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:

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

where $R_x > 0$ and $p$ and $\rho$ 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.](image)
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 $y(n)$.

![Diagram](image1.png)

**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,

$$E\{e^2(n)\} = E\{(y(n) - \hat{y}(n))^2\}$$  \hspace{1cm} (3)

- Second order statistics known, i.e.,

$$p(k) = E\{y(n - k)x(n)\}$$

$$r(k) = E\{x(n - k)x(n)\}$$

$$\rho = E\{y^2(n)\}$$
2.1.2 The inverse filtering problem

\(x(n) = s(n) + v_1(n), \ y(n) = s(n) \) (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 \(0 \leq k \leq n\), so \(h(k) = 0\) for \(k < 0\) and \(k > n\).

- Causal IIR case: \(x(n-k)\) known for \(k \geq 0\), so \(h(k) = 0\) for \(k < 0\) and \(\|h(n)\|^2 < \infty\).

\[
E[e^2(n)] = E[y^2(n)] - 2 \sum_k h(k)E[y(n)x(n-k)] \\
+ \sum_k \sum_l h(l)h(k)E[x(n-k)x(n-l)] \\
= \rho - 2 \sum_k h(k)p(k) + \sum_k \sum_l h(k)h(l)r(l-k)
\]

or

\[
\begin{align*}
  r_e(k) &= r_y(k) - r_y \hat{y}(k) - r_{yy}(k) + r_{y\hat{y}}(k) \\
  &= \rho(k) - \sum_l [h(l)p(l+k) + p(l)h(l+k)] \\
  &\quad + \sum_l \sum_j h(l)r(l+k-j)h(j) \\
  S_e(e^{iw}) &= S_y(e^{iw}) - S_{yx}(e^{iw})H^*(e^{iw}) - S_{yx}^*(e^{iw})H(e^{iw}) + S_x(e^{iw})|H^*(e^{iw})|^2 \\
  &= \left|H(e^{iw}) - \frac{S_{yx}(e^{iw})}{S_x(e^{iw})}\right|^2 S_x(e^{iw}) + \left[S_y(e^{iw}) - \frac{|S_{yx}(e^{iw})|^2}{S_x(e^{iw})}\right]
\end{align*}
\]
bullet the non causal case: \( S_{yx}(e^{jw}) = S_x(e^{jw}) \) and \( S_z(e^{jw}) = S_s(e^{jw}) + S_v(e^{jw}) \), i.e.,

\[
H(e^{jw}) = \frac{S_{yu}(e^{jw})}{S_x(e^{jw})} = \frac{S_s(e^{jw})}{S_s(e^{jw}) + S_v(e^{jw})}
\]

bullet the causal FIR case:

\[
E[e^2(n)] = E[y^2(n)] - 2 \sum_{k=0}^{N} h(k) E[y(n)x(n-k)]
\]

\[
+ \sum_{k=0}^{N} \sum_{l=0}^{N} h(l)h(k)E[x(n-k)x(n-l)]
\]

\[
= \rho - 2 \sum_{k=0}^{N} h(k)p(k) + \sum_{k=0}^{N} \sum_{l=0}^{N} h(k)h(l)r(l-k)
\]

that is minimized for

\[
p(k) = \sum_{n=0}^{N} r(k-n)h(n), \text{ or }
\]

\[
0 = E[e(n)x(n-k)], \text{ for } 0 \leq k \leq N
\]
Let consider two cases:

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

\[
\begin{bmatrix}
R_s + R_u \\
\end{bmatrix} \begin{bmatrix}
h(0) \\
h(N) \\
\vdots \\
h(2N) \\
\end{bmatrix} = \begin{bmatrix}
R_s \\
1 \\
\vdots \\
0 \\
\end{bmatrix}
\]

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:

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

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

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

The solutions are related by:

\[
g_k = h_{N-k} \quad \text{for} \quad k = 1, ..., N - 1
\]
• **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 \( 0 \leq k < \infty . \) 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., \( S_{\nu \nu}(e^{jw}) = 0. \)

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

\[
F(e^{jw}) = \frac{S_{yx}(e^{jw})}{S_y(e^{jw})} \\
S_\nu(e^{jw}) = S_y(e^{jw}) - |F(e^{jw})|^2 = S_y(e^{jw}) - \frac{|S_{yx}(e^{jw})|^2}{S_y(e^{jw})} \\
S_e(e^{jw}) = |F(e^{jw}) - G(e^{jw})H(e^{jw})|^2 + S_\nu(e^{jw})
\]

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,

\[
\hat{y}(n) = b_0x(n) + b_1x(n - 1) + \ldots + b_Nx(n - N) = \theta^T x(n)
\]

where \( \theta = [b_0 \ldots b_N]^T \) and \( x(n) = [x(n) \ldots x(n - N)]^T \).

