E[(\mathbf{x}^T(k-i)\Delta\mathbf{w}(k))^2] = (1-\mu)^2 E[(\mathbf{x}^T(k-i)\Delta\mathbf{w}(k-1))^2] + \mu^2\sigma_n^2$$

or

$$E[\tilde{e}^2_{\text{ap},i+1}(k)] = (1-\mu)^2 E[\tilde{e}^2_{\text{ap},i}(k-1)] + \mu^2\sigma_n^2 \qquad (113)$$

Misadjustment in the AP Algorithm

Note that for $i = 1$ this term corresponds to the second diagonal element of the matrix $E[\tilde{\mathbf{e}}_{\text{ap}}(k)\tilde{\mathbf{e}}_{\text{ap}}^T(k)]$. Specifically we can compute $E[\tilde{e}^2_{\text{ap},2}(k)]$ as

$$\begin{aligned} E[(\mathbf{x}^T(k-1)\Delta\mathbf{w}(k))^2] &= E[\tilde{e}^2_{\text{ap},2}(k)] \\ &= (1-\mu)^2 E[(\mathbf{x}^T(k-1)\Delta\mathbf{w}(k-1))^2] + \mu^2\sigma_n^2 \\ &= (1-\mu)^2 E[\tilde{e}^2_{\text{ap},1}(k-1)] + \mu^2\sigma_n^2 \qquad (114) \end{aligned}$$

Misadjustment in the AP Algorithm

For $i = 2$ equation (113) becomes

$$\begin{aligned} E[(\mathbf{x}^T(k-2)\Delta\mathbf{w}(k))^2] &= E[\tilde{e}^2_{\text{ap},3}(k)] \\ &= (1-\mu)^2 E[(\mathbf{x}^T(k-2)\Delta\mathbf{w}(k-1))^2] + \mu^2\sigma_n^2 \\ &= (1-\mu)^2 E[\tilde{e}^2_{\text{ap},2}(k-1)] + \mu^2\sigma_n^2 \qquad (115) \end{aligned}$$

Misadjustment in the AP Algorithm

By substituting equation (114) in the equation above it follows that

$$E[\tilde{e}^2_{\text{ap},3}(k)] = (1-\mu)^4 E[\tilde{e}^2_{\text{ap},1}(k-2)] + [1+(1-\mu)^2]\mu^2\sigma_n^2 \quad (116)$$

By induction one can prove that

$$E[\tilde{e}^2_{\text{ap},i+1}(k)] \;=\; (1-\mu)^{2i} E[\tilde{e}^2_{\text{ap},1}(k-i)] + \left[1 + \sum_{l=1}^{i-1}(1-\mu)^{2l}\right]\mu^2\sigma_n^2 \quad (117)$$

Misadjustment in the AP Algorithm

By assuming that $E[\tilde{e}^2_{\text{ap},1}(k)] \approx E[\tilde{e}^2_{\text{ap},1}(k-i)]$ for $i = 1, \ldots, L+1$, then

$$E[\tilde{\mathbf{e}}_{\text{ap}}(k)\tilde{\mathbf{e}}^T_{\text{ap}}(k)] = \mathbf{A}E[\tilde{e}^2_{\text{ap},1}(k)] + \mu^2\mathbf{B}\sigma^2_n \qquad (118)$$

Misadjustment in the AP Algorithm

with

$$\mathbf{A} = \begin{bmatrix} 1 & & & & \\ & (1-\mu)^2 & & \mathbf{0} & \\ & & (1-\mu)^4 & & \\ & \mathbf{0} & & \ddots & \\ & & & & (1-\mu)^{2L} \end{bmatrix}$$

Misadjustment in the AP Algorithm

$$
\mathbf{B} = \begin{bmatrix}
0 & & & & & & \\
 & 1 & & & \mathbf{0} & & \\
 & & 1+(1-\mu)^2 & & & & \\
 & & & \ddots & & & \\
 & \mathbf{0} & & & 1+\sum_{l=1}^{i-1}(1-\mu)^{2l} & & \\
 & & & & & \ddots & \\
 & & & & & & 1+\sum_{l=1}^{L-1}(1-\mu)^{2l}
\end{bmatrix}
$$

Misadjustment in the AP Algorithm

where it was also considered that the matrix $E[\tilde{\mathbf{e}}_{\mathrm{ap}}(k)\tilde{\mathbf{e}}^T_{\mathrm{ap}}(k)]$ above was diagonal dominant. Note from the relation above that the convergence factor μ should be chosen in the range $0 < \mu < 2$, so that the elements of the noiseless *a priori* error remain bounded for any value of L.

Misadjustment in the AP Algorithm

We have available all the quantities required to calculate the excess of MSE in the affine projection algorithm. Specifically, we can substitute the result of equation (118) in equation (107) obtaining

$$
\begin{aligned}
&(2-\mu)\left[E[\tilde{e}^2_{\mathrm{ap},1}(k)]\mathrm{tr}\{\mathbf{A}E[\hat{\mathbf{S}}_{\mathrm{ap}}(k)]\} + \mu^2\sigma_n^2\mathrm{tr}\{\mathbf{B}E[\hat{\mathbf{S}}_{\mathrm{ap}}(k)]\}\right] \\
&= \mu\sigma_n^2\mathrm{tr}\{E[\hat{\mathbf{S}}_{\mathrm{ap}}(k)]\} \qquad (119)
\end{aligned}
$$

The 2nd term on the l-h side can be neglected for high SNR. For small μ this term also becomes smaller than the one r-h side. For μ close to one the referred terms become comparable only for large L. We then neglect the term multiplied by μ^2.

Misadjustment in the AP Algorithm

Since the diagonal elements of $E[\hat{\mathbf{S}}_{\rm ap}(k)]$ are equal and the matrix $\mathbf{A}$ multiplying it on the left-hand side is a diagonal matrix, after a few manipulations it is possible to deduce that

$$\begin{aligned} E[\tilde{e}^2_{{\rm ap},1}(k)] &= \frac{\mu}{2-\mu}\sigma_n^2 \frac{{\rm tr}\{E[\hat{\mathbf{S}}_{\rm ap}(k)]\}}{{\rm tr}\{\mathbf{A}E[\hat{\mathbf{S}}_{\rm ap}(k)]\}} \\ &= \frac{(L+1)\mu}{2-\mu}\frac{1-(1-\mu)^2}{1-(1-\mu)^{2(L+1)}}\sigma_n^2 \end{aligned} \tag{120}$$

Misadjustment in the AP Algorithm

Therefore, the misadjustment for the affine projection algorithm is given by

$$M = \frac{(L+1)\mu}{2-\mu}\frac{1-(1-\mu)^2}{1-(1-\mu)^{2(L+1)}} = \frac{(L+1)[1-(1-\mu)^{2(L+1)}]}{(2-\mu)^2} \quad (121)$$

For large L and small $1-\mu$, this equation can be approximated by

$$M = \frac{(L+1)\mu}{(2-\mu)} \quad (122)$$

Behavior in Nonstationary Environments

In a nonstationary environment the error in the coefficients is described by the following vector

$$\Delta \mathbf{w}(k+1) = \mathbf{w}(k+1) - \mathbf{w}_o(k+1) \tag{123}$$

where $\mathbf{w}_o(k+1)$ is the optimal time-varying vector. For this case, equation (92) becomes

$$\Delta \mathbf{w}(k+1) = \Delta \hat{\mathbf{w}}(k) + \mu \mathbf{X}_{\text{ap}}(k) \left(\mathbf{X}_{\text{ap}}^T(k) \mathbf{X}_{\text{ap}}(k) + \gamma \mathbf{I} \right)^{-1} \mathbf{e}_{\text{ap}}(k) \tag{124}$$

where $\Delta \hat{\mathbf{w}}(k) = \mathbf{w}(k) - \mathbf{w}_o(k+1)$.

Behavior in Nonstationary Environments

By premultiplying the expression above by $\mathbf{X}^T_{\text{ap}}(k)$ it follows that

$$\begin{aligned}
\mathbf{X}^T_{\text{ap}}(k)\Delta\mathbf{w}(k+1) &= \mathbf{X}^T_{\text{ap}}(k)\Delta\hat{\mathbf{w}}(k) \\
&+ \mu\mathbf{X}^T_{\text{ap}}(k)\mathbf{X}_{\text{ap}}(k)\left(\mathbf{X}^T_{\text{ap}}(k)\mathbf{X}_{\text{ap}}(k)+\gamma\mathbf{I}\right)^{-1}\mathbf{e}_{\text{ap}}(k) \\
-\tilde{\boldsymbol{\varepsilon}}_{\text{ap}}(k) &= -\tilde{\mathbf{e}}_{\text{ap}}(k) \\
&+ \mu\mathbf{X}^T_{\text{ap}}(k)\mathbf{X}_{\text{ap}}(k)\left(\mathbf{X}^T_{\text{ap}}(k)\mathbf{X}_{\text{ap}}(k)+\gamma\mathbf{I}\right)^{-1}\mathbf{e}_{\text{ap}}(k) \qquad (125)
\end{aligned}$$

Behavior in Nonstationary Environments

By solving the equation (125), it is possible to show that

$$\frac{1}{\mu}\left(\mathbf{X}_{\mathrm{ap}}^{T}(k)\mathbf{X}_{\mathrm{ap}}(k)\right)^{-1}\left[\tilde{\mathbf{e}}_{\mathrm{ap}}(k)-\tilde{\boldsymbol{\varepsilon}}_{\mathrm{ap}}(k)\right]=$$
$$\left(\mathbf{X}_{\mathrm{ap}}^{T}(k)\mathbf{X}_{\mathrm{ap}}(k)+\gamma\mathbf{I}\right)^{-1}\mathbf{e}_{\mathrm{ap}}(k) \tag{126}$$

