2 Adaptive FIR filters

Some algorithms and their limitations

- Wiener filtering.
- Stationary case description (steepest descent, quasi-Newton).
- Traditional updating schemes: LMS, RLS, QR.
- Convergence in the mean and mean square error variance.
- Convergence speed - correlation matrix conditioning trade off.
- Different realizations.

In a general framework, the *Mean Squared Error* (MSE) $E\{e^2(n)\}$, has the following quadratic form:

$$E\{e^2(n)\} = \rho - 2\,\boldsymbol{\theta}^T \boldsymbol{p} + \boldsymbol{\theta}^T \boldsymbol{R_x} \boldsymbol{\theta}$$

where $\boldsymbol{R_x} > 0$ and $\boldsymbol{p}$ and ρ are assumed to be known in an ideal setting or, from a practical implementation point of view, some suitable estimates are at hand.

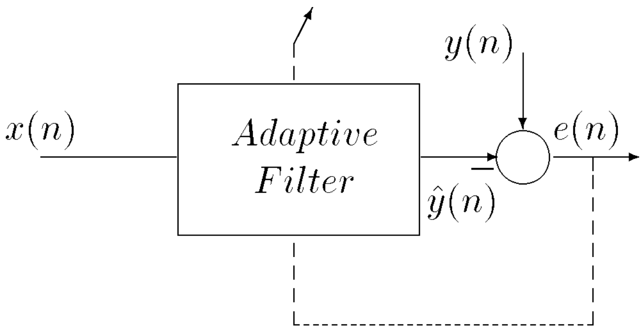


Figure 9: Adaptive filtering general framework.

2.1 Wiener Filtering

2.1.1 Optimal filtering

- **Problem 1: Inverse filtering**: To design $H(z)$, the input (observable signal) $x(n)$ has noise and the reference is not available. The idea is to design $H(z)$ so that $\hat{y}(n) = H(z)x(n)$ approximates $y(n)$.
- **Problem 2: Direct filtering or modeling**: $y(n)$ is the not observable output of the filter to design $H(z)$. The idea is to design $H(z)$ so that $\hat{y}(n) = H(z)x(n)$ approximates $y(n)$.

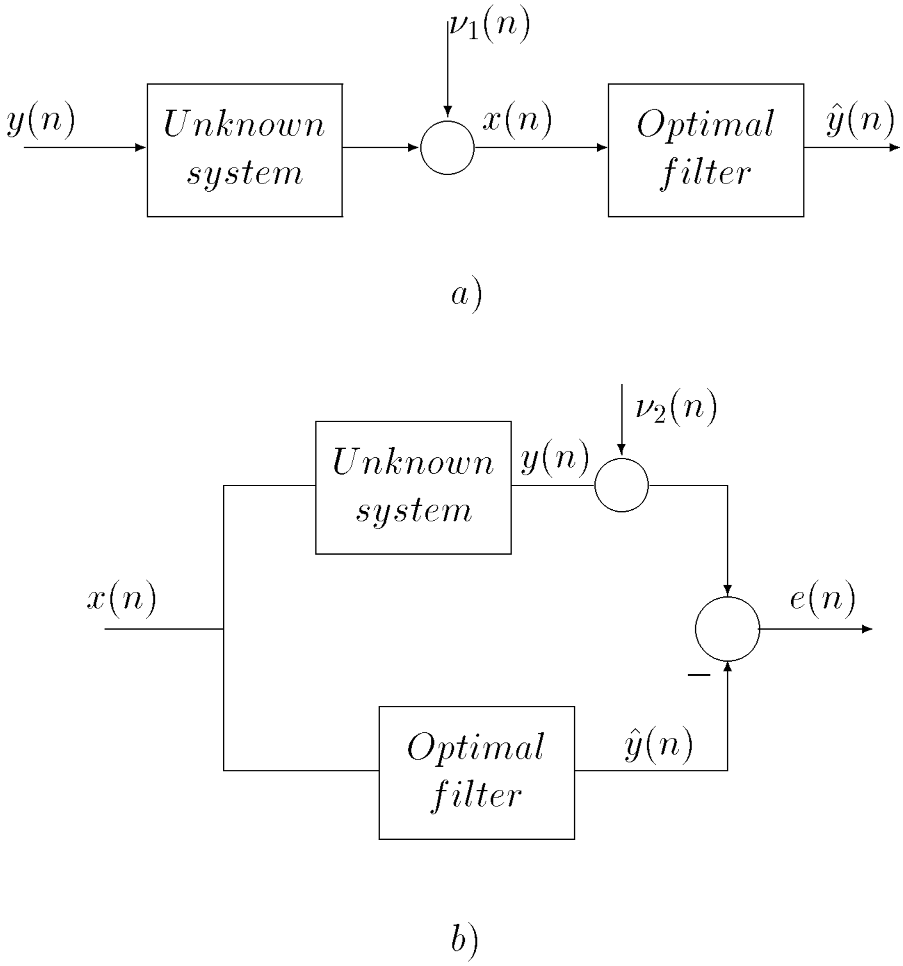


Figure 10: a)Inverse filtering and b) direct filtering or modeling

- Assumption: $x(n)$ and $y(n)$ jointly wide sense stationary and have zero mean and are uncorrelated with the disturbance.
- The degree of approximation is measured by the *Mean squared error*,

$$E\{e^2(n)\} = E\{(y(n) - \hat{y}(n))^2\} \tag{3}$$

- Second order statistics known, i.e.,

$$\begin{aligned} p(k) &= E\{y(n-k)x(n)\} \\ r(k) &= E\{x(n-k)x(n)\} \\ \rho &= E\{y^2(n)\} \end{aligned}$$

2.1.2 The inverse filtering problem

$x(n) = s(n) + \nu_1(n)$, $y(n) = s(n)$ ($s(n)$ recoverable signal). Since $\hat{y}(n) = \sum_k h(k)x(n-k)$, three cases:

- Non causal case: $x(n-k)$ known for all k, no constraints on $h(n)$, i.e., $\|h(n)\|^2 < \infty$.
- Causal FIR case: $x(n-k)$ known for $0 \le k \le n$, so $h(k) = 0$ for $k < 0$ and $k > n$.
- Causal IIR case: $x(n-k)$ known for $k \ge 0$, so $h(k) = 0$ for $k < 0$ and $\|h(n)\|^2 < \infty$.

$$
\begin{aligned}
E[e^2(n)] &= E[y^2(n)] - 2\sum_k h(k)E[y(n)x(n-k)] \\
&\quad + \sum_k\sum_l h(l)h(k)E[x(n-k)x(n-l)] \\
&= \rho - 2\sum_k h(k)p(k) + \sum_k\sum_l h(k)h(l)r(l-k)
\end{aligned}
$$

or

$$
\begin{aligned}
r_e(k) &= r_y(k) - r_{y\,\hat{y}}(k) - r_{\hat{y}\,y}(k) + r_{\hat{y}\,\hat{y}}(k) \\
&= \rho(k) - \sum_l [h(l)p(l+k) + p(l)h(l+k)] \\
&\quad + \sum_l\sum_j h(l)r(l+k-j)h(j) \\
S_e(e^{jw}) &= S_y(e^{jw}) - S_{yx}(e^{jw})H^*(e^{jw}) - S_{yx}^*(e^{jw})H(e^{jw}) + S_x(e^{jw})|H^*(e^{jw})|^2 \\
&= \left| H(e^{jw}) - \frac{S_{yx}(e^{jw})}{S_x(e^{jw})} \right|^2 S_x(e^{jw}) + \left[S_y(e^{jw}) - \frac{|S_{yx}(e^{jw})|^2}{S_x(e^{jw})} \right]
\end{aligned}
$$

- **the non causal case**: $S_{yx}(e^{jw}) = S_s(e^{jw})$ and $S_x(e^{jw}) = S_s(e^{jw}) + S_\nu(e^{jw})$, i.e.,

$$H(e^{jw}) = \frac{S_{y\nu}(e^{jw})}{S_x(e^{jw})} = \frac{S_s(e^{jw})}{S_s(e^{jw}) + S_\nu(e^{jw})}$$

- **the causal FIR case**:

$$\begin{aligned} E[e^2(n)] &= E[y^2(n)] - 2\sum_{k=0}^{N} h(k)E[y(n)x(n-k)] \\ &\quad + \sum_{k=0}^{N}\sum_{l=0}^{N} h(l)h(k)E[x(n-k)x(n-l)] \\ &= \rho - 2\sum_{k=0}^{N} h(k)p(k) + \sum_{k=0}^{N}\sum_{l=0}^{N} h(k)h(l)r(l-k) \end{aligned}$$

that is minimized for

$$\begin{aligned} p(k) &= \sum_{n=0}^{N} r(k-n)h(n), \quad or \\ 0 &= E[e(n)x(n-k)], \quad for\ \ 0 \le k \le N \end{aligned}$$

Let consider two cases:

- Filtering (basic equalization): if $x(n) = s(n) + \nu(n)$ and $y(n) = s(n-N)$, then $(r(n) = r_x(n) + r_\nu(n)$ and $p(n) = r_s(n-N))$:

$$\boldsymbol{R}_s + \boldsymbol{R}_\nu \begin{bmatrix} h(0) \\ \vdots \\ h(N) \\ \vdots \\ h(2N) \end{bmatrix} = \boldsymbol{R}_s \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

  whose solution is a linear phase FIR filter $(h(n) = h(2N-n)$, $n = 0, 1, ..., 2N$.