Then in an stationary environment,

\[
E\{e^2(n)\} = \rho - 2 \theta^T p + \theta^T R_x \theta
\]

where \( R_x = E\{x(n)x^T(n)\} \) and \( p = E\{x(n)y(n)\} \) are known.

A quadratic function of \( \theta(n) \) with

\[
\nabla = \frac{\partial E\{e^2(n)\}}{\partial \theta} = \begin{bmatrix} \frac{\partial E\{e^2(n)\}}{\partial b_0} & \frac{\partial E\{e^2(n)\}}{\partial b_1} & \ldots & \frac{\partial E\{e^2(n)\}}{\partial b_N} \end{bmatrix}^T
\]

\[
= -2p + 2R_x \theta = 0
\]

then the MSE is minimized when

\[
\theta_o = R_x^{-1}p
\]

Note also that

\[
\nabla = \frac{\partial E\{e^2(n)\}}{\partial \theta} = E[2e(n) \frac{\partial e(n)}{\partial \theta}] = -2E[e(n)x(n)] = 0
\]

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 ($a(0) = 1$).

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

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

$$\begin{bmatrix} r(0) & r(1) & \cdots & r(N) \\ r(1) & r(0) & \cdots & r(N-1) \\ \vdots & \vdots & \ddots & \vdots \\ r(N) & r(N-1) & \cdots & r(0) \end{bmatrix} \begin{bmatrix} 1 \\ b(1) \\ \vdots \\ b(N) \end{bmatrix} = \begin{bmatrix} \beta \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$
An important property: the outputs collection of whitening backward filters of orders 1 to \( N \) are orthogonal, i.e.,

\[
E\{e^k_b e^m_b\} = \begin{cases} 
\beta_k & \text{if } m = k \\
0 & \text{if } m \neq k 
\end{cases}
\]

where \( \beta_k = E\{(e^k_b)^2\} \).

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

\[
D_L = \text{diag}[\beta_0, \ldots, \beta_N] \\
L = \begin{bmatrix}
1 & 0 & 0 & \cdots \\
b_N(1) & 1 & \cdots \\
& \ddots & \ddots \\
b_N(N) & \cdots & b_1(1) & 1
\end{bmatrix} \\
R^{-1} = LD_L^{-1}L^T
\]

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

\[
R^{-1} = UD_U^{-1}U^T \\
U = \begin{bmatrix}
1 & a_1(1) & \cdots & a_N(N) \\
0 & 1 & \cdots & a_N(N-1) \\
& \ddots & \ddots & \vdots \\
0 & 0 & \cdots & 1
\end{bmatrix} \\
D_U = \text{diag}[\alpha_0, \ldots, \alpha_N]
\]

where can be shown that \( \alpha_k = \beta_k \).
2.2 Optimization in the ideal setting

2.2.1 Newton algorithm

\[ \theta(n+1) = \theta(n) - \mu R_x^{-1} \nabla(n) = \theta(n) + \mu R_x^{-1}(-2p + 2R_x \theta(n)) = (I - 2\mu I)\theta(n) + 2\mu \theta_o \]

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

2.2.2 Steepest Descent algorithm

Using \( \nabla(n) = 2(R_x \theta(n) - p) \), then

\[ \theta(n+1) = \theta(n) - \mu \nabla(n) = \theta(n) + 2\mu p - 2\mu R_x \theta(n) \]

With \( \tilde{\theta}(n) = \theta(n) - \theta_o \),

\[ \tilde{\theta}(n+1) = (I - 2\mu R_x) \tilde{\theta}(n) = (I - 2\mu R_x)^{n+1} \tilde{\theta}(0) \]