Behavior in Nonstationary Environments

Following the same procedure to derive equation (92), we can now substitute equation (126) in equation (124) in order to deduce that

$$\Delta\mathbf{w}(k+1) - \mathbf{X}_{\mathrm{ap}}(k)\left(\mathbf{X}_{\mathrm{ap}}^{T}(k)\mathbf{X}_{\mathrm{ap}}(k)\right)^{-1}\tilde{\mathbf{e}}_{\mathrm{ap}}(k)$$

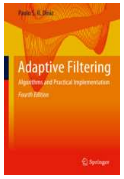
$$= \Delta\hat{\mathbf{w}}(k) - \mathbf{X}_{\mathrm{ap}}(k)\left(\mathbf{X}_{\mathrm{ap}}^{T}(k)\mathbf{X}_{\mathrm{ap}}(k)\right)^{-1}\tilde{\boldsymbol{\varepsilon}}_{\mathrm{ap}}(k) \qquad (127)$$

Behavior in Nonstationary Environments

By computing the energy on both sides of this equation it is possible to show that

$$
\begin{aligned}
& E\left[\|\Delta\mathbf{w}(k+1)\|^2\right] + E\left[\tilde{\mathbf{e}}_{\rm ap}^T(k)\left(\mathbf{X}_{\rm ap}^T(k)\mathbf{X}_{\rm ap}(k)\right)^{-1}\tilde{\mathbf{e}}_{\rm ap}(k)\right] \\
= \ & E\left[\|\Delta\hat{\mathbf{w}}(k)\|^2\right] + E\left[\tilde{\boldsymbol{\varepsilon}}_{\rm ap}^T(k)\left(\mathbf{X}_{\rm ap}^T(k)\mathbf{X}_{\rm ap}(k)\right)^{-1}\tilde{\boldsymbol{\varepsilon}}_{\rm ap}(k)\right]
\end{aligned}
$$

Behavior in Nonstationary Environments



$$= E\left[\|\Delta\mathbf{w}(k) + \Delta\mathbf{w}_o(k+1)\|^2\right] + E\left[\tilde{\boldsymbol{\varepsilon}}^T_{\text{ap}}(k)\left(\mathbf{X}^T_{\text{ap}}(k)\mathbf{X}_{\text{ap}}(k)\right)^{-1}\tilde{\boldsymbol{\varepsilon}}_{\text{ap}}(k)\right]$$

$$\approx E\left[\|\Delta\mathbf{w}(k)\|^2\right] + E\left[\|\Delta\mathbf{w}_o(k+1)\|^2\right]$$
$$+E\left[\tilde{\boldsymbol{\varepsilon}}^T_{\text{ap}}(k)\left(\mathbf{X}^T_{\text{ap}}(k)\mathbf{X}_{\text{ap}}(k)\right)^{-1}\tilde{\boldsymbol{\varepsilon}}_{\text{ap}}(k)\right] \quad (128)$$

where $\Delta\mathbf{w}_o(k+1) = \mathbf{w}_o(k) - \mathbf{w}_o(k+1)$, and in the last equality we have assumed that $E\left[\Delta\mathbf{w}^T(k)\Delta\mathbf{w}_o(k+1)\right] \approx 0$.

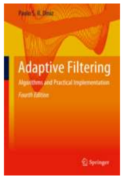
Behavior in Nonstationary Environments

The first-order Markov process is described by

$$\mathbf{w}_o(k) = \lambda_{\mathbf{w}} \mathbf{w}_o(k-1) + \kappa_{\mathbf{w}} \mathbf{n}_{\mathbf{w}}(k) \tag{129}$$

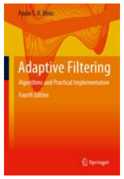
where $\mathbf{n}_{\mathbf{w}}(k)$ is a vector whose elements are zero-mean white noise processes with variance $\sigma_{\mathbf{w}}^2$, and $\lambda_{\mathbf{w}} < 1$. In our derivations of the excess of MSE, the covariance of $\Delta\mathbf{w}_o(k+1) = \mathbf{w}_o(k) - \mathbf{w}_o(k+1)$ is required.

Behavior in Nonstationary Environments



$$
\begin{aligned}
& \mathrm{cov}[\Delta \mathbf{w}_o(k+1)] \\
= \;& E\left[(\mathbf{w}_o(k+1) - \mathbf{w}_o(k))(\mathbf{w}_o(k+1) - \mathbf{w}_o(k))^T\right] \\
= \;& E\left[(\lambda_{\mathbf{w}} \mathbf{w}_o(k) + \kappa_{\mathbf{w}} \mathbf{n}_{\mathbf{w}}(k) - \mathbf{w}_o(k))(\lambda_{\mathbf{w}} \mathbf{w}_o(k) + \kappa_{\mathbf{w}} \mathbf{n}_{\mathbf{w}}(k) - \mathbf{w}_o(k))^T\right] \\
= \;& E\left\{[(\lambda_{\mathbf{w}} - 1)\mathbf{w}_o(k) + \kappa_{\mathbf{w}} \mathbf{n}_{\mathbf{w}}(k)][(\lambda_{\mathbf{w}} - 1)\mathbf{w}_o(k) + \kappa_{\mathbf{w}} \mathbf{n}_{\mathbf{w}}(k)]^T\right\}
\end{aligned} \tag{130}
$$

Behavior in Nonstationary Environments



Since each element of $\mathbf{n}_{\mathbf{w}}(k)$ is a zero-mean white noise process with variance $\sigma^2_{\mathbf{w}}$, and $\lambda_{\mathbf{w}} < 1$, it follows that

$$\begin{aligned}\text{cov}[\Delta\mathbf{w}_o(k+1)] &= \kappa^2_{\mathbf{w}}\sigma^2_{\mathbf{w}}\frac{(1-\lambda_{\mathbf{w}})^2}{1-\lambda^2_{\mathbf{w}}}\mathbf{I} + \kappa^2_{\mathbf{w}}\sigma^2_{\mathbf{w}}\mathbf{I} \\ &= \kappa^2_{\mathbf{w}}\left[\frac{1-\lambda_{\mathbf{w}}}{1+\lambda_{\mathbf{w}}}+1\right]\sigma^2_{\mathbf{w}}\mathbf{I} \end{aligned} \tag{131}$$

Behavior in Nonstationary Environments

By employing this result, we can compute

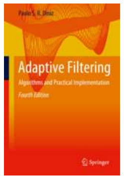
$$
\begin{aligned}
E\left[\|\Delta\mathbf{w}_o(k+1)\|^2\right] &= \operatorname{tr}\{\operatorname{cov}[\Delta\mathbf{w}_o(k+1)]\} \\
&= (N+1)\left[\frac{2\kappa_{\mathbf{w}}^2}{1+\lambda_{\mathbf{w}}}\right]\sigma_{\mathbf{w}}^2 \qquad (132)
\end{aligned}
$$

Behavior in Nonstationary Environments

Again by assuming that the algorithm has converged, that is, the Euclidean norm of the coefficients increment remains in average unchanged, then $E\left[\|\Delta\mathbf{w}(k+1)\|^2\right] = E\left[\|\Delta\mathbf{w}(k)\|^2\right]$. As a result, equation (128) can be rewritten as

$$\begin{aligned} & E\left[\tilde{\mathbf{e}}_{\mathrm{ap}}^T(k)\left(\mathbf{X}_{\mathrm{ap}}^T(k)\mathbf{X}_{\mathrm{ap}}(k)\right)^{-1}\tilde{\mathbf{e}}_{\mathrm{ap}}(k)\right] \\ = \; & E\left[\tilde{\boldsymbol{\varepsilon}}_{\mathrm{ap}}^T(k)\left(\mathbf{X}_{\mathrm{ap}}^T(k)\mathbf{X}_{\mathrm{ap}}(k)\right)^{-1}\tilde{\boldsymbol{\varepsilon}}_{\mathrm{ap}}(k)\right] \\ & +(N+1)\left[\frac{2\kappa_{\mathbf{w}}^2}{1+\lambda_{\mathbf{w}}}\right]\sigma_{\mathbf{w}}^2 \end{aligned} \tag{133}$$

Behavior in Nonstationary Environments



Leading to the equivalent of equation (98) as follows

$$\mu^2 E\left[\mathbf{e}_{\mathrm{ap}}^T(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\hat{\mathbf{R}}_{\mathrm{ap}}(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\mathbf{e}_{\mathrm{ap}}(k)\right]$$
$$= \mu E\left[\tilde{\mathbf{e}}_{\mathrm{ap}}^T(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\mathbf{e}_{\mathrm{ap}}(k) + \mathbf{e}_{\mathrm{ap}}^T(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\tilde{\mathbf{e}}_{\mathrm{ap}}(k)\right]$$
$$+(N+1)\left[\frac{2\kappa_{\mathbf{w}}^2}{1+\lambda_{\mathbf{w}}}\right]\sigma_{\mathbf{w}}^2 \qquad (134)$$

Behavior in Nonstationary Environments

By solving this equation

$$\xi_{\text{lag}} = \frac{N+1}{\mu(2-\mu)} \left[\frac{2\kappa_{\mathbf{w}}^2}{1+\lambda_{\mathbf{w}}} \right] \sigma_{\mathbf{w}}^2 \tag{135}$$