- Prediction:
  * Forward: if $\hat{y}(n) = \sum_{k=1}^{N} h_k x(n-k)$ and $y(n) = x(n)$, then:

$$\begin{bmatrix} r(1) & r(2) & \cdots & r(N) \\ r(2) & r(1) & \cdots & r(N-1) \\ & & \vdots & \\ r(N) & r(N-1) & \cdots & r(1) \end{bmatrix} \begin{bmatrix} h(1) \\ h(2) \\ \vdots \\ h(N) \end{bmatrix} = \begin{bmatrix} r(1) \\ r(2) \\ \vdots \\ r(N) \end{bmatrix}$$

  * Backward: if $\hat{y}(n-N) = \sum_{k=1}^{N} g_k x(n-k+1)$ and $y(n) = x(n-M)$, then:

$$\begin{bmatrix} r(1) & r(2) & \cdots & r(N) \\ r(2) & r(1) & \cdots & r(N-1) \\ & & \vdots & \\ r(N) & r(N-1) & \cdots & r(1) \end{bmatrix} \begin{bmatrix} g(1) \\ g(2) \\ \vdots \\ g(N) \end{bmatrix} = \begin{bmatrix} r(N) \\ r(N-1) \\ \vdots \\ r(1) \end{bmatrix}$$

    The solutions are related by:

$$g_k = h_{N-k} \quad for\, k = 1, ...N-1$$

- **the causal IIR case**: Here, in a similar form that for the FIR case, except for $N \to \infty$,

$$p(k) \;=\; \sum_{n=0}^{\infty} r(k-n)h(n),$$

is the optimal condition, for $0 \le k < \infty$. A frequency domain solution is obtained if

  - $x(n) = G(z)u(n)$, $u(n)$ white noise, i.e., $S_u(e^{jw}) = 1$ and $G(z)$ is invertible.
  - $y(n) = F(z)u(n) + \nu(n)$, $\nu(n)$ colored noise uncorrelated with $u(n)$, i.e., $S_{u\nu}(e^{jw}) = 0$.

then with $S_{yx}(e^{jw})$ and $S_y(e^{jw})$ respectively,

$$\begin{aligned}
F(e^{jw}) &= \frac{S_{yx}(e^{jw})}{G(e^{jw})^*} \\
S_\nu(e^{jw}) &= S_y(e^{jw}) - |F(e^{jw})|^2 = S_y(e^{jw}) - \frac{|S_{yx}(e^{jw})|^2}{S_x(e^{jw})} \\
S_e(e^{jw}) &= |F(e^{jw}) - G(e^{jw})H(e^{jw})|^2 + S_\nu(e^{jw})
\end{aligned}$$

and $H(z) = H_0(z)/G(z) = F_+(z)/G(z)$, where $F_+(z)$ is the causal part of F(z).

2.1.3 The direct filtering or modeling problem

Here $y(n) = H(z)x(n) + \nu_2(n)$, $\hat{y}(n) = \hat{H}(z)x(n)$.

- In the FIR case,

$$\begin{aligned}\hat{y}(n) &= b_0 x(n) + b_1 x(n-1) + ... + b_N x(n-N) \\ &= \boldsymbol{\theta}^T \boldsymbol{x}(n)\end{aligned}$$

where $\boldsymbol{\theta} = [b_0 ... b_N]^T$ and $\boldsymbol{x}(n) = [x(n)...x(n-N)]^T$.
Then in an stationary environment,

$$E\{e^2(n)\} = \rho - 2\,\boldsymbol{\theta}^T \boldsymbol{p} + \boldsymbol{\theta}^T \boldsymbol{R_x} \boldsymbol{\theta}$$

where $\boldsymbol{R_x} = E\{\boldsymbol{x}(n)\boldsymbol{x}^T(n)\}$ and $\boldsymbol{p} = E\{\boldsymbol{x}(n)y(n)\}$ are known.
A quadratic function of $\boldsymbol{\theta}(n)$ with

$$\begin{aligned}\boldsymbol{\nabla} = \frac{\partial E\{e^2(n)\}}{\partial \boldsymbol{\theta}} &= [\frac{\partial E\{e^2(n)\}}{\partial b_0}\ \frac{\partial E\{e^2(n)\}}{\partial b_1} ... \frac{\partial E\{e^2(n)\}}{\partial b_N}]^T \\ &= -2\,\boldsymbol{p} + 2\,\boldsymbol{R}_x \boldsymbol{\theta} = 0\end{aligned}$$

then the MSE is minimized when

$$\boldsymbol{\theta}_o = \boldsymbol{R_x}^{-1} \boldsymbol{p}$$

Note also that

$$\begin{aligned}\boldsymbol{\nabla} = \frac{\partial E\{e^2(n)\}}{\partial \boldsymbol{\theta}} &= E[2\,e(n)\,\frac{\partial e(n)}{\partial \boldsymbol{\theta}}] \\ &= -2E[e(n)\boldsymbol{x}(n)] = 0\end{aligned}$$

i.e., the *normal equation.*

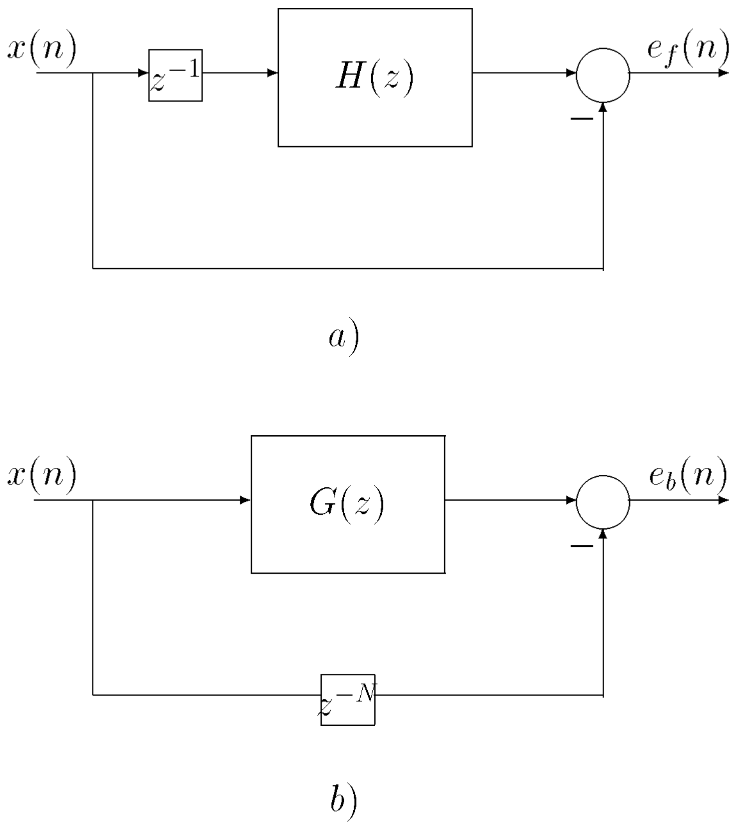


Figure 11: Relationship between prediction and whitening filtering. a) forward predictor and whitening filter, b) backward predictor and whitening filter.

- Whitening a forward prediction filter: with $e_f(n) = y(n) + \sum_{k=1}^{N} a_k y(n-k)$, find $A(z)$, constrained to be a monic FIR filter ($a(0) = 1$).

$$\begin{bmatrix} r(0) & r(1) & \cdots & r(N) \\ r(1) & r(0) & \cdots & r(N-1) \\ & & \vdots & \\ r(N) & r(N-1) & \cdots & r(0) \end{bmatrix} \begin{bmatrix} 1 \\ a(1) \\ \vdots \\ a(N) \end{bmatrix} = \begin{bmatrix} \alpha \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

  The N-order (forward) prediction filter and the $N+1$-order whitening filter are related by $A_{N+1}(z) = 1 - z^{-1} H_N(z)$.