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

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

Then, SD algorithm converges to the Wiener solution if, for \( n \to \infty \), \( \mu \) satisfy

\[ 0 < \mu < \frac{2}{\lambda_{max}} \]

where \( \lambda_{max} \) is the maximum eigenvalue of \( R_x \).
2.2.3 Conjugate direction algorithm

Consider \( d_j, j = 0, ..., N \), that verify

\[
d_j^T R_x d_k = 0 \quad \text{if} \ j \neq k
\]

as conjugate directions. Then:

Theorem: The sequence

\[
\theta(n + 1) = \theta(n) + \gamma_n d_n
\]

where \( \gamma_n = -(d_n^T R_x d_n)^{-1} d_n^T \nabla(n) \) and \( \nabla(n) = 2(R_x \theta(n) - p) \)

converges to \( \theta_o = R_x^{-1} p \) after \( N + 1 \) steps, i.e., \( \theta(n) = \theta_o \).

Assuming to minimize the MSE with \( \theta(n) \) constrained in \( \theta(0) + D_N \), where \( D_N = [d_0 \ d_1 \ ... \ d_{N-1}] \). Then \( (\theta(n) - \theta_o) \) is given by the \( R_x \)-orthogonal projection of \( (\theta_o - \theta(0)) \) onto \( D_N \), i.e.,

\[
\theta(n) = \theta(0) + D_N (D_N^T R_x D_N)^{-1} D_N^T R_x (\theta_o - \theta(0))
\]

\[
= \theta(0) + \sum_{k=0}^{N-1} d_k (d_k^T R_x d_k)^{-1} d_k^T R_x (\theta_o - \theta(0))
\]

But \( d_k^T R_x \theta_o = d_k^T R_x \theta(k) \) so that with \( p = R_x \theta_o \) is possible to verify

\[
d_k^T R_x (\theta_o - \theta(k)) = -d_k^T \nabla(k)
\]

that serves to justify the gain \( \gamma_k \).
2.3 Updating algorithms

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

- An estimate is \textit{unbiased} if \( E\{\theta(n)\} = \theta_o \).
- An estimate is consistent if \( \theta(n) \to \theta_o \) as \( n \to \infty \).

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 \( \nabla \) is not available, a suitable estimate is \( \nabla \approx -2e(n)x(n) \), the LMS algorithm

\[
\theta(n + 1) = \theta(n) + \mu x(n)e(n)
\]

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

Some useful variants

\[
\begin{align*}
\theta(n + 1) &= \theta(n) + \mu x(n) \text{sgn}[e(n)] \quad \text{Sign Error} \\
\theta(n + 1) &= \theta(n) + \mu \text{sgn}[x(n)] e(n) \quad \text{Sign Data} \\
\theta(n + 1) &= \theta(n) + \mu \text{sgn}[x(n)] \text{sgn}[e(n)] \quad \text{Sign Sign}
\end{align*}
\]
2.3.2 Convergence in the Mean and Error variance of the LMS

Using some simplificatory hypotesis, and by defining 
\[ \tilde{\theta}(n) = \theta(n) - \theta_o, \]
and rewritten (4) as follows

\[ \tilde{\theta}(n + 1) = [I - \mu x(n)x^T(n)] \tilde{\theta}(n) + \mu x(n)(y(n) - x^T(n)\theta_o) \]  

then

\[ E\{\tilde{\theta}(n + 1)\} = E\{[I - \mu x(n)x^T(n)] \tilde{\theta}(n)\} + \mu E\{x(n)(y(n) - x^T(n)\theta_o)\} \]  

Using the hypotesis

\[ E\{\tilde{\theta}(n + 1)\} = [I - \mu R_x] E\{\theta(n)\} \]  

Using \( R_x = Q\Lambda Q^T \) (Cholesky) and pre-multiplying (7) by \( Q^T \) and defining

\[ \vartheta(n) = Q^T\tilde{\theta}(n) \]  

is possible to obtain

\[ E\{\vartheta(n + 1)\} = [I - \mu \Lambda] E\{\vartheta(n)\} \]
\[ = [I - \mu \Lambda]^{n+1} E\{\vartheta(0)\} \]  

Then, in order that \( \vartheta(n) \) converge in the mean to the Wiener solution

\[ 0 < \mu < \frac{1}{\lambda_{max}} \]
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

$$
\Delta \xi(n) = \xi(n) - \xi_{min} = E\{ \tilde{\theta}^T(n) R_x \tilde{\theta}(n) \} = E\{ tr(R_x \tilde{\theta}(n)) \tilde{\theta}^T(n) \} = tr \left( E\{ R_x \tilde{\theta}(n) \tilde{\theta}^T(n) \} \right)
$$

where $tr(AB) = tr(BA)$ was used.