The additional noise and the time-varying parameters to be estimated, the overall excess of MSE is given by

$$\begin{aligned}
\xi_{\text{exc}} &= \frac{(L+1)\mu}{2-\mu} \frac{1-(1-\mu)^2}{1-(1-\mu)^{2(L+1)}} \sigma_n^2 + \frac{N+1}{\mu(2-\mu)} \left[\frac{2\kappa_{\mathbf{w}}^2}{1+\lambda_{\mathbf{w}}} \right] \sigma_{\mathbf{w}}^2 \\
&= \frac{1}{2-\mu} \left\{ (L+1)\mu \frac{1-(1-\mu)^2}{1-(1-\mu)^{2(L+1)}} \sigma_n^2 + \frac{N+1}{\mu} \left[\frac{2\kappa_{\mathbf{w}}^2}{1+\lambda_{\mathbf{w}}} \right] \sigma_{\mathbf{w}}^2 \right\}
\end{aligned} \tag{136}$$

Behavior in Nonstationary Environments

If $\kappa_{\mathbf{w}} = 1$, L is large, and $|1-\mu| < 1$, the expression above becomes simpler

$$\xi_{\text{exc}} = \frac{1}{2-\mu}\left\{(L+1)\mu\sigma_n^2 + \frac{2(N+1)}{\mu(1+\lambda_{\mathbf{w}})}\sigma_{\mathbf{w}}^2\right\} \qquad (137)$$

As can be observed, the contribution due to the lag is inversely proportional to the value of μ. This is an expected result since for small values of μ an adaptive filtering algorithm will face difficulties in tracking the variations in the unknown system.

Transient Behavior

Given the expression

$$
\begin{aligned}
& E\left[\|\Delta\mathbf{w}(k+1)\|^2\right] \\
= \; & E\left[\|\Delta\mathbf{w}(k)\|^2\right] + \mu^2 E\left[\mathbf{e}_{\mathrm{ap}}^T(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\hat{\mathbf{R}}_{\mathrm{ap}}(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\mathbf{e}_{\mathrm{ap}}(k)\right] \\
& -\mu E\left[\tilde{\mathbf{e}}_{\mathrm{ap}}^T(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\mathbf{e}_{\mathrm{ap}}(k) + \mathbf{e}_{\mathrm{ap}}^T(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\tilde{\mathbf{e}}_{\mathrm{ap}}(k)\right] \qquad (138)
\end{aligned}
$$

Transient Behavior

Since from equation (90)

$$\mathbf{e}_{\mathrm{ap}}(k) = \tilde{\mathbf{e}}_{\mathrm{ap}}(k) + \mathbf{n}_{\mathrm{ap}}(k) = -\mathbf{X}_{\mathrm{ap}}^{T}(k)\Delta\mathbf{w}(k) + \mathbf{n}_{\mathrm{ap}}(k)$$

the expression (138) above can be rewritten as

$$\begin{aligned}
&E\left[\|\Delta\mathbf{w}(k+1)\|^2\right] = E\left[\|\Delta\mathbf{w}(k)\|^2\right] \\
&+\mu^2 E\Big[\left(-\Delta\mathbf{w}^T(k)\mathbf{X}_{\mathrm{ap}}(k) + \mathbf{n}_{\mathrm{ap}}^T(k)\right)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\hat{\mathbf{R}}_{\mathrm{ap}}(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k) \\
&\Big(-\mathbf{X}_{\mathrm{ap}}^T(k)\Delta\mathbf{w}(k) + \mathbf{n}_{\mathrm{ap}}(k)\Big)\Big] \\
&-\mu E\Big[\left(-\Delta\mathbf{w}^T(k)\mathbf{X}_{\mathrm{ap}}(k)\right)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\Big(-\mathbf{X}_{\mathrm{ap}}^T(k)\Delta\mathbf{w}(k) + \mathbf{n}_{\mathrm{ap}}(k)\Big) \\
&+\left(-\Delta\mathbf{w}^T(k)\mathbf{X}_{\mathrm{ap}}(k) + \mathbf{n}_{\mathrm{ap}}^T(k)\right)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\Big(-\mathbf{X}_{\mathrm{ap}}^T(k)\Delta\mathbf{w}(k)\Big)\Big] \quad (139)
\end{aligned}$$

Transient Behavior

By considering the noise white and uncorrelated with the other quantities of this recursion, the above equation can be simplified to

$$
\begin{aligned}
& E\left[\|\Delta\mathbf{w}(k+1)\|^2\right] \\
= \; & E\left[\|\Delta\mathbf{w}(k)\|^2\right] - 2\mu E\left[\Delta\mathbf{w}^T(k)\mathbf{X}_{\text{ap}}(k)\hat{\mathbf{S}}_{\text{ap}}(k)\mathbf{X}_{\text{ap}}^T(k)\Delta\mathbf{w}(k)\right] \\
& +\mu^2 E\left[\Delta\mathbf{w}^T(k)\mathbf{X}_{\text{ap}}(k)\hat{\mathbf{S}}_{\text{ap}}(k)\hat{\mathbf{R}}_{\text{ap}}(k)\hat{\mathbf{S}}_{\text{ap}}(k)\mathbf{X}_{\text{ap}}^T(k)\Delta\mathbf{w}(k)\right] \\
& +\mu^2 E\left[\mathbf{n}_{\text{ap}}^T(k)\hat{\mathbf{S}}_{\text{ap}}(k)\hat{\mathbf{R}}_{\text{ap}}(k)\hat{\mathbf{S}}_{\text{ap}}(k)\mathbf{n}_{\text{ap}}(k)\right] \qquad (140)
\end{aligned}
$$

Transient Behavior

By applying the property that $\mathrm{tr}[\mathbf{AB}] = \mathrm{tr}[\mathbf{BA}]$, this relation is equivalent to

$$
\begin{aligned}
& \mathrm{tr}\{\mathrm{cov}[\Delta\mathbf{w}(k+1)]\} \\
= \; & \mathrm{tr}\left[\mathrm{cov}\Delta\mathbf{w}(k)\right] - 2\mu\mathrm{tr}\left\{E\left[\mathbf{X}_{\mathrm{ap}}(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\mathbf{X}_{\mathrm{ap}}^{T}(k)\Delta\mathbf{w}(k)\Delta\mathbf{w}^{T}(k)\right]\right\} \\
& +\mu^{2}\mathrm{tr}\left\{E\left[\mathbf{X}_{\mathrm{ap}}(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\hat{\mathbf{R}}_{\mathrm{ap}}(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\mathbf{X}_{\mathrm{ap}}^{T}(k)\Delta\mathbf{w}(k)\Delta\mathbf{w}^{T}(k)\right]\right\} \\
& +\mu^{2}\mathrm{tr}\left\{E\left[\hat{\mathbf{S}}_{\mathrm{ap}}(k)\hat{\mathbf{R}}_{\mathrm{ap}}(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\right]E\left[\mathbf{n}_{\mathrm{ap}}(k)\mathbf{n}_{\mathrm{ap}}^{T}(k)\right]\right\} \qquad (141)
\end{aligned}
$$

Transient Behavior

By assuming that the $\Delta\mathbf{w}(k+1)$ is independent of the data and the noise is white, it follows that

$$
\begin{aligned}
& \operatorname{tr}\{\operatorname{cov}[\Delta\mathbf{w}(k+1)]\} \\
= \; & \operatorname{tr}\left\{\left[\mathbf{I} - E\left(2\mu\mathbf{X}_{\text{ap}}(k)\hat{\mathbf{S}}_{\text{ap}}(k)\mathbf{X}_{\text{ap}}^{T}(k)\right.\right.\right. \\
& \left.\left.\left. -\mu^2\mathbf{X}_{\text{ap}}(k)\hat{\mathbf{S}}_{\text{ap}}(k)\hat{\mathbf{R}}_{\text{ap}}(k)\hat{\mathbf{S}}_{\text{ap}}(k)\mathbf{X}_{\text{ap}}^{T}(k)\right)\right]\operatorname{cov}[\Delta\mathbf{w}(k)]\right\} \\
& +\mu^2\sigma_n^2\operatorname{tr}\left\{E\left[\hat{\mathbf{S}}_{\text{ap}}(k)\hat{\mathbf{R}}_{\text{ap}}(k)\hat{\mathbf{S}}_{\text{ap}}(k)\right]\right\} \qquad (142)
\end{aligned}
$$

Transient Behavior

Now by recalling that

$$\hat{\mathbf{S}}_{\rm ap}(k) \approx \hat{\mathbf{R}}_{\rm ap}^{-1}(k)\left[\mathbf{I} - \gamma\hat{\mathbf{R}}_{\rm ap}^{-1}(k)\right]$$

and by utilizing the unitary matrix $\mathbf{Q}$, that in the present discussion diagonalizes
$E[\mathbf{X}_{\rm ap}(k)\hat{\mathbf{S}}_{\rm ap}(k)\mathbf{X}_{\rm ap}^{T}(k)]$, the following relation is valid