- Whitening a backward prediction filter: with $e_b(n) = y(n-N) + \sum_{k=1}^{N} b_k y(n-k+1)$, find $B(z)$, constrained to be a monic FIR filter ($b(0) = 1$). Then

$$\begin{bmatrix} r(0) & r(1) & \cdots & r(N) \\ r(1) & r(0) & \cdots & r(N-1) \\ & & \vdots & \\ r(N) & r(N-1) & \cdots & r(0) \end{bmatrix} \begin{bmatrix} 1 \\ b(1) \\ \vdots \\ b(N) \end{bmatrix} = \begin{bmatrix} \beta \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

An important property: the outputs collection of whitening backward filters of orders 1 to N are orthogonal, i.e.,

$$E\{e_b^k e_b^m\} = \begin{cases} \beta_k & if\ m = k \\ 0 & if\ m \neq k \end{cases}$$

where $\beta_k = E\{(e_b^k)^2\}$.

This property can be used to obtain an useful decomposition (low Cholesky in this case) of the correlation matrix $\boldsymbol{R}$,

$$\begin{aligned}
\boldsymbol{D}_L &= diag[\beta_0, ..., \beta_N] \\
&= E\{\boldsymbol{e}_b^N \boldsymbol{e}_b^N\} = E\{\boldsymbol{L}\boldsymbol{x}\boldsymbol{x}^T \boldsymbol{L}^T\} \\
\boldsymbol{L} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ b_N(1) & 1 & \cdots & \\ & & \vdots & \\ b_N(N) & \cdots & b_1(1) & 1 \end{bmatrix} \\
\boldsymbol{R}^{-1} &= \boldsymbol{L}\boldsymbol{D}_L^{-1}\boldsymbol{L}^T
\end{aligned}$$

A similar factorization can be obtained but related to the whitening forward filtering, i.e., (upper Cholesky)

$$\begin{aligned}
\boldsymbol{R}^{-1} &= \boldsymbol{U}\boldsymbol{D}_U^{-1}\boldsymbol{U}^T \\
\boldsymbol{U} &= \begin{bmatrix} 1 & a_1(1) & \cdots & a_N(N) \\ 0 & 1 & \cdots & a_N(N-1) \\ & & \vdots & a_N(1) \\ 0 & 0 & \cdots & 1 \end{bmatrix} \\
\boldsymbol{D}_U &= diag[\alpha_0, ..., \alpha_N]
\end{aligned}$$

where can be shown that $\alpha_k = \beta_k$.

2.2 Optimization in the ideal setting

2.2.1 Newton algorithm

$$\begin{aligned}\boldsymbol{\theta}(n+1) &= \boldsymbol{\theta}(n) - \mu \boldsymbol{R}_x^{-1} \boldsymbol{\nabla}(n) \\ &= \boldsymbol{\theta}(n) + \mu \boldsymbol{R}_x^{-1}(-2\boldsymbol{p} + 2\boldsymbol{R}_x \boldsymbol{\theta}(n)) = (\boldsymbol{I} - 2\mu \boldsymbol{I})\boldsymbol{\theta}(n) + 2\mu \boldsymbol{\theta}_o\end{aligned}$$

if $\mu = 1/2$ the Wiener solution is reached in one step!.

2.2.2 Steepest Descent algorithm

Using $\boldsymbol{\nabla}(n) = 2(\boldsymbol{R}_x \boldsymbol{\theta}(n) - \boldsymbol{p})$, then

$$\begin{aligned}\boldsymbol{\theta}(n+1) &= \boldsymbol{\theta}(n) - \mu \boldsymbol{\nabla}(n) \\ &= \boldsymbol{\theta}(n) + 2\mu \boldsymbol{p} - 2\mu \boldsymbol{R}_x \boldsymbol{\theta}(n)\end{aligned}$$

With $\tilde{\boldsymbol{\theta}}(n) = \boldsymbol{\theta}(n) - \boldsymbol{\theta}_o$,

$$\begin{aligned}\tilde{\boldsymbol{\theta}}(n+1) &= (\boldsymbol{I} - 2\mu \boldsymbol{R}_x)\, \tilde{\boldsymbol{\theta}}(n) \\ &= (\boldsymbol{I} - 2\mu \boldsymbol{R}_x)^{n+1}\, \tilde{\boldsymbol{\theta}}(0)\end{aligned}$$

Since $\boldsymbol{R}_x > 0$, $\boldsymbol{R}_x = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T$, where $\boldsymbol{Q}$ is an orthogonal and $\boldsymbol{\Lambda}$ is the diagonal eigenvalue matrix. Then with $\boldsymbol{\vartheta}(n) = \boldsymbol{Q}^T \tilde{\boldsymbol{\theta}}(n)$,

$$\begin{aligned}E\left\{\tilde{\boldsymbol{\vartheta}}(n+1)\right\} &= [\boldsymbol{I} - \mu \boldsymbol{\Lambda}]\, E\left\{\tilde{\boldsymbol{\vartheta}}(n)\right\} \\ &= [\boldsymbol{I} - \mu \boldsymbol{\Lambda}]^{n+1} \left\{\tilde{\boldsymbol{\vartheta}}(0)\right\} \\ &= \begin{bmatrix} (1-2\mu\lambda_0)^{n+1} & 0 & \cdots & 0 \\ 0 & (1-2\mu\lambda_1)^{n+1} & & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & & (1-2\mu\lambda_N)^{n+1} \end{bmatrix} \tilde{\boldsymbol{\vartheta}}(0)\end{aligned}$$

Then, SD algorithm converges to the Wiener solution if, for $n \to \infty$, μ satisfy

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

where λ_{max} is the maximum eigenvalue of $\boldsymbol{R}_x$.

2.2.3 Conjugate direction algorithm

Consider $\boldsymbol{d}_j$, $j = 0, ..., N$, that verify

$$\boldsymbol{d}_j^T \boldsymbol{R}_x \boldsymbol{d}_k = 0 \quad if\ j \neq k$$

as *conjugate directions*. Then:

Theorem: The sequence

$$\boldsymbol{\theta}(n+1) = \boldsymbol{\theta}(n) + \gamma_n \boldsymbol{d}_n$$

where $\gamma_n = -(\boldsymbol{d}_n^T \boldsymbol{R}_x \boldsymbol{d}_n)^{-1} \boldsymbol{d}_n^T \boldsymbol{\nabla}(n)$ and $\boldsymbol{\nabla}(n) = 2(\boldsymbol{R}_x\ \boldsymbol{\theta}(n) - \boldsymbol{p})$ converges to $\boldsymbol{\theta}_o = \boldsymbol{R}_x^{-1}\boldsymbol{p}$ after $N+1$ steps, i.e., $\boldsymbol{\theta}(n) = \boldsymbol{\theta}_o$.

Assuming to minimize the MSE with $\boldsymbol{\theta}(n)$ constrained in $\boldsymbol{\theta}(0) + \boldsymbol{D}_N$, where $\boldsymbol{D}_N = [\boldsymbol{d}_0\, \boldsymbol{d}_1 \ldots \boldsymbol{d}_{N-1}]$. Then $(\boldsymbol{\theta}(n) - \boldsymbol{\theta}_o)$ is given by the $\boldsymbol{R}_x$-orthogonal projection of $(\boldsymbol{\theta}_o - \boldsymbol{\theta}(0))$ onto $\boldsymbol{D}_N$, i.e.,

$$\begin{aligned}
\boldsymbol{\theta}(n) &= \boldsymbol{\theta}(0) + \boldsymbol{D}_N(\boldsymbol{D}_N^T \boldsymbol{R}_x \boldsymbol{D}_N)^{-1} \boldsymbol{D}_N^T \boldsymbol{R}_x(\boldsymbol{\theta}_o - \boldsymbol{\theta}(0)) \\
&= \boldsymbol{\theta}(0) + \sum_{k=0}^{N-1} \boldsymbol{d}_k (\boldsymbol{d}_k^T \boldsymbol{R}_x \boldsymbol{d}_k)^{-1} \boldsymbol{d}_k^T \boldsymbol{R}_x(\boldsymbol{\theta}_o - \boldsymbol{\theta}(0))
\end{aligned}$$

But $\boldsymbol{d}_k^T \boldsymbol{R}_x \boldsymbol{\theta}_o = \boldsymbol{d}_k^T \boldsymbol{R}_x \boldsymbol{\theta}(k)$ so that with $\boldsymbol{p} = \boldsymbol{R}_x \boldsymbol{\theta}_o$ is possible to verify

$$\boldsymbol{d}_k^T \boldsymbol{R}_x (\boldsymbol{\theta}_o - \boldsymbol{\theta}_k) = -\boldsymbol{d}_k^T \boldsymbol{\nabla}(k)$$

that serves to justify the gain γ_k.

2.3 Updating algorithms

Two important properties related to estimation (updating) algorithms are in order:

- An estimate is *unbiased* if $E\{\boldsymbol{\theta}(n)\} = \boldsymbol{\theta}_o$.
- An estimate is consistent if $\boldsymbol{\theta}(n) \rightarrow \boldsymbol{\theta}_o$ as $n \rightarrow \infty$.

Since second order statistics are not usually available, some simplifications in the ideal method are necessary.