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

$$
\Delta \xi(n) \approx \frac{\mu \sigma^2_\nu \sum_{k=0}^N \lambda_k}{1 - \mu \sum_{k=0}^N \lambda_k} = \frac{\mu \sigma^2_\nu tr[R]}{1 - \mu tr[R]}
$$

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

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

and assuming $\mu$ small enough,

$$
\xi_{exc} \approx \mu \sigma^2_\nu tr[R] = \mu (N + 1) \sigma^2_\nu \sigma^2_x
$$

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

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

$$
\xi(n) = \xi_{\text{min}} + E\{\tilde{\theta}^T(n)\Lambda\tilde{\theta}(n)\}
$$

$$
= \xi_{\text{min}} + \sum_{k=0}^{N} \lambda_k \tilde{\theta}^2_k(n)
$$

$$
= \xi_{\text{min}} + \sum_{k=0}^{N} \lambda_k (1 - \mu \lambda_k)^2 \tilde{\theta}^2_k(0)
$$

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 $\theta^*(n) = \theta(n) + \Delta \theta(n)$, given by $e^2_*(n)$.

- Then

$$
\Delta e^2(n) = e^2_*(n) - e^2(n)
= -2\Delta \theta^T(n)x(n)e(n) \\
+ \Delta \theta^T(n)x(n)x^T(n)\Delta \theta(n)
$$

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

$$
\mu(n) = \left( \frac{1}{2x^T(n)x(n)} \right)
$$
2.3.5 The Transform-Domain LMS Algorithm

- Convergence speed is related to different principal axes length of the **MSE surface** contours.

- If these contours are circular the optimum situation is at hand. This can be achieved only if the eigenvectors of the $R_x$ matrix are known.

- The MSE surface is changed by a coordinate transform, $\hat{x}(n) = T x(n)$ where $TT^T = I$, or

\[
\xi = E\{e^2(n)\} = \xi_{min} + \tilde{\theta}^T E\{\hat{x}(n)\hat{x}^T(n)\} \tilde{\theta}
\]

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

\[
\xi - \xi_{min} = \tilde{\theta}^T TR_x T^T \tilde{\theta}
\]

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 coordinates is \( \xi - \xi_{\text{min}} = [TR_xT']_ii \hat{\theta}_i \).

• For an hypersphere it is necessary that \( |\hat{\theta}_i| = |\hat{\theta}_j| \) for all \((i, j)\).

• This conditions can be achieved, at least approximately, using an scaling factor

\[
[TR_xT']_ii \cong E\{\hat{x}_i^2(n)\} = \hat{\sigma}_i^2
\]

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

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

where \( \Lambda = \text{diag} [\hat{\sigma}_1^2, ..., \hat{\sigma}_N^2] \) and \( \hat{\sigma}_i^2(n + 1) = (1 - \mu_\sigma)\hat{\sigma}_i^2(n) + \mu_\sigma \hat{x}_i^2(n) \), with \( \mu_\sigma \) 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 $R_x^{-1}$.

- A possible algorithm is the following

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

where

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

- $P(n+1)$ represents an estimate of $R_x^{-1}$ 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:

\[ y(n) = \sum_{k=1}^{N} \theta_k^0 x(n-k) + \nu(n) \]

The RLS algorithm estimates the \( \theta_o \) parameters by minimizing

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

where \( e(n) = y(n) - \theta^T(n) x_N(n) \). The well known recursive solution of this problem is

\[ \theta(n) = \theta(n-1) + \kappa_N(n) (y(n) - \theta^T(n-1) x_N(n)) \]

where

\[ \kappa_N(n) = \frac{1}{R_{N-1}(n)} x_N(n) \quad n \geq N \]

\[ R_{N-1}(n) = R_{N-1}(n-1) + x_N(n)x_N^T(n) \]
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) $\kappa_N(n - 1) \rightarrow \kappa_N(n)$.