Transient Behavior

$$
\begin{aligned}
& \operatorname{tr}\{\operatorname{cov}[\Delta\mathbf{w}(k+1)]\mathbf{Q}\mathbf{Q}^T\} \\
= \; & \operatorname{tr}\left\{\mathbf{Q}\mathbf{Q}^T\left[\mathbf{I} - E\left(2\mu\mathbf{X}_{\mathrm{ap}}(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\mathbf{X}_{\mathrm{ap}}^T(k) - (1-\gamma)\mu^2\mathbf{X}_{\mathrm{ap}}(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\mathbf{X}_{\mathrm{ap}}^T(k)\right)\right]\right. \\
+ \; & (1-\gamma)\mu^2\sigma_n^2\operatorname{tr}\left\{E\left[\hat{\mathbf{S}}_{\mathrm{ap}}(k)\right]\right\}
\end{aligned}
$$

Transient Behavior

Again by applying the property that $\mathrm{tr}[\mathbf{AB}] = \mathrm{tr}[\mathbf{BA}]$ and assuming γ small, it follows that

$$
\begin{aligned}
& \mathrm{tr}\{\mathbf{Q}^T \mathrm{cov}[\Delta\mathbf{w}(k+1)]\mathbf{Q}\} \\
= \; & \mathrm{tr}\left\{\mathbf{Q}\left[\mathbf{I} - \mathbf{Q}^T E\left(2\mu\mathbf{X}_{\mathrm{ap}}(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\mathbf{X}_{\mathrm{ap}}^T(k)\right.\right.\right. \\
& - \left.\left.\mu^2\mathbf{X}_{\mathrm{ap}}(k)\hat{\mathbf{S}}_{\mathrm{ap}}(k)\mathbf{X}_{\mathrm{ap}}^T(k)\right)\mathbf{Q}\right]\mathbf{Q}^T\mathrm{cov}[\Delta\mathbf{w}(k)]\mathbf{Q}\mathbf{Q}^T\Big\} \\
& + \mu^2\sigma_n^2 \mathrm{tr}\left\{E\left[\hat{\mathbf{S}}_{\mathrm{ap}}(k)\right]\right\} \qquad (144)
\end{aligned}
$$

$$\Delta \mathbf{w}'(k+1) = \Delta \mathbf{w}(k+1)\mathbf{Q}$$

$$\begin{aligned}
\text{tr}\{\text{cov}[\Delta \mathbf{w}'(k+1)]\} &= \text{tr}\Big\{\mathbf{Q}^T\mathbf{Q}\Big[\mathbf{I} - \mathbf{Q}^T E\Big(-2\mu \mathbf{X}_{\text{ap}}(k)\hat{\mathbf{S}}_{\text{ap}}(k)\mathbf{X}_{\text{ap}}^T(k) \\
&- \ \mu^2 \mathbf{X}_{\text{ap}}(k)\hat{\mathbf{S}}_{\text{ap}}(k)\mathbf{X}_{\text{ap}}^T(k)\Big)\mathbf{Q}\Big]\,\text{cov}[\Delta \mathbf{w}'(k)]\Big\} \\
&+\mu^2\sigma_n^2 \text{tr}\Big\{E\Big[\hat{\mathbf{S}}_{\text{ap}}(k)\Big]\Big\} \\
&= \text{tr}\Big\{\Big[\mathbf{I} - 2\mu\hat{\mathbf{\Lambda}} + \mu^2\hat{\mathbf{\Lambda}}\Big]\,\text{cov}[\Delta \mathbf{w}'(k)]\Big\} \\
&+\mu^2\sigma_n^2 \text{tr}\Big\{E\Big[\hat{\mathbf{S}}_{\text{ap}}(k)\Big]\Big\}
\end{aligned} \tag{145}$$

where $\hat{\mathbf{\Lambda}}$ have the eigenvalues of $E[\mathbf{X}_{\text{ap}}(k)\hat{\mathbf{S}}_{\text{ap}}(k)\mathbf{X}_{\text{ap}}^T(k)]$, $\hat{\lambda}_i$.

Transient Behavior

By using the assumption that $\text{cov}[\Delta\mathbf{w}'(k+1)]$ and $\hat{\mathbf{S}}_{\text{ap}}(k)$ are diagonal dominant, we can disregard the trace operator and observe that the geometric decaying curves have ratios $r_{\text{cov}[\Delta\mathbf{w}(k)]} = (1 - 2\mu\hat{\lambda}_i + \mu^2\hat{\lambda}_i)$. As a result, it is possible to infer that the convergence time constant is given by

$$\tau_{ei} = \tau_{\text{cov}[\Delta\mathbf{w}(k)]} = \frac{1}{\mu\hat{\lambda}_i}\frac{1}{2-\mu} \tag{146}$$

The time constants for error convergence are dependent on the inverse of the eigenvalues of $E[\mathbf{X}_{\text{ap}}(k)\hat{\mathbf{S}}_{\text{ap}}(k)\mathbf{X}^T_{\text{ap}}(k)]$. However, since μ is not constrained by these eigenvalues.

Complex Affine Projection Algorithm

Using the method of Lagrange multipliers to transform the constrained minimization into an unconstrained one, the unconstrained function to be minimized is

$$\begin{aligned} F[\mathbf{w}(k+1)] &= \frac{1}{2}\|\mathbf{w}(k+1)-\mathbf{w}(k)\|^2 \\ &\quad +\mathrm{re}\left\{\boldsymbol{\lambda}_{\mathrm{ap}}^T(k)[\mathbf{d}_{\mathrm{ap}}(k)-\mathbf{X}_{\mathrm{ap}}^H(k)\mathbf{w}^*(k+1)]\right\} \end{aligned} \tag{147}$$

where $\boldsymbol{\lambda}_{\mathrm{ap}}(k)$ is a complex $(L+1)\times 1$ vector of Lagrange multipliers, and the real part operator is required in order to turn the overall objective function real valued.

Complex Affine Projection Algorithm

The expression above can be rewritten as

$$
\begin{aligned}
F[\mathbf{w}(k+1)] \;=\; & \frac{1}{2}[\mathbf{w}(k+1)-\mathbf{w}(k)]^H\left[\mathbf{w}(k+1)-\mathbf{w}(k)\right] \\
& +\frac{1}{2}\boldsymbol{\lambda}_{\mathrm{ap}}^T(k)\left[\mathbf{d}_{\mathrm{ap}}^*(k)-\mathbf{X}_{\mathrm{ap}}^H(k)\mathbf{w}(k+1)\right] \\
& +\frac{1}{2}\boldsymbol{\lambda}_{\mathrm{ap}}^H(k)\left[\mathbf{d}_{\mathrm{ap}}(k)-\mathbf{X}_{\mathrm{ap}}^T(k)\mathbf{w}^*(k+1)\right] \qquad (148)
\end{aligned}
$$

Complex Affine Projection Algorithm

The gradient of $F[\mathbf{w}(k+1)]$ with respect to $\mathbf{w}^*(k+1)$ is given by

$$\begin{aligned}\frac{\partial F[\mathbf{w}(k+1)]}{\partial \mathbf{w}^*(k+1)} &= \mathbf{g}_{\mathbf{w}^*}\{F[\mathbf{w}(k+1)]\} \\ &= \frac{1}{2}\left[\mathbf{w}(k+1) - \mathbf{w}(k)\right] - \frac{1}{2}\mathbf{X}_{\mathrm{ap}}(k)\boldsymbol{\lambda}^*_{\mathrm{ap}}(k) \quad (149)\end{aligned}$$

After setting the gradient of $F[\mathbf{w}(k+1)]$ with respect to $\mathbf{w}^*(k+1)$ equal to zero, the expression below follows

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mathbf{X}_{\mathrm{ap}}(k)\boldsymbol{\lambda}^*_{\mathrm{ap}}(k) \quad (150)$$

Complex Affine Projection Algorithm

By replacing equation (150) in the constraint relation $\mathbf{d}_{\text{ap}}^{*}(k) - \mathbf{X}_{\text{ap}}^{H}(k)\mathbf{w}(k+1) = \mathbf{0}$, we generate the expression

$$\mathbf{X}_{\text{ap}}^{H}(k)\mathbf{X}_{\text{ap}}(k)\boldsymbol{\lambda}_{\text{ap}}^{*}(k) = \mathbf{d}_{\text{ap}}^{*}(k) - \mathbf{X}_{\text{ap}}^{H}(k)\mathbf{w}(k) = \mathbf{e}_{\text{ap}}^{*}(k) \qquad (151)$$

The update equation is now given by equation (150) with $\boldsymbol{\lambda}_{\text{ap}}(k)$ being the solution of equation (151), i.e.,

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mathbf{X}_{\text{ap}}(k)\left(\mathbf{X}_{\text{ap}}^{H}(k)\mathbf{X}_{\text{ap}}(k)\right)^{-1}\mathbf{e}_{\text{ap}}^{*}(k) \qquad (152)$$

Complex Affine Projection Algorithm

Including a convergence factor

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu \mathbf{X}_{\mathrm{ap}}(k) \left(\mathbf{X}_{\mathrm{ap}}^{H}(k) \mathbf{X}_{\mathrm{ap}}(k) \right)^{-1} \mathbf{e}_{\mathrm{ap}}^{*}(k) \qquad (153)$$

Complex Affine Projection Algorithm

Algorithm 6.6

Complex Affine Projection Algorithm

Initialization

$\mathbf{x}(0) = \mathbf{w}(0) = [0\,0 \ldots 0]^T$

choose μ in the range $0 < \mu \leq 2$

γ = small constant

Do for $k \geq 0$

$\mathbf{e}^*_{\mathrm{ap}}(k) = \mathbf{d}^*_{\mathrm{ap}}(k) - \mathbf{X}^H_{\mathrm{ap}}(k)\mathbf{w}(k)$

$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu \mathbf{X}_{\mathrm{ap}}(k)\left(\mathbf{X}^H_{\mathrm{ap}}(k)\mathbf{X}_{\mathrm{ap}}(k) + \gamma \mathbf{I}\right)^{-1} \mathbf{e}^*_{\mathrm{ap}}(k)$

Analytical Examples

Example 4.3: Stochastic Gradient Algorithm

Derive the update equation for a stochastic gradient algorithm designed to minimize the following objective function.