2.3.1 The Least-Mean-Square (LMS) algorithm

When the gradient $\boldsymbol{\nabla}$ is not available, a suitable estimate is $\nabla \approx -2e(n)\boldsymbol{x}(n)$, the LMS algorithm

$$\boldsymbol{\theta}(n+1) = \boldsymbol{\theta}(n) + \mu\boldsymbol{x}(n)e(n) \tag{4}$$

where $\mu > 0$. As can be expected by the analyisis of the ideal SD algorithm, this parameter is related to convergence speed and stability of the algorithm.

Some useful variants

$$\begin{aligned}
\boldsymbol{\theta}(n+1) &= \boldsymbol{\theta}(n) + \mu\, \boldsymbol{x}(n)\, sgn[e(n)] \quad \textit{Sign Error} \\
\boldsymbol{\theta}(n+1) &= \boldsymbol{\theta}(n) + \mu\, sgn[\boldsymbol{x}(n)]\, e(n) \quad \textit{Sign Data} \\
\boldsymbol{\theta}(n+1) &= \boldsymbol{\theta}(n) + \mu\, sgn[\boldsymbol{x}(n)]\, sgn[e(n)] \quad \textit{Sign Sign}
\end{aligned}$$

2.3.2 Convergence in the Mean and Error variance of the LMS

Using some simplificatory hypotesis, and by defining $\tilde{\boldsymbol{\theta}}(n) = \boldsymbol{\theta}(n) - \boldsymbol{\theta}_o$, and rewritten (4) as follows

$$\tilde{\boldsymbol{\theta}}(n+1) = \left[\boldsymbol{I} - \mu\boldsymbol{x}(n)\boldsymbol{x}^T(n)\right]\tilde{\boldsymbol{\theta}}(n) + \mu\boldsymbol{x}(n)(y(n) - \boldsymbol{x}^T(n)\boldsymbol{\theta}_o) \tag{5}$$

then

$$E\left\{\tilde{\boldsymbol{\theta}}(n+1)\right\} = E\left\{\left[\boldsymbol{I} - \mu\boldsymbol{x}(n)\boldsymbol{x}^T(n)\right]\tilde{\boldsymbol{\theta}}(n)\right\} + \mu E\left\{\boldsymbol{x}(n)(y(n) - \boldsymbol{x}^T(n)\boldsymbol{\theta}_o)\right\} \tag{6}$$

Using the hypotesis

$$E\left\{\tilde{\boldsymbol{\theta}}(n+1)\right\} = \left[\boldsymbol{I} - \mu\boldsymbol{R}_x\right]E\left\{\tilde{\boldsymbol{\theta}}(n)\right\} \tag{7}$$

Using $\boldsymbol{R}_x = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T$ (Cholesky) and pre-multiplying (7) by $\boldsymbol{Q}^T$ and defining

$$\boldsymbol{\vartheta}(n) = \boldsymbol{Q}^T\tilde{\boldsymbol{\theta}}(n) \tag{8}$$

is possible to obtain

$$\begin{aligned} E\left\{\tilde{\boldsymbol{\vartheta}}(n+1)\right\} &= \left[\boldsymbol{I} - \mu\boldsymbol{\Lambda}\right]E\left\{\tilde{\boldsymbol{\vartheta}}(n)\right\} \\ &= \left[\boldsymbol{I} - \mu\boldsymbol{\Lambda}\right]^{n+1}E\left\{\tilde{\boldsymbol{\vartheta}}(0)\right\} \end{aligned} \tag{9}$$

Then, in order that $\boldsymbol{\theta}(n)$ converge in the mean to the Wiener solution

$$0 < \mu < \frac{1}{\lambda_{max}} \tag{10}$$

Since the gradient is noisy, some residual MSE after convergence can be expected. This residual error is called **Excess in the MSE** and is defined at time n by

$$
\begin{aligned}
\Delta\xi(n) = \xi(n) - \xi_{min} &= E\{\tilde{\boldsymbol{\theta}}^T(n)\boldsymbol{R}_x\tilde{\boldsymbol{\theta}}(n)\} \\
&= E\{tr(\boldsymbol{R}_x\tilde{\boldsymbol{\theta}}(n))\tilde{\boldsymbol{\theta}}^T(n)\} \\
&= tr\left(E\{\boldsymbol{R}_x\tilde{\boldsymbol{\theta}}(n)\tilde{\boldsymbol{\theta}}^T(n)\}\right)
\end{aligned}
$$

where $tr(\boldsymbol{AB}) = tr(\boldsymbol{BA})$ was used.

Using this, and after some not trivial intermediate steps, it is possible to shown that

$$
\begin{aligned}
\Delta\xi(n) &\cong \frac{\mu\sigma_\nu^2 \sum_{k=0}^{N}\lambda_k}{1-\mu\sum_{k=0}^{N}\lambda_k} \\
&= \frac{\mu\sigma_\nu^2 tr[\boldsymbol{R}]}{1-\mu tr[\boldsymbol{R}]}
\end{aligned}
$$

where $\sigma_\nu^2 = E\{\nu^2(n)\}$. Finally, for $n \to \infty$

$$
\xi_{exc} = \lim_{n\to\infty}\Delta\xi(n) \cong \frac{\mu\sigma_\nu^2 tr[\boldsymbol{R}]}{1-\mu tr[\boldsymbol{R}]}
$$

and assuming μ small enough,

$$
\xi_{exc} \cong \mu\sigma_n^2 tr[\boldsymbol{R}] = \mu(N+1)\sigma_\nu^2\sigma_x^2
$$

Note that ξ_{exc} is a relative quantity. In order to compare different algorithms a more suitable parameter is the **Misadjustment**:

$$
M \triangleq \frac{\xi_{exc}}{\xi_{min}} = \frac{\mu tr[\boldsymbol{R}]}{1-\mu tr[\boldsymbol{R}]}
$$

2.3.3 MSE transient

The essential drawback related to the LMS algorithm is that convergence speed depends directly on the correlation matrix eigenvalue spread.

Using the expression of the MSE at time n it is not hard to show that

$$\begin{aligned}\xi(n) &= \xi_{min} + E\{\tilde{\boldsymbol{\vartheta}}^T(n)\boldsymbol{\Lambda}\tilde{\boldsymbol{\vartheta}}(n)\}\\ &= \xi_{min} + \sum_{k=0}^{N} \lambda_k \tilde{\vartheta}_k^2(n)\\ &= \xi_{min} + \sum_{k=0}^{N} \lambda_k (1-\mu\lambda_k)^{2n} \tilde{\vartheta}_k^2(0)\end{aligned}$$

Then the transient that characterizes the behavior of the MSE convergence is related to $N+1$ geometric ratios, $r_k = 1-2\mu\lambda_k$. Using the usual exponential envelope $r_k = 1 - 2\mu\lambda_k \cong 1 - \frac{1}{\tau_k}$, then

$$\tau_k \cong \frac{1}{2\mu\lambda_k}$$

with $k = 0, ..., N$. This is the time constants related to parameter convergence. For MSE convergence speed

$$\tau_{xi_k} \cong \frac{1}{4\mu\lambda_k}$$

2.3.4 The Normalized LMS algorithm

- To optimize the convergence speed: a time variant convergence factor $\mu(n)$ in the LMS algorithm.
- Consider the difference between the instantaneous squared error $e^2(n)$ and the squared error obtained by $\boldsymbol{\theta}_*(n) = \boldsymbol{\theta}(n)+\boldsymbol{\Delta\theta}(n)$, given by $e_*^2(n)$.
- Then

$$\begin{aligned} \Delta e^2(n) &= e_*^2(n) - e^2(n) \\ &= -2\boldsymbol{\Delta\theta}^T(n)\boldsymbol{x}(n)e(n) \\ &\quad +\boldsymbol{\Delta\theta}^T(n)\boldsymbol{x}(n)\boldsymbol{x}^T(n)\boldsymbol{\Delta\theta}(n) \end{aligned}$$