- This updating can be seen as composed of time update and order update.

  1. $\kappa_N(n - 1) \rightarrow \kappa_{N+1}(n)$,

  2. $\kappa_{N+1}(n) \rightarrow \kappa_N(n)$.

- Due to the shifted structure of the regressor $x_N(n)$,

  $$x_{N+1}(n) = \begin{bmatrix} x(n) \\ x_N(n-1) \end{bmatrix} = \begin{bmatrix} x_N(n) \\ x(n-N) \end{bmatrix}$$

  where $x_{N+1,1:N}(n) = x_N(n-1)$ and $x_{N+1,0:N-1}(n) = x_N(n)$.

- Then, assuming that $x_N(n) = 0$ for $n \leq 0$,

  $$R_{N-1}(n-1) = R_{1:N}(n) \quad R_{N-1}(n) = R_{0:N-1}(n)$$

  where $R_{1:N}(n)$ and $R_{0:N-1}(n)$ are the lower right and upper left corner of $R_N(n)$, respectively.
• Kalman gain at times \( n - 1 \) and \( n \) can be written

\[
\kappa_N(n - 1) = R_{1:N}^{-1}(n - 1)x_{N+1:1:N}(n) \\
\kappa_N(n) = R_{0:N-1}^{-1}(n)x_{N+1:0:N-1}(n) \\
\kappa_{N+1}(n) = R_{N}^{-1}(n)x_{N+1}(n)
\]

• Following the first step (time update) above, the problem can be stated has: given \( \kappa_N(n - 1) \) and \( \kappa_N(n) \), find \( \kappa_{N+1}(n) \) as the solution to the \( N + 1 \)-dimensional problem

\[
\text{Minimize} \quad z = \kappa_{N+1}(n) \left( \frac{1}{2} z^T R_N(n) z - x_{N+1}^T(n) z \right)
\]

• This can be achieved with a conjugate direction algorithm with

\[
d_N = [1 \ a_N^T(n)]^T \\
d_N^T \nabla(n) = [1 \ a_N^T(n)]x_{N+1}(n) = e_N^f(n) \\
d_N^T R_N(n) d_N = \xi_N^f(n)
\]

where \( a_N(n) \) 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

\[
\kappa_{N+1}(n) = \begin{bmatrix} 0 \\ \kappa_N(n - 1) \end{bmatrix} + \begin{bmatrix} 1 \\ \alpha_N(n) \end{bmatrix} (\xi_N^f(n))^{-1} e_N^f(n)
\]
• For the second step (order update), i.e., $\kappa_{N+1}(n) \rightarrow \kappa_{N}(n)$, the problem can be stated as: given $\kappa_{N}(n-1)$ and $\kappa_{N}(n)$, find $\kappa_{N+1}(n)$ as the solution to the $N+1$-dimensional problem

$$
\minimize \ z = \kappa_{N+1}(n) \ \left( \frac{1}{2} z^T R_N(n) z - x_{N+1}^T(n) z \right)
$$

• This can be achieved using a conjugate direction algorithm with

$$
d_N = \begin{bmatrix} b_N^T(n) \\ 1 \end{bmatrix}^T
$$

$$
d_N^T \nabla(n) = [b_N^T(n) \ 1]x_{N+1}(n) = e_{N}^b(n)
$$

$$
d_N^T R_N(n) d_N = \xi_N^b(n)
$$

where $b_N(n)$ 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 a posteriori backward prediction error.

• Using this results,

$$
\kappa_{N+1}(n) = \begin{bmatrix} \kappa_{N}(n) \\ 0 \end{bmatrix} + \begin{bmatrix} b_N(n) \\ 1 \end{bmatrix} (\xi_N^b(n))^{-1} e_{N}^b(n)
$$

• Since the required solution is the Kalman time update, $\kappa_{N}(n)$ is obtained as a function of $\kappa_{N+1}(n)$ 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 $\theta(n)$. 
2.4.3 QR decomposition based RLS algorithm

- If the standard RLS algorithm

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

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

\[
\begin{bmatrix}
e(n) \\
0 \\
\vdots \\
0
\end{bmatrix} - \begin{bmatrix}
x^T(n) \\
\lambda^{1/2}x^T(n-1) \\
\vdots \\
\lambda^{1/2}x^T(0)
\end{bmatrix} \tilde{\theta}(n) = 0 \quad (13)
\]