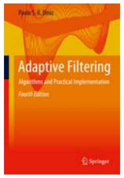
$$E\left[F[\mathbf{w}(k)]\right] = E\left[a|d(k) - \mathbf{w}_1^H(k)\mathbf{x}(k)|^4 + b|d(k) - \mathbf{w}_2^T(k)\mathbf{x}(k)|^4\right]$$

where

$$\mathbf{w}(k) = \begin{bmatrix} \mathbf{w}_1(k) \\ \mathbf{w}_2(k) \end{bmatrix}$$

and $\mathbf{w}_2(k)$ is a vector with real-valued entries. The parameters a and b are also real.

Analytical Examples



Solution:

The given objective function can be rewritten as

$$\begin{aligned} F[\mathbf{w}(k)] &= a\left\{(d(k)-\mathbf{w}_1^H(k)\mathbf{x}(k))^2(d^*(k)-\mathbf{w}_1^T(k)\mathbf{x}^*(k))^2\right\} \\ &\quad +b\left\{(d(k)-\mathbf{w}_2^T(k)\mathbf{x}(k))^2(d^*(k)-\mathbf{w}_2^T(k)\mathbf{x}^*(k))^2\right\} \end{aligned}$$

where by denoting $e_1(k) = d(k) - \mathbf{w}_1^H(k)\mathbf{x}(k)$ and
$e_2(k) = d(k) - \mathbf{w}_2^T(k)\mathbf{x}(k)$.

Analytical Examples

It is possible to compute the gradient expression as

$$\mathbf{g}_{\mathbf{w}^*}\{F[\mathbf{w}(k)]\} \quad = \quad \begin{bmatrix} -2ae_1^*(k)\mathbf{x}(k)|e_1(k)|^2 \\ -2be_2^*(k)\mathbf{x}(k)|e_2(k)|^2 - 2be_2(k)\mathbf{x}^*(k)|e_2(k)|^2 \end{bmatrix}$$

Analytical Examples

The updating equation is then given by

$$\begin{aligned}
\mathbf{w}(k+1) &= \mathbf{w}(k) - \mu \begin{bmatrix} -2ae_1^*(k)\mathbf{x}(k)|e_1(k)|^2 \\ -4b\ \mathrm{re}\left[e_2^*(k)\mathbf{x}(k)\right]|e_2(k)|^2 \end{bmatrix} \\
&= \mathbf{w}(k) + \mu \begin{bmatrix} 2ae_1^*(k)\mathbf{x}(k)|e_1(k)|^2 \\ 4b\ \mathrm{re}\left[e_2^*(k)\mathbf{x}(k)\right]|e_2(k)|^2 \end{bmatrix}
\end{aligned}$$

Analytical Examples

Example 4.4: Normalized LMS Algorithm

(a) A normalized LMS algorithm using convergence factor equal to one has the following data available

$$\mathbf{x}(0) = \begin{bmatrix} 2+\epsilon_1 \\ 2 \end{bmatrix}$$

$$d(0) = 1$$

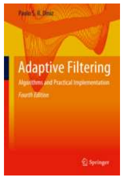
and

$$\mathbf{x}(1) = \begin{bmatrix} 1 \\ 1 + \epsilon_2 \end{bmatrix}$$

$$d(1) = 0$$

where the initial values for the coefficients are zero and ϵ_1 and ϵ_2 are real-valued constants.

Analytical Examples



Determine the hyperplanes

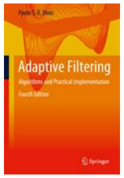
$$\mathcal{S}(k) = \{\mathbf{w}(k+1) \in \mathbb{R}^2 : d(k) - \mathbf{w}^T(k+1)\mathbf{x}(k) = 0\}$$

for two updates.

(b) If the given data belong to an identification problem without additional noise, what would be the coefficients of the unknown system?

(c) What would be the solution if $\epsilon_1 = \epsilon_2 = 0$?

Analytical Examples



Solution:

(a) The hyperplanes defined by the given data vectors are respectively given by (a) The hyperplanes defined by the given data vectors are respectively given by

$$\mathcal{S}(0) = \{\mathbf{w}(1) \in \mathbb{R}^2 : 1 - (2 + \epsilon_1)w_0(1) - 2w_1(1) = 0\}$$

and

$$\mathcal{S}(1) = \{\mathbf{w}(2) \in \mathbb{R}^2 : 0 - w_0(2) - (1 + \epsilon_2)w_1(2) = 0\}$$

Analytical Examples

(b) The solution lies on $\mathcal{S}(0) \cap \mathcal{S}(1)$. Thus

$$
\begin{aligned}
(2+\epsilon_1)w_0 + 2w_1 &= 1 \\
w_0 + (1+\epsilon_2)w_1 &= 0
\end{aligned}
$$

whose solution is

$$
\mathbf{w}_o = \begin{bmatrix} \frac{1+\epsilon_2}{\epsilon_1+\epsilon_1\epsilon_2+2\epsilon_2} \\ \frac{-1}{\epsilon_1+\epsilon_1\epsilon_2+2\epsilon_2} \end{bmatrix}
$$

assuming $\epsilon_1 \neq 0$ and $\epsilon_2 \neq 0$.

(c) For $\epsilon_1 = \epsilon_2 = 0$ the hyperplanes $\mathcal{S}(1)$ and $\mathcal{S}(2)$ are parallel and the solution before is not valid. In this case there is no solution.

Analytical Examples

Example 4.5: Complex Normalized LMS Algorithm

Which objective function is actually minimized by the complex normalized LMS algorithm with regularization factor γ and convergence factor μ_n?

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{\mu_n}{\gamma + \mathbf{x}^H(k)\mathbf{x}(k)}\mathbf{x}(k)\mathrm{e}^*(k) \tag{154}$$

Assume that γ is included for regularization purposes.

Analytical Examples

Solution:

Define

$$\alpha = \left(\frac{1}{\mu_n} - 1 + \alpha_p \gamma \right) \tag{155}$$

The objective function to be minimized with respect to the coefficients $\mathbf{w}^*(k+1)$ is given by

$$\xi(k) = \alpha \|\mathbf{w}(k+1) - \mathbf{w}(k)\|^2 + \alpha_p \|d(k) - \mathbf{x}^T(k)\mathbf{w}^*(k+1)\|^2 \tag{156}$$

Analytical Examples

where

$$\alpha_p = \frac{1}{\gamma + \mathbf{x}^H(k)\mathbf{x}(k)} \tag{157}$$

This result can be verified by computing the derivative of the objective function with respect to $\mathbf{w}^*(k+1)$ as following described.

$$\frac{\partial \xi(k)}{\partial \mathbf{w}^*(k+1)} = \alpha[\mathbf{w}(k+1) - \mathbf{w}(k)] - \alpha_p \mathbf{x}(k)\left[d^*(k) - \mathbf{x}^H(k)\mathbf{w}(k+1)\right]$$

Analytical Examples

By setting this result to zero it follows that

$$\begin{aligned}\left[\alpha\mathbf{I}+\alpha_p\mathbf{x}(k)\mathbf{x}^H(k)\right]\mathbf{w}(k+1) &= \alpha\mathbf{w}(k)+\alpha_p\mathbf{x}(k)d^*(k)-\alpha_p\mathbf{x}(k)\mathbf{x}^H(k)\mathbf{w}(k)\\ &\quad +\alpha_p\mathbf{x}(k)\mathbf{x}^H(k)\mathbf{w}(k)\\ &= \left[\alpha\mathbf{I}+\alpha_p\mathbf{x}(k)\mathbf{x}^H(k)\right]\mathbf{w}(k)+\alpha_p\mathbf{x}(k)e^*(k)\end{aligned}$$

This equation can be rewritten as

$$\mathbf{w}(k+1)=\mathbf{w}(k)+\alpha_p\left[\alpha\mathbf{I}+\alpha_p\mathbf{x}(k)\mathbf{x}^H(k)\right]^{-1}\mathbf{x}(k)e^*(k) \qquad (158)$$

Analytical Examples

After applying the matrix inversion lemma we get

$$
\begin{aligned}
\left[\alpha \mathbf{I} + \alpha_p \mathbf{x}(k)\mathbf{x}^H(k)\right]^{-1} &= \frac{\mathbf{I}}{\alpha} - \frac{\mathbf{I}}{\alpha}\mathbf{x}(k)\left[\frac{\mathbf{x}^H(k)\mathbf{x}(k)}{\alpha} + \frac{1}{\alpha_p}\right]^{-1}\mathbf{x}^H(k)\frac{\mathbf{I}}{\alpha} \\
&= \frac{1}{\alpha}\left[\mathbf{I} - \frac{\mathbf{x}(k)\mathbf{x}^H(k)}{\mathbf{x}^H(k)\mathbf{x}(k) + \frac{\alpha}{\alpha_p}}\right]
\end{aligned}
$$

Analytical Examples

Since the above equation will be multiplied on the right-hand side by $\mathbf{x}(k)$, then

$$
\begin{aligned}
\frac{1}{\alpha}\left[\mathbf{I}-\frac{\mathbf{x}(k)\mathbf{x}^H(k)}{\mathbf{x}^H(k)\mathbf{x}(k)+\frac{\alpha}{\alpha_p}}\right]\mathbf{x}(k) &= \frac{1}{\alpha}\left[\frac{\alpha}{\alpha_p}\frac{\mathbf{x}(k)}{\mathbf{x}^H(k)\mathbf{x}(k)+\frac{\alpha}{\alpha_p}}\right] \\
&= \frac{\mathbf{x}(k)}{\alpha_p\mathbf{x}^H(k)\mathbf{x}(k)+\alpha}
\end{aligned}
$$

Analytical Examples

By employing the relation $\alpha = \left(\frac{1}{\mu_n} - 1 + \alpha_p \gamma\right)$ it follows that

$$\frac{\mathbf{x}(k)}{\alpha_p \mathbf{x}^H(k)\mathbf{x}(k) + \alpha} = \mu_n \mathbf{x}(k)$$

By replacing this result in equation (158),

$$\begin{aligned}
\mathbf{w}(k+1) &= \mathbf{w}(k) + \mu_n \alpha_p \mathbf{x}(k) e^*(k) \\
&= \mathbf{w}(k) + \mu_n \left(\gamma + \mathbf{x}^H(k)\mathbf{x}(k)\right)^{-1} \mathbf{x}(k) e^*(k)
\end{aligned}$$

Analytical Examples

Example 4.6: Transform-Domain LMS algorithm

A transform-domain LMS algorithm is used in an application requiring two coefficients and employing the DCT.

(a) Show in detail the update equation related to each adaptive filter coefficient as a function of the input signal, given γ and σ_x^2, where the former is the regularization factor and the latter is the variance of the input signal $x(k)$.

(b) Which value of μ would generate an *a posteriori* error equal to zero?

Analytical Examples

Solution:

(a) The transform matrix in this case is given by

$$\mathbf{T} = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix}$$

The update equation of the first coefficient is

$$\begin{aligned} \hat{w}_0(k+1) &= \hat{w}_0(k) + \frac{2\mu}{\gamma + \sigma_0^2(k)} e(k) s_0(k) \\ &= \hat{w}_0(k) + \frac{2\mu}{\sqrt{2}(\gamma + \sigma_0^2(k))} e(k)(x_0(k) + x_1(k)) \end{aligned}$$

Analytical Examples

and of the second coefficient is

$$\begin{aligned}\hat{w}_1(k+1) &= \hat{w}_1(k) + \frac{2\mu}{\gamma + \sigma_1^2(k)} e(k) s_1(k) \\ &= \hat{w}_1(k) + \frac{2\mu}{\sqrt{2}(\gamma + \sigma_1^2(k))} e(k)(x_0(k) - x_1(k))\end{aligned}$$

where $\sigma_0^2(k) = \sigma_1^2(k) = \frac{1}{2}\sigma_{x_0}^2(k) + \frac{1}{2}\sigma_{x_1}^2(k)$. These variances are estimated by $\sigma_{x_i}^2(k) = \alpha x_i^2(k) + (1-\alpha)\sigma_{x_i}^2(k-1)$, for $i = 0, 1$, α is a small factor chosen in the range $0 < \alpha \leq 0.1$, and γ is the regularization factor.

Analytical Examples

(b) In matrix form the above updating equation can be rewritten as

$$\hat{\mathbf{w}}(k+1) = \hat{\mathbf{w}}(k) + 2\mu e(k)\boldsymbol{\Sigma}^{-2}(k)\mathbf{s}(k) \tag{159}$$

where $\boldsymbol{\Sigma}^{-2}(k)$ is a diagonal matrix added to the regularization factor γ. By replacing the above expression in the *a posteriori* error definition, it follows that

$$\begin{aligned} \varepsilon(k) &= d(k) - \mathbf{s}^T(k)\hat{\mathbf{w}}(k+1) \\ &= d(k) - \mathbf{s}^T(k)\hat{\mathbf{w}}(k) - 2\mu e(k)\mathbf{s}^T(k)\boldsymbol{\Sigma}^{-2}(k)\mathbf{s}(k) = 0 \end{aligned}$$

leading to

$$\mu = \frac{1}{2\mathbf{s}^T(k)\boldsymbol{\Sigma}^{-2}(k)\mathbf{s}(k)}$$

The Affine Projection Algorithm

Example 4.7

An adaptive filtering algorithm is used to identify the system described in using the affine projection algorithm using $L = 0$, $L = 1$ and $L = 4$.

The Affine Projection Algorithm

Solution:

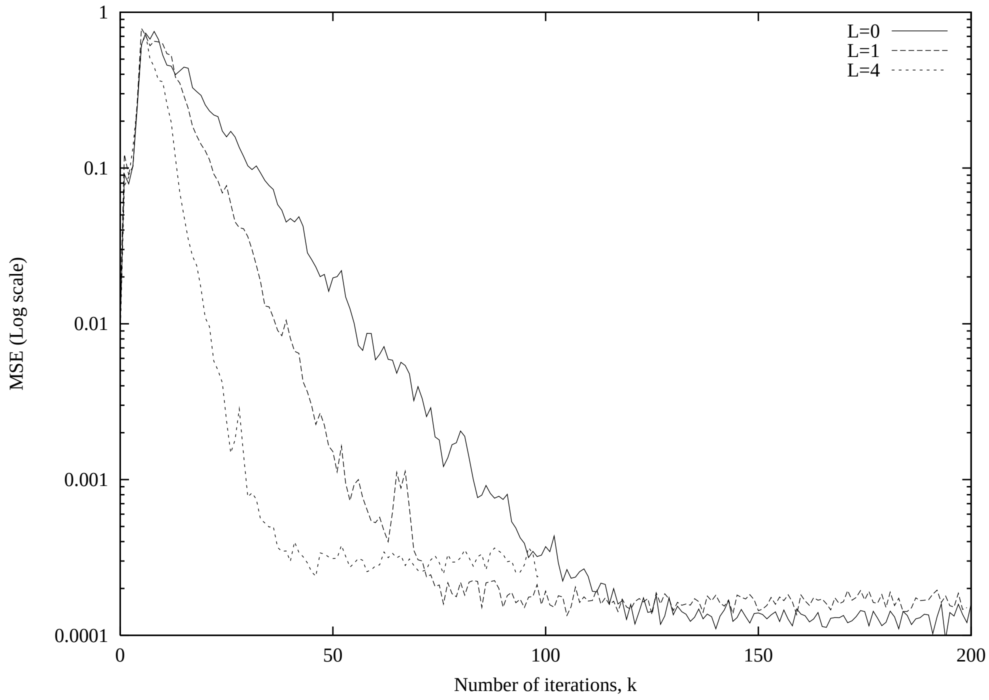


Figure 13: Learning curves for the affine projection algorithms for $L = 0$, $L = 1$, and $L = 4$, eigenvalue spread equal 1.

The Affine Projection Algorithm

Table 1: Evaluation of the Affine Projection Algorithm, $\mu = 0.4$

$\frac{\lambda_{max}}{\lambda_{min}}$	Misadjustment, $L = 0$	Misadjustment, $L = 1$	Misadjustment, $L = 4$
1	0.32	0.67	2.05
20	0.35	0.69	2.29
80	0.37	0.72	2.43

The Affine Projection Algorithm

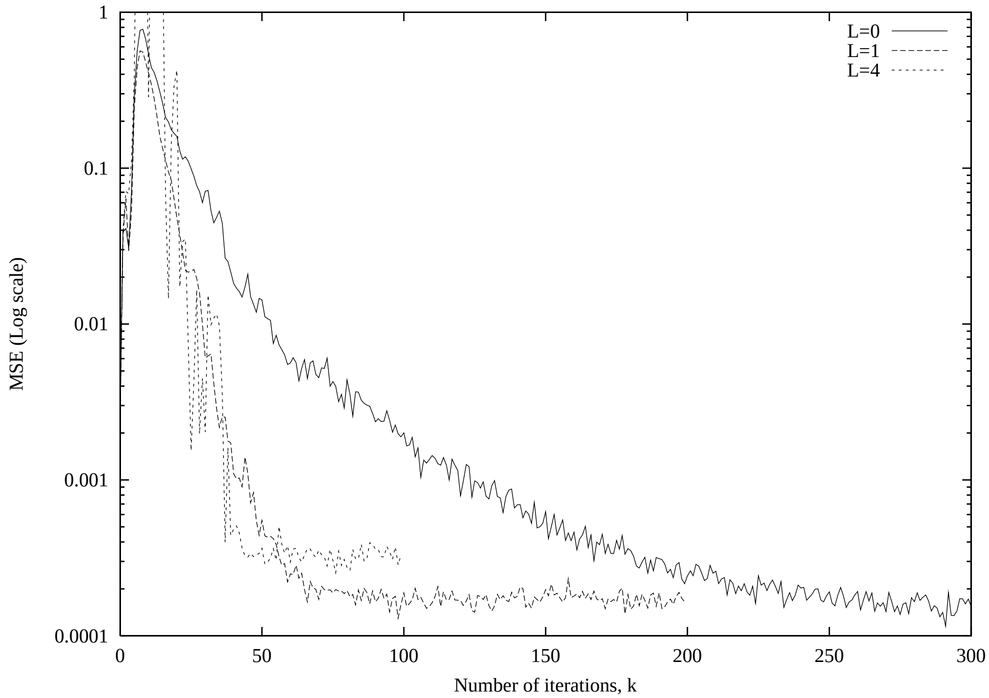


Figure 14: Learning curves for the affine projection algorithms for $L = 0$, $L = 1$, and $L = 4$, eigenvalue spread equal 80.

The Affine Projection Algorithm

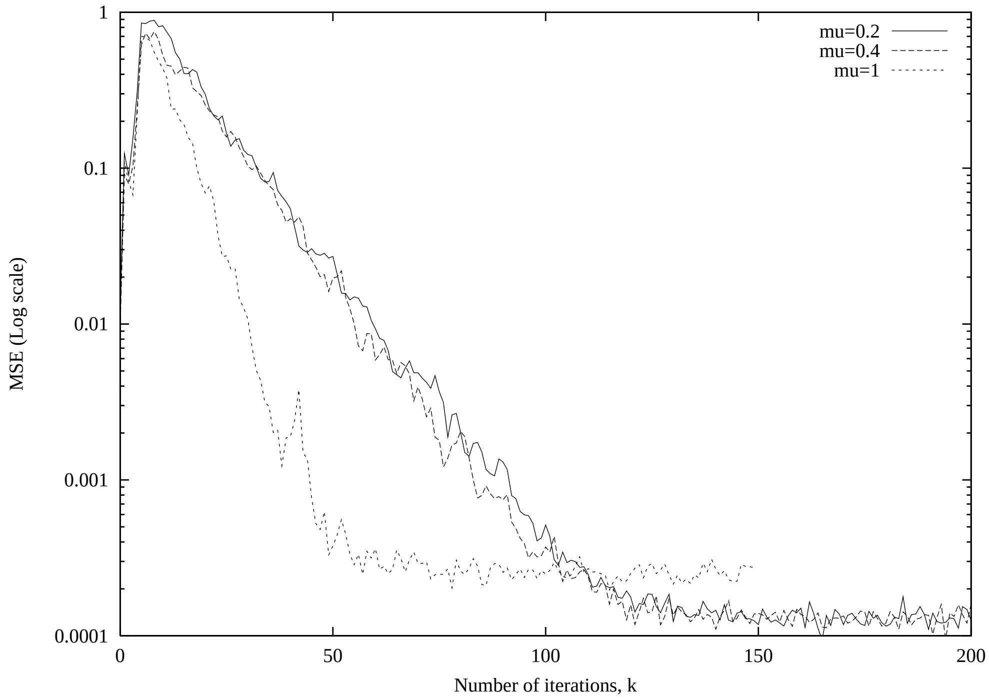


Figure 15: Learning curves for the affine projection algorithms for $\mu = 0.2$, $\mu = 0.4$, and $\mu = 1$.

System Identification Simulations

Transform-Domain Algorithm

The transform domain LMS algorithm was also employed to identify the system described in the LMS Chapter. We run the algorithm with $\mu = 0.01$, with $\alpha = 0.05$ and $\gamma = 10^{-6}$. In Fig. 16 are illustrated the learning curves for the eigenvalue spreads 20 and 80.

System Identification Simulations

The improvement is achieved without increasing the misadjustment as can be verified in Table 2.

The finite precision implementation of the transform domain LMS algorithm presents similar performance to the LMS algorithm, see Tables 3.2 and 3. Notice that an eigenvalue spread of one was used in this example. μ was 0.01, $\gamma = 2^{-b_d}$ and $\alpha = 0.05$.

System Identification Simulations

Table 2: Evaluation of the Transform-Domain LMS Algorithm

$\frac{\lambda_{max}}{\lambda_{min}}$	Misadjustment
1	0.2027
20	0.2037
80	0.2093

System Identification Simulations

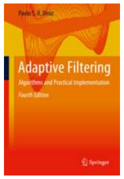


Table 3: Results of the Finite Precision Implementation of the Transform-Domain LMS Algorithm

	$E[\xi(k)_Q]$	$E[\|\|\Delta\mathbf{w}(k)_Q\|\|^2]$
No of bits	Experiment	Experiment
16	$1.627\ 10^{-3}$	$1.313\ 10^{-4}$
12	$1.640\ 10^{-3}$	$1.409\ 10^{-4}$
10	$1.648\ 10^{-3}$	$1.536\ 10^{-4}$

System Identification Simulations

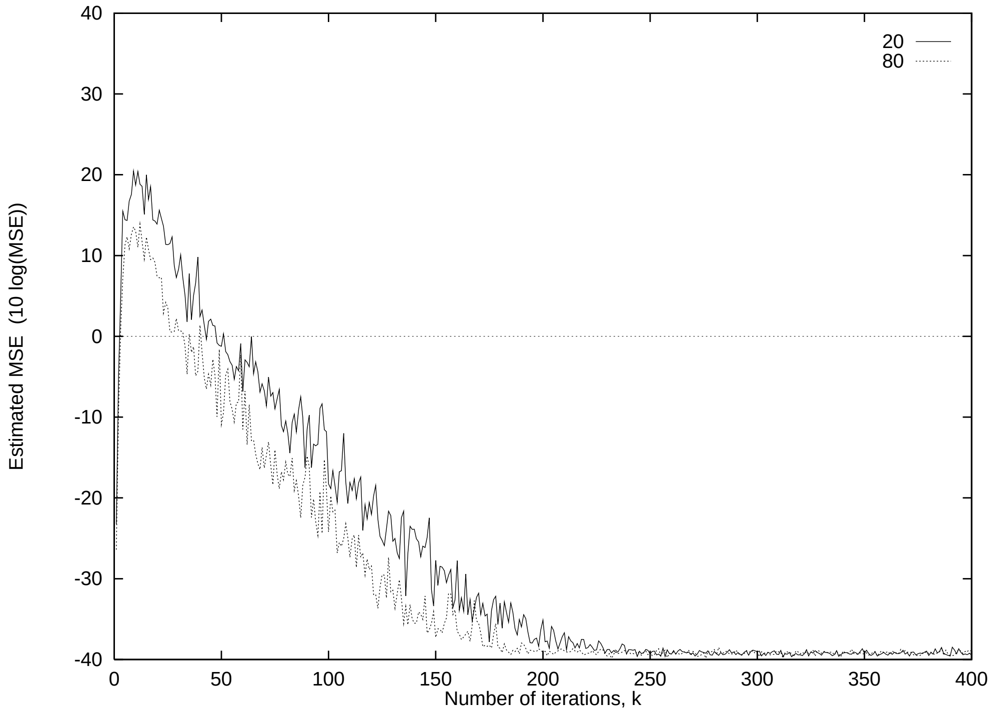


Figure 16: Learning curves for the transform-domain LMS algorithm for eigenvalue spreads: 20 and 80.

Signal Enhancement Simulations

In a signal enhancement problem, the reference signal is

$$r(k) = sin 0.2\pi k + n_r(k)$$

where $n_r(k)$ is a zero mean Gaussian white noise with variance $\sigma^2_{n_r} = 1$. The input signal is given by $n_r(k)$ filtered by filter with

$$H(z) = 0.6 \frac{z - 0.9}{z^2 - 1.36z + 0.79}$$

The adaptive filter is a fourth-order FIR filter. In all examples, a delay $L = 2$ was applied to the reference signal.

Signal Enhancement Simulations

LMS-Based Algorithms

Using the sign error, power-of-two error with $b_d = 12$, and normalized LMS algorithms

(a) Choose an appropriate μ in each case, run an ensemble of 50 experiments, and plot the average learning curve.

(b) Plot the output errors and comment on the results.

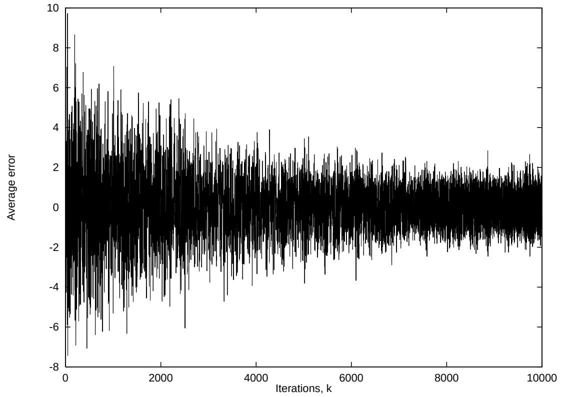
Signal Enhancement Simulations

The values of μ for the sign error and power-of-two LMS algorithms were chosen 0.0207 and 0.0204, respectively.

The coefficients of the adaptive filter were initialized with zero.

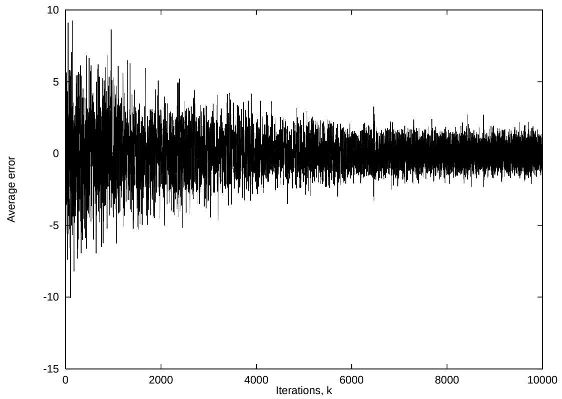
For the normalized LMS algorithm $\mu_n = 0.4$ and $\gamma = 10^{-6}$ were used. Fig. 17 depicts the learning curves for the three algorithms. The results are similar. The reader should notice that the MSE after convergence is not small since we are dealing with an example where the signal to noise ratio is small. In Fig. 18 the output error for the experiment using the sign error algorithm is shown, as can be verified the output error tends to produce a signal with the same period of the sine after 300 iterations.

Signal Enhancement Simulations



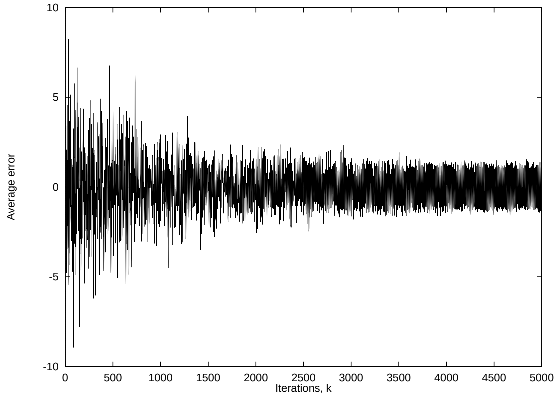
(a)

Signal Enhancement Simulations



(b)

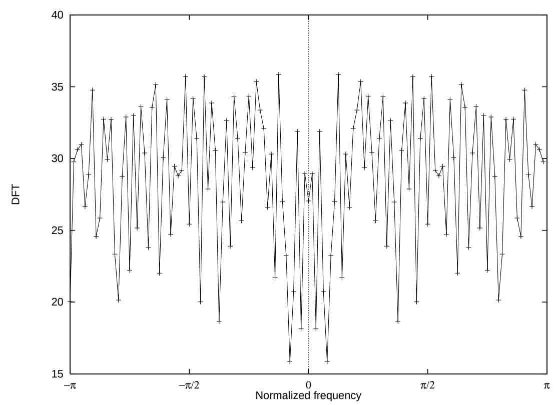
Signal Enhancement Simulations



(c)

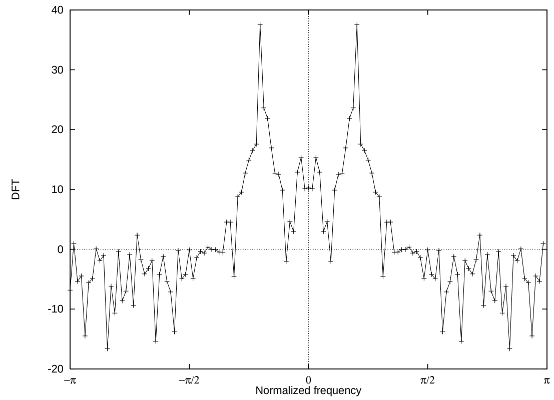
Figure 17: Learning curves for the (a) Sign-error, (b) Power-of-two, and (c) Normalized LMS algorithms.

Signal Enhancement Simulations



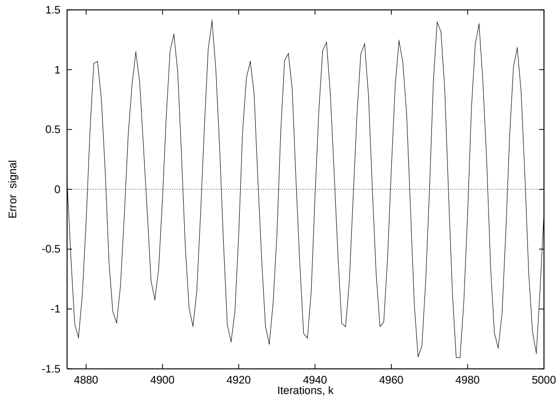
(a)

Signal Enhancement Simulations



(b)

Signal Enhancement Simulations



(c)

Figure 18: (a) DFT of the input signal, (b) DFT of the error signal, (c) The output error for the normalized LMS algorithm.

Signal Prediction Simulations

In a prediction problem the input signal is

$$x(k) = -\sqrt{2}\ sin 0.2\pi k + \sqrt{2}\ sin 0.05\pi k + n_x(k)$$

where $n_x(k)$ is a zero mean Gaussian white noise with variance $\sigma^2_{n_x} = 1$. The adaptive filter is a fourth-order FIR filter.

(a) Run an ensemble of 50 experiments, and plot the average learning curve.

(b) Determine the zeros of the resulting FIR filter and comment on the results.

Signal Prediction Simulations

LMS-Based Algorithms

We solved the problem above using the sign error, power-of-two error with $b_d = 12$, and normalized LMS algorithms

The values of μ for the sign error and power-of-two LMS algorithms were chosen 0.0028 and 0.0044, respectively. The coefficients of the adaptive filter were initialized with zero.

Signal Prediction Simulations

For the normalized LMS algorithm $\mu_n = 0.4$ and $\gamma = 10^{-6}$ were used. In all cases there is a strong attenuation of the predictor response around the frequencies of the two sinusoids. See for example the response depicted in Fig. 19 obtained by running the power-of-two LMS algorithm. The learning curves are depicted in Fig. 20. The zeros of the transfer function

Signal Prediction Simulations

Sign error:

$$-0.2515;\ -0.2915 \pm j0.3355;\ -0.6716 \pm j0.3355$$

Power-of-two:

$$-0.3939;\ -0.2351 \pm j0.3876;\ -0.6766 \pm j0.3422$$

Normalized:

$$-0.7739;\ 0.8044 \pm j0.1733;\ 0.8087 \pm j0.5713$$

Signal Prediction Simulations

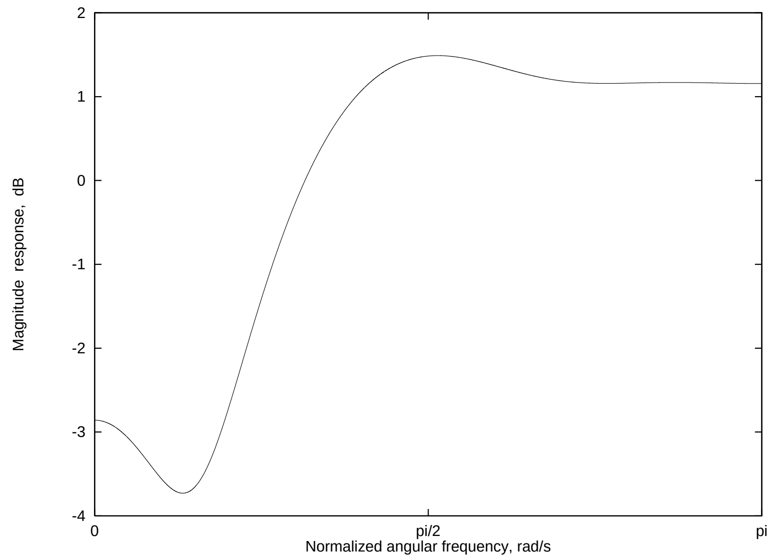
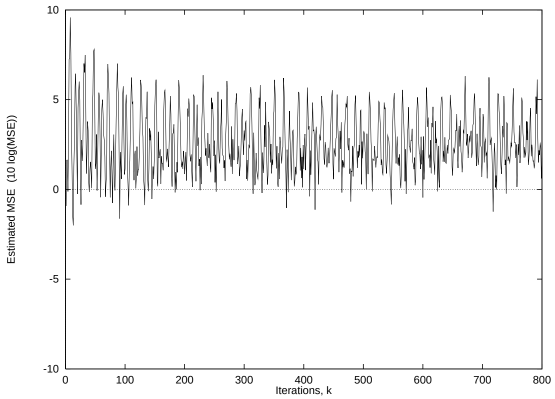


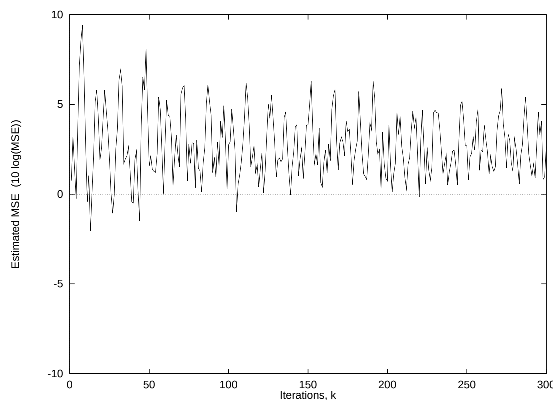
Figure 19: Magnitude response of the FIR adaptive filter at a given iteration after convergence using the power-of-two LMS algorithm.

Signal Prediction Simulations



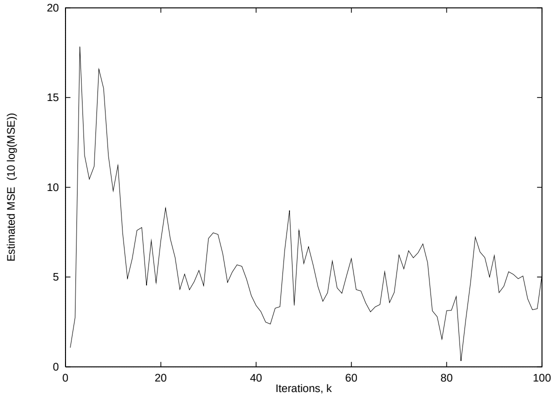
(a)

Signal Prediction Simulations



(b)

Signal Prediction Simulations



(c)

Figure 20: Learning curves for the (a) Sign-error, (b) Power-of-two, and (c) Normalized LMS algorithms.