- Using the $\boldsymbol{\Delta\theta}(n)$ obtained from the LMS algorithm and by minimization of the previous equation with respect to $\mu(n)$,

$$\mu(n) = \left(\frac{1}{2\boldsymbol{x}^T(n)\boldsymbol{x}(n)}\right)$$

2.3.5 The Transform-Domain LMS Algorithm

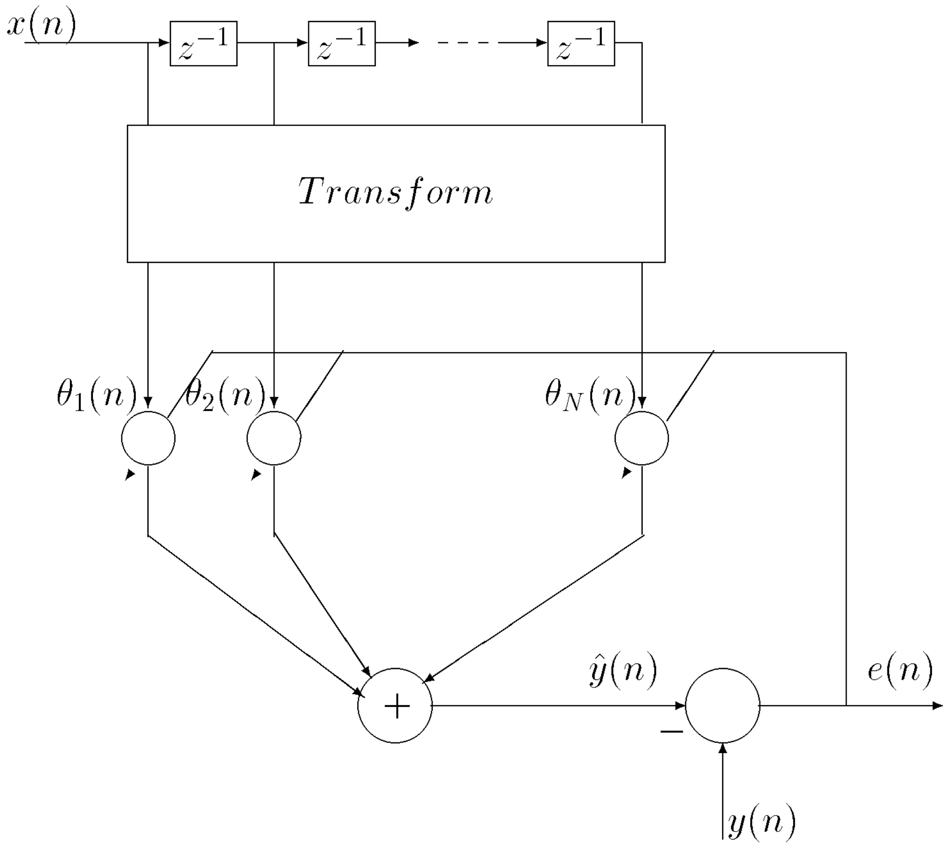


Figure 12: The transform domain adaptive filter

- Convergence speed is related to different principal axes length of the **MSE surface** countours.
- If these countours are circular the optimum situation is at hand. This can be achieved only if the eigenvectors of the $\boldsymbol{R}_x$ matrix are known.
- The MSE surface is changed by a coordinate transform, $\hat{\boldsymbol{x}}(n) = \boldsymbol{T}\boldsymbol{x}(n)$ where $\boldsymbol{T}\boldsymbol{T}^T = \boldsymbol{I}$, or

$$\xi \;=\; E\{e^2(n)\} = \xi_{min} + \tilde{\boldsymbol{\theta}}^T \, E\{\hat{\boldsymbol{x}}(n)\hat{\boldsymbol{x}}^T(n)\} \, \tilde{\boldsymbol{\theta}}$$

where $\tilde{\boldsymbol{\theta}} = \hat{\boldsymbol{\theta}}(n) - \hat{\boldsymbol{\theta}}_o$. Then

$$\xi - \xi_{min} \;=\; \tilde{\boldsymbol{\theta}}^T \, \boldsymbol{T}\boldsymbol{R}_x\boldsymbol{T}^T \, \tilde{\boldsymbol{\theta}}$$

that represent a **rotation** of the parameter space related to the direct form FIR filter.

- The intersection of the different MSE contour with the i-th space parameter coordenates is $\xi - \xi_{min} = \left[\boldsymbol{T}\boldsymbol{R}_x\boldsymbol{T}^T\right]_{ii} \tilde{\theta}_i$.
- For an hypersphere it is necessary that $|\tilde{\theta}_i| = |\tilde{\theta}_j|$ for all (i, j).
- This conditions can be achieved, at least approximately, using an scaling factor

$$\left[\boldsymbol{T}\boldsymbol{R}_x\boldsymbol{T}^T\right]_{ii} \cong E\{\hat{x}_i^2(n)\} = \hat{\sigma}_i^2$$

- The updating equation of the Transform Domain LMS algorithm is the following

$$\hat{\boldsymbol{\theta}}(n+1) = \hat{\boldsymbol{\theta}}(n) + \mu\boldsymbol{\Lambda}^{-1}\hat{\boldsymbol{x}}(n)e(n) = \hat{\boldsymbol{\theta}}(n) + \mu\boldsymbol{\Lambda}^{-1}\boldsymbol{T}\boldsymbol{x}(n)e(n)$$

  where $\boldsymbol{\Lambda} = diag\,[\hat{\sigma}_1^2, ..., \hat{\sigma}_N^2]$ and $\hat{\sigma}_i^2(n+1) = (1-\mu_\sigma)\hat{\sigma}_i^2(n) + \mu_\sigma \hat{x}_i^2(n)$, with μ_σ a small constant.
- Two suitable transform for this algorithm are the *Discrete Fourier Transform* (complex) and the *Discrete Cosine Transform* (real), given by

$$\hat{x}_i(n) \;=\; \sqrt{\frac{2}{N+1}} \sum_{k=0}^{N} x(k-n) \cos\left(\pi i \frac{(2k+1)}{2(N+1)}\right)$$

2.3.6 The Quasi-Newton algorithm

- Higher complexity than the LMS but with fast (initial) convergence speed using an estimate of $\boldsymbol{R}_x^{-1}$.
- A possible algorithm is the following

$$\boldsymbol{\theta}(n+1) = \boldsymbol{\theta}(n) + \mu \boldsymbol{P}(n+1)\boldsymbol{x}(n)e(n) \tag{11}$$

where

$$\boldsymbol{P}(n+1) = \left(\frac{1}{1-\mu}\right)\left(\boldsymbol{P}(n) - \frac{\boldsymbol{P}(n)\boldsymbol{x}(n)\boldsymbol{x}^T(n)\boldsymbol{P}(n)}{\frac{1-\mu}{\mu} + \boldsymbol{x}^T(n)\boldsymbol{P}(n)\boldsymbol{x}(n)}\right) \tag{12}$$

- $\boldsymbol{P}(n+1)$ represents an estimate of $\boldsymbol{R}_x^{-1}$ at time $n+1$, in this case using the *matrix inversion lemma*. This algorithm is called *Quasi-Newton*.

2.4 Other algorithms

2.4.1 The RLS algorithm

Assuming a linear regressor model:

$$y(n) \;=\; \sum_{k=1}^{N} \theta_k^o x(n-k) + \nu(n)$$

The RLS algorithm estimates the $\boldsymbol{\theta}_o$ parameters by minimizing

$$V_N(\theta) \;=\; \frac{1}{N}\sum_{n=1}^{N} e^2(n)$$

where $e(n) = y(n) - \boldsymbol{\theta}^T(n)\boldsymbol{x}_N(n)$. The well known recursive solution of this problem is

$$\boldsymbol{\theta}(n) \;=\; \boldsymbol{\theta}(n-1) + \boldsymbol{\kappa}_N(n)\left(y(n) - \boldsymbol{\theta}^T(n-1)\boldsymbol{x}_N(n)\right)$$

where

$$\begin{aligned}
\boldsymbol{\kappa}_N(n) &= \boldsymbol{R}_{N-1}^{-1}(n)\boldsymbol{x}_N(n) \qquad n \geq N \\
\boldsymbol{R}_{N-1}(n) &= \boldsymbol{R}_{N-1}(n-1) + \boldsymbol{x}_N(n)\boldsymbol{x}_N^T(n)
\end{aligned}$$

2.4.2 The fast RLS algorithm

- The fast RLS will be derived by close relationship with the *conjugate direction algorithm* and the forward and backward prediction filters.
- The choice of two particular conjugate directions is essential for the present derivation of fast RLS algorithm. These conjugate directions are related to the **forward and backward prediction filter coefficients** as discussed below.
- The fast RLS algorithm is related to the Kalman gain updating (in time) $\boldsymbol{\kappa}_N(n-1) \rightarrow \boldsymbol{\kappa}_N(n)$.
- This updating can be seen as composed of time update and order update.
  1. $\boldsymbol{\kappa}_N(n-1) \rightarrow \boldsymbol{\kappa}_{N+1}(n)$,
  2. $\boldsymbol{\kappa}_{N+1}(n) \rightarrow \boldsymbol{\kappa}_N(n)$.
- Due to the shifted structure of the regressor $\boldsymbol{x}_N(n)$,

$$\boldsymbol{x}_{N+1}(n) = \begin{bmatrix} x(n) \\ \boldsymbol{x}_N(n-1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_N(n) \\ x(n-N) \end{bmatrix}$$

  where $\boldsymbol{x}_{N+1,1:N}(n) = \boldsymbol{x}_N(n-1)$ and $\boldsymbol{x}_{N+1,0:N-1}(n) = \boldsymbol{x}_N(n)$.
- Then, assuming that $\boldsymbol{x}_N(n) = 0$ for $n \leq 0$,

$$\boldsymbol{R}_{N-1}(n-1) = \boldsymbol{R}_{1:N}(n) \qquad \boldsymbol{R}_{N-1}(n) = \boldsymbol{R}_{0:N-1}(n)$$

  where $\boldsymbol{R}_{1:N}(n)$ and $\boldsymbol{R}_{0:N-1}(n)$ are the lower right and upper left corner of $\boldsymbol{R}_N(n)$, respectively.

- Kalman gain at times $n-1$ and n can be written

$$\begin{aligned}
\boldsymbol{\kappa}_N(n-1) &= \boldsymbol{R}_{1:N}^{-1}(n-1)\boldsymbol{x}_{N+1,1:N}(n) \\
\boldsymbol{\kappa}_N(n) &= \boldsymbol{R}_{0:N-1}^{-1}(n)\boldsymbol{x}_{N+1,0:N-1}(n) \\
\boldsymbol{\kappa}_{N+1}(n) &= \boldsymbol{R}_N^{-1}(n)\boldsymbol{x}_{N+1}(n)
\end{aligned}$$

- Following the first step (time update) above, the problem can be stated has: given $\boldsymbol{\kappa}_N(n-1)$ and $\boldsymbol{\kappa}_N(n)$, find $\boldsymbol{\kappa}_{N+1}(n)$ as the solution to the $N+1$-dimensional problem

$$\overset{Minimize}{\boldsymbol{z} = \boldsymbol{\kappa}_{N+1}(n)} \quad \left(\frac{1}{2}\boldsymbol{z}^T\boldsymbol{R}_N(n)\boldsymbol{z} - \boldsymbol{x}_{N+1}^T(n)\boldsymbol{z}\right)$$

- This can be achieved with a conjugate direction algorithm with

$$\begin{aligned}
\boldsymbol{d}_N &= [1 \;\; \boldsymbol{a}_N^T(n)]^T \\
\boldsymbol{d}_N^T\boldsymbol{\nabla}(n) &= [1 \;\; \boldsymbol{a}_N^T(n)]\boldsymbol{x}_{N+1}(n) = e_N^f(n) \\
\boldsymbol{d}_N^T\boldsymbol{R}_N(n)\boldsymbol{d}_N &= \xi_N^f(n)
\end{aligned}$$

  where $\boldsymbol{a}_N(n)$ are the coefficients of the forward prediction filter, $\xi_N^f(n)$ is an estimate of the *least square forward prediction error* and $e_N^f(n)$ is the *aposteriori forward prediction error*.

- Then

$$\boldsymbol{\kappa}_{N+1}(n) = \begin{bmatrix} 0 \\ \boldsymbol{\kappa}_N(n-1) \end{bmatrix} + \begin{bmatrix} 1 \\ \boldsymbol{a}_N(n) \end{bmatrix} \left(\xi_N^f(n)\right)^{-1} e_N^f(n)$$

- For the second step (order update), i.e., $\boldsymbol{\kappa}_{N+1}(n) \to \boldsymbol{\kappa}_N(n)$, the problem can be stated has: given $\boldsymbol{\kappa}_N(n-1)$ and $\boldsymbol{\kappa}_N(n)$, find $\boldsymbol{\kappa}_{N+1}(n)$ as the solution to the $N+1$-dimensional problem

$$\overset{Minimize}{\boldsymbol{z} = \boldsymbol{\kappa}_{N+1}(n)} \quad \left(\frac{1}{2}\boldsymbol{z}^T \boldsymbol{R}_N(n)\boldsymbol{z} - \boldsymbol{x}_{N+1}^T(n)\boldsymbol{z}\right)$$

- This can be achieved using a conjugate direction algorithm with

$$\begin{aligned}
\boldsymbol{d}_N &= [\boldsymbol{b}_N^T(n)\ \ 1]^T \\
\boldsymbol{d}_N^T \boldsymbol{\nabla}(n) &= [\boldsymbol{b}_N^T(n)\ \ 1]\boldsymbol{x}_{N+1}(n) = e_N^b(n) \\
\boldsymbol{d}_N^T \boldsymbol{R}_N(n)\boldsymbol{d}_N &= \xi_N^b(n)
\end{aligned}$$

where $\boldsymbol{b}_N(n)$ are the coefficients of the backward prediction filter, $\xi_N^b(n)$ is an estimate of the *least square backward prediction error* and $e_N^b(n)$ is the *aposteriori backward prediction error.*

- Using this results,

$$\boldsymbol{\kappa}_{N+1}(n) = \begin{bmatrix} \boldsymbol{\kappa}_N(n) \\ 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{b}_N(n) \\ 1 \end{bmatrix} \left(\xi_N^b(n)\right)^{-1} e_N^b(n)$$

- Since the required solution is the Kalman time update, $\boldsymbol{\kappa}_N(n)$ is obtained as a function of $\boldsymbol{\kappa}_{N+1}(n)$ from the previous equation.

- The complete fast RLS algorithm requires 2 CD algorithms for the time update of the Kalman gain and 2 CD algorithms to obtain:
  - a) the time update of the prediction filter coefficients and
  - b) the parameter updates $\boldsymbol{\theta}(n)$.

2.4.3 QR decomposition based RLS algorithm

- If the standard RLS algorithm

$$\boldsymbol{\theta}(n+1) \;=\; \boldsymbol{\theta}(n) + \left[\sum_{k=0}^{n} \lambda^{n-k}\boldsymbol{x}(k)\boldsymbol{x}^T(k)\right]^{-1} \boldsymbol{x}(n)e(n)$$

  where $e(n) = y(n) - \boldsymbol{\theta}^T(n)\boldsymbol{x}(n)$ ($\boldsymbol{x}(n) = \boldsymbol{x}_N(n)$) and $0 << \lambda \leq 1$ is the forgetting factor, is rewritten as

$$\begin{bmatrix} e(n) \\ 0 \\ \vdots \\ 0 \end{bmatrix} - \begin{bmatrix} \boldsymbol{x}^T(n) \\ \lambda^{1/2}\boldsymbol{x}^T(n-1) \\ \vdots \\ \lambda^{1/2}\boldsymbol{x}^T(0) \end{bmatrix} \tilde{\boldsymbol{\theta}}(n) \tag{13}$$

  where $\tilde{\boldsymbol{\theta}}(n) = \boldsymbol{\theta}(n+1) - \boldsymbol{\theta}(n)$.

- Then $e(n)\boldsymbol{u}_1 - \boldsymbol{X}(n)\tilde{\boldsymbol{\theta}}(n)$, where $\boldsymbol{u}_1$ is the unit vector with a "1" in the first position, and

$$\boldsymbol{X}(n) \;=\; \begin{bmatrix} \boldsymbol{x}^T(n) \\ \lambda^{1/2}\boldsymbol{X}(n-1) \end{bmatrix}$$

- If an $n \times n$ (with $n \geq N+1$) orthogonal matrix $\boldsymbol{Q}(n-1)$ is known at time $n-1$ such that

$$\boldsymbol{Q}(n-1)\boldsymbol{X}(n-1) \;=\; \begin{bmatrix} \bigcirc \\ \boldsymbol{R}(n-1) \end{bmatrix}$$

  where $\boldsymbol{R}(n-1)$ is an upper triangular matrix of dimension $N \times N$ (dimension of $\boldsymbol{\theta}(n)$).

- Then

$$\begin{bmatrix} 1 & \\ & \boldsymbol{Q}(n-1) \end{bmatrix} \big(e(n)\boldsymbol{u}_1 - \boldsymbol{X}(n)\tilde{\boldsymbol{\theta}}(n)\big) = \begin{bmatrix} e(n) \\ 0 \\ \vdots \\ 0 \end{bmatrix} - \begin{bmatrix} \boldsymbol{x}^T(n) \\ \bigcirc \\ \lambda^{1/2}\boldsymbol{R}(n-1) \end{bmatrix}$$

- If $\boldsymbol{R}(n-1)$ is known, the triangularization at time n can be completed by introducing zeros into the locations occupied by the most recent vector $\boldsymbol{x}(n)$.

- This is achieved by an $n \times n$ orthogonal matrix $\hat{\boldsymbol{Q}}(n)$

$$\hat{\boldsymbol{Q}}(n) \begin{bmatrix} e(n) \\ 0 \\ \vdots \\ 0 \end{bmatrix} - \hat{\boldsymbol{Q}}(n) \begin{bmatrix} \boldsymbol{x}^T(n) \\ \bigcirc \\ \lambda^{1/2}\boldsymbol{R}(n-1) \end{bmatrix} \tilde{\boldsymbol{\theta}}(n) = e(n)\hat{\boldsymbol{q}}_1(n) - \begin{bmatrix} \bigcirc \\ \boldsymbol{R}(n) \end{bmatrix} \tilde{\boldsymbol{\theta}}(n) \tag{14}$$

where $\hat{\boldsymbol{q}}_1(n)$ is the first column of $\hat{\boldsymbol{Q}}(n)$.

- (14) can be performed using *Givens rotations*, such that

$$\hat{\boldsymbol{Q}}(n) = \hat{\boldsymbol{Q}}_N \ldots \hat{\boldsymbol{Q}}_1$$

with

$$\hat{\boldsymbol{Q}}_k = \begin{bmatrix} \cos\varphi_k & & -\sin\varphi_k & \\ & \boldsymbol{I}_{n+k-N-1} & & \\ \sin\varphi_k & & \cos\varphi_k & \\ & & & \boldsymbol{I}_{N-k} \end{bmatrix}$$

- The proper selection of the rotation angles $\{\varphi_k\}$ will annihilate the elements of $\boldsymbol{x}^T(n)$ appearing in (14).

- The term $\hat{\boldsymbol{q}}_1(n)$ in (14) in closed form is

$$\hat{\boldsymbol{q}}_1(n) = \begin{bmatrix} \prod_{k=1}^{N} \cos\varphi_k \\ \mathbf{0} \\ \boldsymbol{g} \end{bmatrix}$$

where $\boldsymbol{g} = [g_1 ..., g_N]^T$, $g_k = \sin\varphi_k \prod_{i=1}^{k-1} \cos\varphi_k$.

- The parameter update $\tilde{\boldsymbol{\theta}}(n)$ is then solved from

$$e(n)\boldsymbol{g} = \boldsymbol{R}(n)\tilde{\boldsymbol{\theta}}(n)$$

using back substitution.

- An useful scaled algorithm can be obtained considering the QR decomposition of $\boldsymbol{X}(n)$

$$\boldsymbol{Q}(n)\boldsymbol{X}(n) = \begin{bmatrix} \bigcirc \\ \boldsymbol{R}(n) \end{bmatrix}$$

- Because $\boldsymbol{Q}(n)$ is orthogonal, we have

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

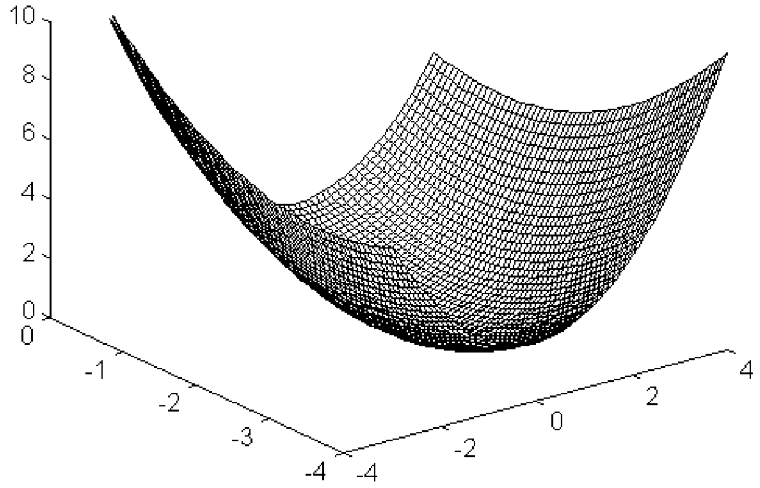
- If $\boldsymbol{x}(n)$ is stationary,

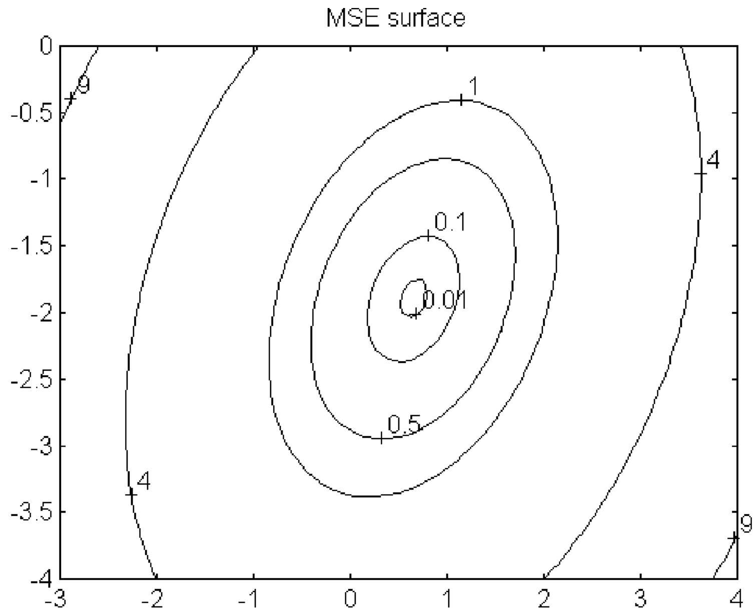
$$\lim_{n \to \infty} E\{\boldsymbol{R}^T(n)\boldsymbol{R}(n)\} = \frac{E\{\boldsymbol{x}(n)\boldsymbol{x}^T(n)\}}{1-\lambda}$$

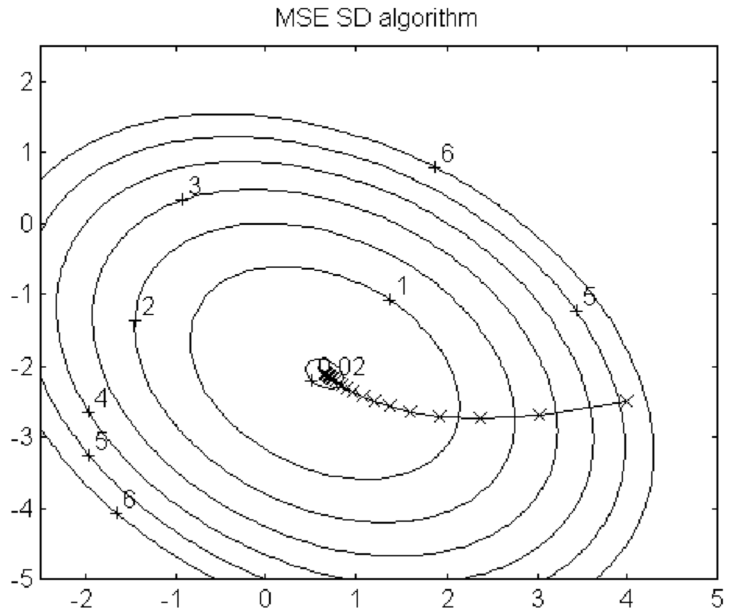
- Then for $\lambda \to 1$, the elements of $\boldsymbol{R}(n)$ can become large.
- Overflow in $\boldsymbol{R}(n)$ can be avoided considering in (13) that if $\{\boldsymbol{x}(k)\}_{k=0}^{n}$ and $e(n)$ are similarly scaled, the LS is left unchanged.
- From (15) an appropiate choice is $\sqrt{1-\lambda}$.

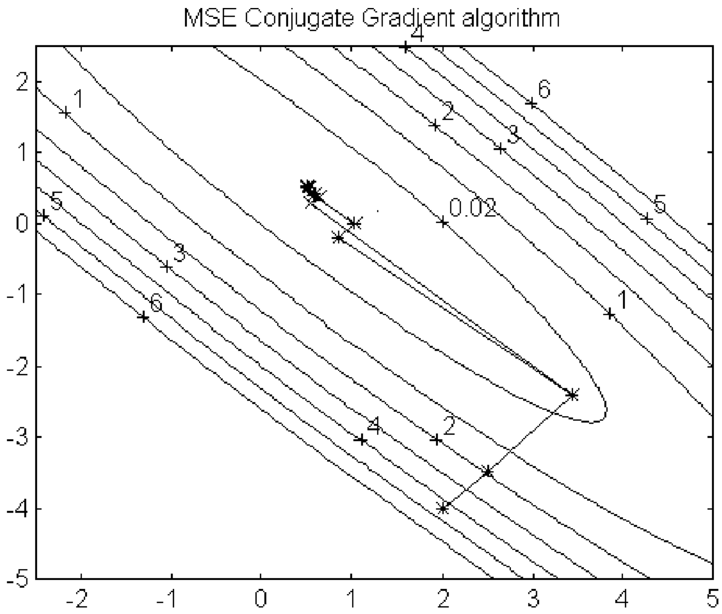
Examples

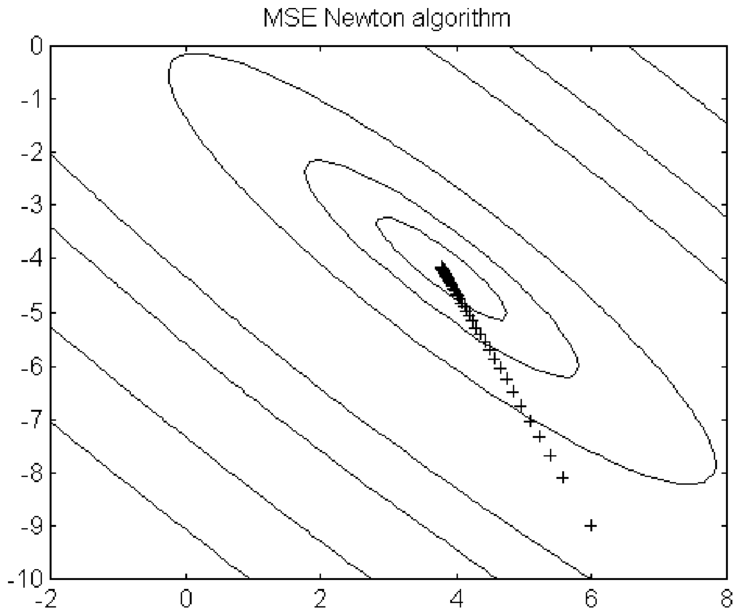
- An (adaptive) signal-cancelling application, with two taps!
- $x(n) = \sin w_0 n + \nu(n)$ with $E\{\nu^2(n)\} = r$,
- $y(n) = 2\cos w_0 n$,
- $\hat{y}(n) = \theta_0 x(n) + \theta_1 x(n-1)$,
- $e(n) = y(n) - \hat{y}(n)$
- $\boldsymbol{R}_x = \frac{1}{2}\begin{bmatrix} 1+2\,r & \cos w_0 \\ \cos w_0 & 1+2\,r \end{bmatrix}$ and $\boldsymbol{p} = 2\begin{bmatrix} 0 \\ -\sin w_0 \end{bmatrix}$.
- $\boldsymbol{\theta}^* = \boldsymbol{R}^{-1}\boldsymbol{p}$.
- $E\{e^2(n)\} = (\frac{1}{2} + r)(\theta_0^2 + \theta_1^2) + \theta_0\theta_1 \cos w_0 + 2\theta_1 \sin w_0 + 2$

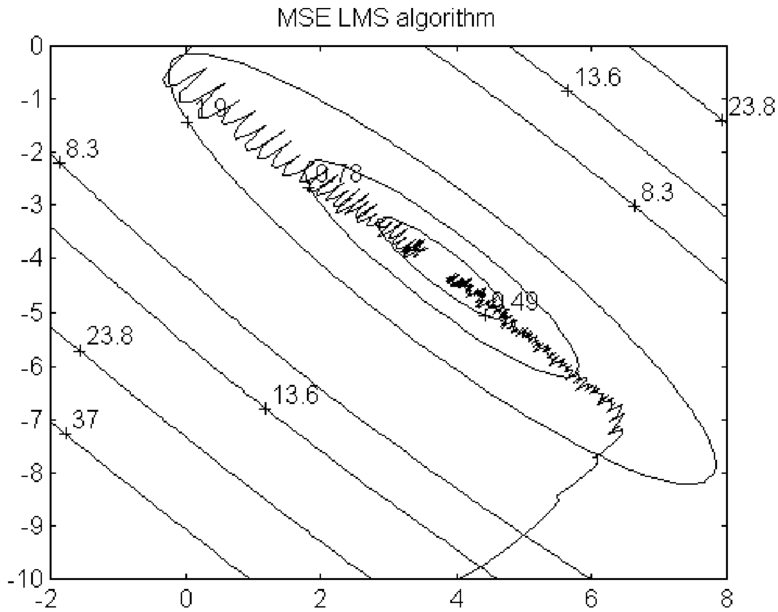

10
8
6
4
2
0
0
-1
-2
-3
-4
-4
-2
0
2
4

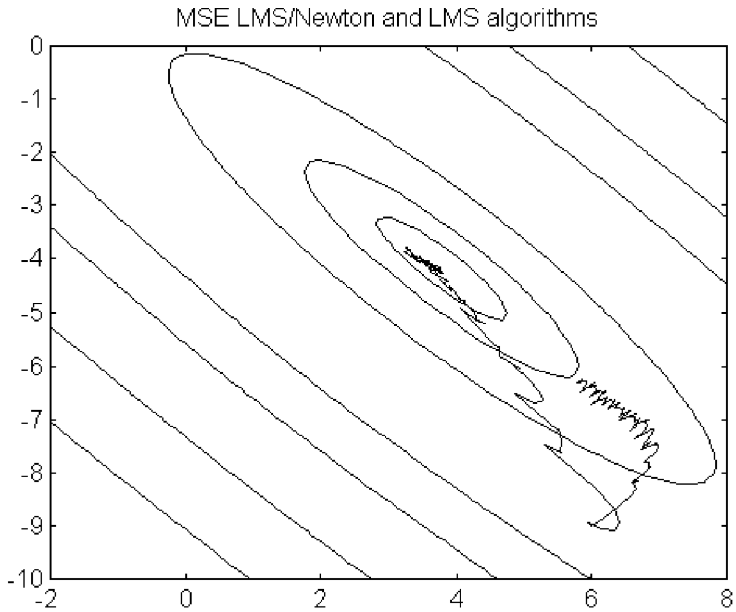

MSE surface
0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
-4
-3
-2
-1
0
1
2
3
4
9
1
4
0.1
0.01
0.5
4
9


MSE SD algorithm
6
3
1
5
2
0.02
4
5
6
2
1
0
-1
-2
-3
-4
-5
-2
-1
0
1
2
3
4
5


MSE Conjugate Gradient algorithm
4
6
2
3
1
5
0.02
5
3
1
6
4
2
2
1
0
-1
-2
-3
-4
-5
-2
-1
0
1
2
3
4
5


MSE Newton algorithm
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-2
0
2
4
6
8


MSE LMS algorithm
13.6
23.8
8.3
8.3
0.78
0.49
23.8
13.6
37
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-2
0
2
4
6
8


MSE LMS/Newton and LMS algorithms
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-2
0
2
4
6
8

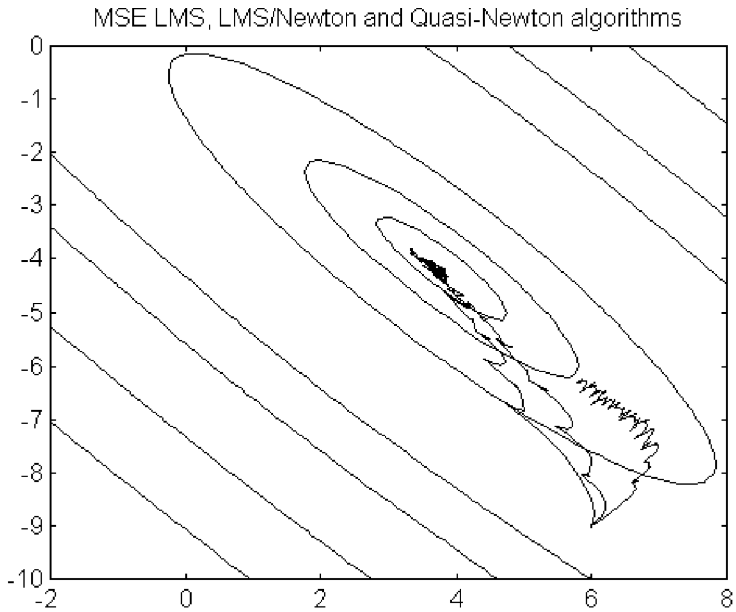
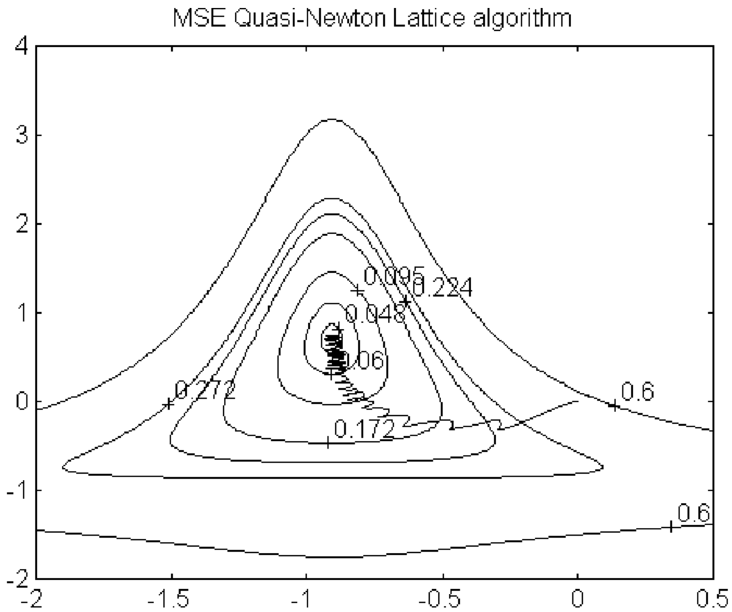

MSE LMS, LMS/Newton and Quasi-Newton algorithms
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
-2
0
2
4
6
8

Figure 21:



MSE Quasi-Newton Lattice algorithm
0.095
0.224
0.048
0.06
0.272
0.6
0.172
0.6
4
3
2
1
0
-1
-2
-2
-1.5
-1
-0.5
0
0.5