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

- Then \(e(n)u_1 - X(n)\tilde{\theta}(n)\), where \(u_1\) is the unit vector with a ”1” in the first position, and

\[
X(n) = \begin{bmatrix}
x^T(n) \\
\lambda^{1/2}X(n-1)
\end{bmatrix}
\]

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

\[
Q(n-1)X(n-1) = \begin{bmatrix}
\bigcirc \\
R(n-1)
\end{bmatrix}
\]

where \(R(n-1)\) is an upper triangular matrix of dimension \(N \times N\) (dimension of \(\theta(n)\)).
• Then

\[
\begin{bmatrix}
1 \\
Q(n-1)
\end{bmatrix}
(e(n)u_1 - X(n)\hat{\theta}(n)) =
\begin{bmatrix}
e(n) \\
0 \\
\vdots \\
0
\end{bmatrix}
- \begin{bmatrix}
x^T(n) \\
\circ \\
\lambda^{1/2} R(n-1)
\end{bmatrix}
\]

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

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

\[
\hat{Q}(n)
\begin{bmatrix}
e(n) \\
0 \\
\vdots \\
0
\end{bmatrix}
- \hat{Q}(n)
\begin{bmatrix}
x^T(n) \\
\circ \\
\lambda^{1/2} R(n-1)
\end{bmatrix}
\tilde{\theta}(n) = e(n)\hat{q}_1(n) - \begin{bmatrix}
\circ \\
R(n)
\end{bmatrix}\tilde{\theta}(n)
\]

(14)

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

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

\[
\hat{Q}(n) = \hat{Q}_N...\hat{Q}_1
\]

with

\[
\hat{Q}_k = \begin{bmatrix}
\cos \varphi_k & -\sin \varphi_k \\
\sin \varphi_k & \cos \varphi_k
\end{bmatrix}
\begin{bmatrix}
I_{n+k-N-1} \\
I_{N-k}
\end{bmatrix}
\]

• The proper selection of the rotation angles \( \{\varphi_k\} \) will annihilate the elements of \( x^T(n) \) appearing in (14).
• The term $\hat{q}_1(n)$ in (14) in closed form is

$$\hat{q}_1(n) = \left[ \begin{array}{c} \prod_{k=1}^{N} \cos \varphi_k \\ 0 \\ g \end{array} \right]$$

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

• The parameter update $\hat{\theta}(n)$ is then solved from

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

using back substitution.

• An useful scaled algorithm can be obtained considering the QR decomposition of $X(n)$

$$Q(n)X(n) = \left[ \begin{array}{c} \bigcirc \\ R(n) \end{array} \right]$$

• Because $Q(n)$ is orthogonal, we have

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

• If $x(n)$ is stationary,

$$n \xrightarrow{\lim} \infty \ E\{R^T(n)R(n)\} = \frac{E\{x(n)x^T(n)\}}{1 - \lambda}$$

• Then for $\lambda \to 1$, the elements of $R(n)$ can become large.

• Overflow in $R(n)$ can be avoided considering in (13) that if $\{x(k)\}_{k=0}^{n}$ and $e(n)$ are similarly scaled, the LS is left unchanged.

• From (15) an appropriate choice is $\sqrt{1 - \lambda}$. 
Examples

- An (adaptive) signal-cancelling application, with two taps!
- \( x(n) = \sin w_0 n + \nu(n) \) with \( E\{\nu^2(n)\} = r \),
- \( y(n) = 2 \cos w_0 n \),
- \( y(n) = \theta_0 x(n) + \theta_1 x(n - 1) \),
- \( e(n) = y(n) - \hat{y}(n) \)
- \( R_x = \frac{1}{2} \begin{bmatrix} 1 + 2r & \cos w_0 \\ \cos w_0 & 1 + 2r \end{bmatrix} \) and \( p = 2 \begin{bmatrix} 0 \\ -\sin w_0 \end{bmatrix} \).
- \( \theta^* = R^{-1} p \).
- \( E\{e^2(n)\} = (\frac{1}{2} + r)(\theta_0^2 + \theta_1^2) + \theta_0 \theta_1 \cos w_0 + 2\theta_1 \sin w_0 + 2 \)
Figure 21: