SGN 21006 Advanced Signal Processing: Lecture 7 Least squares and RLS algorithms

Ioan Tabus

Department of Signal Processing
Tampere University of Technology
Finland

Outline

- Linear LS estimation problem;
- Normal equations
- Properties of Least-Squares estimates;
- The exponentially weighted Least squares
- Recursive-in-time solution
- Initialization of the algorithm
- Recursion for MSE criterion
- Examples: Noise canceller, Channel equalization, Echo cancellation

References : Chapters 8 and 9 from *S. Haykin- Adaptive Filtering Theory - Prentice Hall, 2002.*

Linear LS estimation problem

Problem statement

- Given the set of input samples $\{u(1), u(2), \ldots, u(N)\}$ and the set of desired response $\{d(1), d(2), \ldots, d(N)\}$
- In the family of linear filters computing their output according to

$$y(n) = \sum_{k=0}^{M-1} w_k u(n-k), \quad n = 0, 1, 2, \ldots \tag{1}$$

- Find the parameters $\{w_0, w_1, \ldots, w_{M-1}\}$ such as to minimize the sum of error squares

$$\mathcal{E}(w_0, w_1, \ldots, w_{M-1}) = \sum_{i=i_1}^{i_2} [e(i)^2] = \sum_{i=i_1}^{i_2} [d(i) - \sum_{k=0}^{M-1} w_k u(i-k)]^2$$

where the error signal is

$$e(i) = d(i) - y(i) = d(i) - \sum_{k=0}^{M-1} w_k u(i-k)$$

□

Data windows

Using the vector notations:

$$\begin{aligned} \underline{u}(n) &= \begin{bmatrix} u(n) & u(n-1) & u(n-2) & \dots & u(n-M+1) \end{bmatrix}^T \\ \underline{w} &= \begin{bmatrix} w_0 & w_1 & \dots & w_{M-1} \end{bmatrix}^T \end{aligned} \tag{2}$$

we can write the filter output at time instant i

$$y(i) = \sum_{k=0}^{M-1} w_k u(i-k) = \begin{bmatrix} u(i) & u(i-1) & u(i-2) & \dots & u(i-M+1) \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \dots \\ w_{M-1} \end{bmatrix} = \underline{u}(n)^T \underline{w}$$

The criterion $\mathcal{E}(w_0, w_1, \dots, w_{M-1})$ will make use of the following error values:

$$\begin{bmatrix} e(i_1) \\ e(i_1+1) \\ \dots \\ e(i_2) \end{bmatrix} = \begin{bmatrix} d(i_1) \\ d(i_1+1) \\ \dots \\ d(i_2) \end{bmatrix} - \begin{bmatrix} y(i_1) \\ y(i_1+1) \\ \dots \\ y(i_2) \end{bmatrix}$$

$$= \begin{bmatrix} d(i_1) \\ d(i_1+1) \\ \dots \\ d(i_2) \end{bmatrix} - \begin{bmatrix} u(i_1) & u(i_1-1) & u(i_1-2) & \dots & u(i_1-M+1) \\ u(i_1+1) & u(i_1) & u(i_1-1) & \dots & u(i_1-M+2) \\ . & . & . & \dots & . \\ u(i_2) & u(i_2-1) & u(i_2-2) & \dots & u(i_2-M+1) \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \dots \\ w_{M-1} \end{bmatrix}$$

Data windows

Making use of available data in LS criterion: Selecting the limits i_1 and i_2
There are four ways of selecting the limits i_1 and i_2 and making use of simplifying assumptions:

- **Covariance method**: Uses *only* available data: $i_1 = M$ and $i_2 = N$

$$A = \begin{bmatrix} u(M) & u(M-1) & u(M-2) & \dots & u(1) \\ u(M+1) & u(M) & u(M-1) & \dots & u(2) \\ . & . & . & \dots & . \\ u(N) & u(N-1) & u(N-2) & \dots & u(N-M+1) \end{bmatrix}$$

- **Autocorrelation (Pre– and Post–windowing) method**: Uses unavailable data: $i_1 = 1$ and $i_2 = N + M - 1$. Assumes input data prior to $u(1)$ and after $u(N)$ are zero

$$A = \begin{bmatrix} u(1) & 0 & 0 & \dots & 0 \\ u(2) & u(1) & 0 & \dots & 0 \\ . & . & . & \dots & . \\ u(M) & u(M-1) & u(M-2) & \dots & u(1) \\ u(M+1) & u(M) & u(M-1) & \dots & u(2) \\ . & . & . & \dots & . \\ u(N) & u(N-1) & u(N-2) & \dots & u(N-M+1) \\ 0 & u(N) & u(N-1) & \dots & u(N-M+2) \\ . & . & . & \dots & . \\ 0 & 0 & 0 & \dots & u(N) \end{bmatrix}$$

Data windows

- **Prewindowing method**: Uses unavailable data: $i_1 = 1$ and $i_2 = N$. Assumes input data prior to $u(1)$ are zero

$$A = \begin{bmatrix} u(1) & 0 & 0 & \dots & 0 \\ u(2) & u(1) & 0 & \dots & 0 \\ . & . & . & \dots & . \\ u(M) & u(M-1) & u(M-2) & \dots & u(1) \\ u(M+1) & u(M) & u(M-1) & \dots & u(2) \\ . & . & . & \dots & . \\ u(N) & u(N-1) & u(N-2) & \dots & u(N-M+1) \end{bmatrix}$$

- **Post–windowing method**: Uses unavailable data: $i_1 = M$ and $i_2 = N + M - 1$. Assumes input data after $u(N)$ are zero

$$A = \begin{bmatrix} u(M) & u(M-1) & u(M-2) & \dots & u(1) \\ u(M+1) & u(M) & u(M-1) & \dots & u(2) \\ . & . & . & \dots & . \\ u(N) & u(N-1) & u(N-2) & \dots & u(N-M+1) \\ 0 & u(N) & u(N-1) & \dots & u(N-M+2) \\ . & . & . & \dots & . \\ 0 & 0 & 0 & \dots & u(N) \end{bmatrix}$$

Principle of orthogonality for LS filters

- **Principle of orthogonality for LS filters** When the minimum value of the criterion will be attained, the gradient of criterion with respect to parameter vector will be zero:

$$\nabla_{\underline{w}} \mathcal{E}(\underline{w}) = \nabla_{\underline{w}} \sum_{i=i_1}^{i_2} [e(i)^2] = 2 \sum_{i=i_1}^{i_2} e(i) \nabla_{\underline{w}} e(i) = 0$$

which can be written for each component of the gradient vector

$$\nabla_k \mathcal{E}(\underline{w}) = 2 \sum_{i=i_1}^{i_2} e(i) \nabla_k e(i) = 2 \sum_{i=i_1}^{i_2} e(i) \frac{\partial}{\partial w_k} [d(i) - \sum_{l=0}^{M-1} w_l u(i-l)] = -2 \sum_{i=i_1}^{i_2} e(i) u(i-k) = 0$$

$$\sum_{i=i_1}^{i_2} e(i) u(i-k) =$$

$$= \begin{bmatrix} e(i_1) & e(i_1+1) & \dots & e(i_2) \end{bmatrix} \begin{bmatrix} u(i_1-k) & u(i_1-k+1) & \dots & u(i_2-k) \end{bmatrix}^T = 0$$

Principle of orthogonality for LS filters

$$\sum_{i=i_1}^{i_2} e_o(i) u(i-k) = 0 \qquad k = 0, 1, \dots, M-1$$

The minimum error time series is orthogonal to the input time series shifted backward with k units, for $k = 0, 1, 2, \dots, M-1$

Principle of orthogonality for LS filters

- The output of the filter for optimal parameters is also orthogonal to the errors:

$$\sum_{i=i_1}^{i_2} e_o(i) y_o(i) = \sum_{i=i_1}^{i_2} e_o(i) \sum_{l=0}^{M-1} \hat{w}_l u(i-l) = \sum_{l=0}^{M-1} \hat{w}_l \sum_{i=i_1}^{i_2} e_o(i) u(i-l) = 0$$

Corollary of principle of orthogonality

$$\sum_{i=i_1}^{i_2} e_o(i) y_o(i) = 0$$

The minimum error time series is orthogonal to the optimal LS filter output time series

Normal equations

- Normal equations

Rearranging the orthogonality equations we have for all $k = 0, 1, \ldots, M-1$

$$\sum_{i=i_1}^{i_2} e_o(i)u(i-k) = 0$$

$$\sum_{i=i_1}^{i_2} [d(i) - \sum_{l=0}^{M-1} \hat{w}_l u(i-l)]u(i-k) = 0$$

$$\sum_{i=i_1}^{i_2} d(i)u(i-k) = \sum_{i=i_1}^{i_2} \sum_{l=0}^{M-1} \hat{w}_l u(i-l)u(i-k)$$

$$\sum_{i=i_1}^{i_2} d(i)u(i-k) = \sum_{l=0}^{M-1} \hat{w}_l \sum_{i=i_1}^{i_2} u(i-l)u(i-k)$$

and denoting

$$\Phi(l,k) = \sum_{i=i_1}^{i_2} u(i-l)u(i-k) = \Phi(k,l) \quad \text{and} \quad \psi(k) = \sum_{i=i_1}^{i_2} d(i)u(i-k)$$

we obtain the system of equations

$$\sum_{l=0}^{M-1} \hat{w}_l \Phi(l,k) = \psi(k), \qquad k = 0, 1, \ldots, M-1$$

Normal equations

$$\begin{cases} \Phi(0,0)\hat{w}_0 + \Phi(1,0)\hat{w}_1 + \ldots + \Phi(M-1,0)\hat{w}_{M-1} &= \psi(0) \\ \Phi(0,1)\hat{w}_0 + \Phi(1,1)\hat{w}_1 + \ldots + \Phi(M-1,1)\hat{w}_{M-1} &= \psi(1) \\ \ldots &= \ldots \\ \Phi(0,M-1)\hat{w}_0 + \Phi(1,M-1)\hat{w}_1 + \ldots + \Phi(M-1,M-1)\hat{w}_{M-1} &= \psi(M-1) \end{cases}$$

and using the vector notation

$$\underline{\psi} = \begin{bmatrix} \psi(0) & \psi(1) & \psi(2) & \ldots & \psi(M-1) \end{bmatrix}^T$$

we may rewrite the normal equations:

$$\begin{bmatrix} \Phi(0,0) & \Phi(1,0) & \Phi(2,0) & \ldots & \Phi(M-1,0) \\ \Phi(0,1) & \Phi(1,1) & \Phi(2,1) & \ldots & \Phi(M-1,1) \\ \Phi(0,2) & \Phi(1,2) & \Phi(2,2) & \ldots & \Phi(M-1,2) \\ . & . & . & \ldots & . \\ . & . & . & \ldots & . \\ \Phi(0,M-1) & \Phi(1,M-1) & \Phi(2,M-1) & \ldots & \Phi(M-1,M-1) \end{bmatrix} \begin{bmatrix} \hat{w}_0 \\ \hat{w}_1 \\ \hat{w}_2 \\ . \\ . \\ \hat{w}_{M-1} \end{bmatrix} = \begin{bmatrix} \psi(0) \\ \psi(1) \\ \psi(2) \\ . \\ . \\ \psi(M-1) \end{bmatrix}$$

or in compact notations

$$\Phi\underline{\hat{w}} = \underline{\psi}$$

$$\underline{\hat{w}} = [\Phi]^{-1}\underline{\psi}$$

Minimum sum of Error Squares in LS estimation problem

Minimum sum of Error Squares

$$\begin{aligned}
\mathcal{E}(\underline{\hat{w}}) &= \sum_{i=i_1}^{i_2}[e_o(i)^2] = \sum_{i=i_1}^{i_2} e_o(i)(d(i) - y_o(i)) = \sum_{i=i_1}^{i_2} e_o(i)d(i) - \sum_{i=i_1}^{i_2} e_o(i)y_o(i) = \sum_{i=i_1}^{i_2}(d(i) - y_o(i))d(i) \\
&= \sum_{i=i_1}^{i_2}(d(i))^2 - \sum_{i=i_1}^{i_2}\sum_{l=0}^{M-1} \hat{w}_l u(i-l)d(i) = \sum_{i=i_1}^{i_2}(d(i))^2 - \sum_{l=0}^{M-1} \hat{w}_l \sum_{i=i_1}^{i_2} u(i-l)d(i) \\
&= \sum_{i=i_1}^{i_2}(d(i))^2 - \sum_{l=0}^{M-1} \hat{w}_l \psi(l) = \sum_{i=i_1}^{i_2}(d(i))^2 - \underline{\hat{w}}^T \underline{\psi} = \sum_{i=i_1}^{i_2}(d(i))^2 - \underline{\hat{w}}^T \Phi \underline{\hat{w}}
\end{aligned}$$

Compact formulations

Compact formulations using data matrices

$$A = \begin{bmatrix} u(i_1) & u(i_1-1) & u(i_1-2) & \dots & u(i_1-M+1) \\ u(i_1+1) & u(i_1) & u(i_1-1) & \dots & u(i_1-M+2) \\ . & . & . & \dots & . \\ u(i_2) & u(i_2-1) & u(i_2-2) & \dots & u(i_2-M+1) \end{bmatrix} = \begin{bmatrix} \underline{u}(i_1)^T \\ \underline{u}(i_1+1)^T \\ . \\ \underline{u}(i_2)^T \end{bmatrix}$$

Computing $A^T A = \Phi$

$$\begin{bmatrix} u(i_1) & u(i_1+1) & \dots & u(i_2) \\ u(i_1-1) & u(i_1) & \dots & u(i_2-1) \\ . & . & \dots & . \\ u(i_1-M+1) & u(i_1-M+2) & \dots & u(i_2-M+1) \end{bmatrix} \begin{bmatrix} u(i_1) & u(i_1-1) & \dots & u(i_1-M+1) \\ u(i_1+1) & u(i_1) & \dots & u(i_1-M+2) \\ . & . & \dots & . \\ u(i_2) & u(i_2-1) & \dots & u(i_2-M+1) \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{i=i_1}^{i_2} u(i)^2 & \sum_{i=i_1}^{i_2} u(i)u(i-1) & \dots & \sum_{i=i_1}^{i_2} u(i)u(i-M+1) \\ \sum_{i=i_1}^{i_2} u(i-1)u(i) & \sum_{i=i_1}^{i_2} u(i-1)^2 & \dots & \sum_{i=i_1}^{i_2} u(i-1)u(i_1-M+2) \\ . & . & \dots & . \\ \sum_{i=i_1}^{i_2} u(i-M+1)u(i) & \sum_{i=i_1}^{i_2} u(i-M+1)u(i-1) & \dots & \sum_{i=i_1}^{i_2} u(i-M+1)^2 \end{bmatrix}$$

$$\Phi = A^T A = \begin{bmatrix} \underline{u}(i_1) & \underline{u}(i_1+1) & \dots & \underline{u}(i_2) \end{bmatrix} \begin{bmatrix} \underline{u}(i_1)^T \\ \underline{u}(i_1+1)^T \\ . \\ \underline{u}(i_2)^T \end{bmatrix} = \sum_{i=i_1}^{i_2} \underline{u}(i)\underline{u}(i)^T$$

Compact formulations

$$A^T \underline{d} = \begin{bmatrix} u(i_1) & u(i_1+1) & u(i_1+2) & \dots & u(i_2) \\ u(i_1-1) & u(i_1) & u(i_1+1) & \dots & u(i_2-1) \\ . & . & . & \dots & . \\ u(i_1-M+1) & u(i_1-M+2) & u(i_1-M+3) & \dots & u(i_2-M+1) \end{bmatrix} \begin{bmatrix} d(i_1) \\ d(i_1+1) \\ d(i_1+2) \\ . \\ . \\ d(i_2) \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{i=i_1}^{i_2} u(i)d(i) \\ \sum_{i=i_1}^{i_2} u(i-1)d(i) \\ \sum_{i=i_1}^{i_2} u(i-2)d(i) \\ . \\ . \\ \sum_{i=i_1}^{i_2} u(i-M+1)d(i) \end{bmatrix} = \underline{\psi}$$

$$\underline{\psi} = A^T \underline{d} = \begin{bmatrix} \underline{u}(i_1) & \underline{u}(i_1+1) & \dots & \underline{u}(i_2) \end{bmatrix} \begin{bmatrix} d(i_1) \\ d(i_1+1) \\ d(i_1+2) \\ . \\ . \\ d(i_2) \end{bmatrix} = \sum_{i=i_1}^{i_2} \underline{u}(i)d(i)$$

Compact formulations

Normal equations:

$$(A^T A)\underline{\hat{w}} = (A^T \underline{d})$$
$$\underline{\hat{w}} = (A^T A)^{-1} A^T \underline{d}$$

Minimum sum of error squares

$$\mathcal{E}(\underline{\hat{w}}) \quad = \quad \sum_{i=i_1}^{i_2} (d(i))^2 - \underline{\psi}^T [\Phi]^{-1} \underline{\psi} = \underline{d}^T \underline{d} - \underline{d}^T A (A^T A)^{-1} A^T \underline{d}$$

Projection operator Denote the time series provided by the output of LS filter

$$\underline{\hat{y}} = \begin{bmatrix} \hat{y}(i_1) & \hat{y}(i_1+1) & \hat{y}(i_1+2) & \dots & \hat{y}(i_2) \end{bmatrix}^T$$

$$\underline{\hat{y}} = A\underline{\hat{w}} = A(A^T A)^{-1} A^T \underline{d}$$

The matrix

$$P = A(A^T A)^{-1} A^T$$

is the projector operator onto the linear space spanned by the columns of the data matrix A.

Properties of Least-Squares estimates

- *Property 1* The least squares estimate $\underline{\hat{w}}$ is unbiased, provided that the measurement error process $\underline{\varepsilon}_o$ has zero mean.
Proof When discussing about unbiasedness, we assume the data was generated by a "true" parameter vector $\underline{w}_o$, and corrupted by the error vector $\underline{\varepsilon}_o$, therefore the model of the data is

$$\underline{d} \quad = \quad A\underline{w}_o + \underline{\varepsilon}_o$$

and the LS estimate can be written

$$\begin{aligned} \underline{\hat{w}} &= (A^T A)^{-1}(A^T d) = (A^T A)^{-1} A^T (A\underline{w}_o + \underline{\varepsilon}_0) \\ &= \underline{w}_o + (A^T A)^{-1} A^T \underline{\varepsilon}_0 \end{aligned}$$

Since by hypothesis $E\underline{\varepsilon}_o = 0$,

$$E\underline{\hat{w}} \quad = \quad \underline{w}_o + E(A^T A)^{-1} A^T \underline{\varepsilon}_0 = \underline{w}_o + (A^T A)^{-1} A^T E\underline{\varepsilon}_0 = \underline{w}_o$$

- *Property 2* When the measurement error process $\varepsilon_o(i)$ is white with zero mean and variance σ^2, the covariance matrix of the LS estimate $\underline{\hat{w}}$ equals $\sigma^2(A^T A)^{-1}$.
Proof Under the mentioned hypothesis on $\varepsilon_o(i)$, the vector $\underline{\varepsilon}_o$ has zero mean and covariance matrix

$$E(\underline{\varepsilon}_o \underline{\varepsilon}_o^T) = \sigma^2 I$$

Now the covariance matrix of $\underline{\hat{w}}$ is

$$\begin{aligned} cov(\underline{\hat{w}}) &= E(\underline{\hat{w}} - \underline{w}_o)(\underline{\hat{w}} - \underline{w}_o)^T = E(A^T A)^{-1} A^T \underline{\varepsilon}_0 \underline{\varepsilon}_0^T A (A^T A)^{-1} \\ &= (A^T A)^{-1} A^T E[\underline{\varepsilon}_0 \underline{\varepsilon}_0^T] A (A^T A)^{-1} = (A^T A)^{-1} A^T \sigma^2 A (A^T A)^{-1} = \sigma^2 (A^T A)^{-1} \end{aligned}$$

Properties of Least-Squares estimates

- *Property 3* When the measurement error process $\varepsilon_o(i)$ is white with zero mean and variance σ^2, the LS estimate $\underline{\hat{w}}$ is the best linear unbiased estimate (BLUE).
Proof Consider any unbiased estimator $\tilde{w}$

$$\underline{\tilde{w}} = B\underline{d}$$

where B is an $M \times (N - m + 1)$ matrix, such that $E\underline{\tilde{w}} = \underline{w}_o$, i.e.

$$E\underline{\tilde{w}} = EB\underline{d} = EB(A\underline{w}_o + \underline{\varepsilon}_o) = BA\underline{w}_o + EB\underline{\varepsilon}_o = \underline{w}_o$$

therefore for the unbiasedness of $\underline{\tilde{w}}$ it is necessary that

$$BA = I$$

The covariance matrix of $\underline{\tilde{w}} = BA\underline{w}_o + B\underline{\varepsilon}_o$ is

$$cov(\underline{\tilde{w}}) = E(\underline{\tilde{w}} - \underline{w}_o)(\underline{\tilde{w}} - \underline{w}_o)^T = EB\underline{\varepsilon}_o\underline{\varepsilon}_o^T B^T = \sigma^2 BB^T$$

We show now that $cov(\underline{\tilde{w}}) \geq cov(\underline{\hat{w}})$. Consider the matrix $\Psi = B - (A^T A)^{-1}A^T$ and the product

$$\begin{aligned}\Psi\Psi^T &= (B - (A^T A)^{-1}A^T)(B - (A^T A)^{-1}A^T)^T = \\ &= BB^T - (A^T A)^{-1}A^T B^T - BA(A^T A)^{-1} + (A^T A)^{-1}A^T A(A^T A)^{-1} = BB^T - (A^T A)^{-1}\end{aligned}$$

But $\Psi\Psi^T$ is a semipositive definte matrix (because $x^T\Psi\Psi^T x = ||\Psi^T x||^2 \geq 0$, therfore $BB^T - (A^T A)^{-1} \geq 0$, or $cov(\underline{\tilde{w}}) \geq cov(\underline{\hat{w}})$, which finishes the proof of the property 3.
- One can also show that:
Property 4 When the measurement error process $\varepsilon_o(i)$ is white and Gaussian, with zero mean, the LS estimate $\underline{\hat{w}}$ achieves the Cramer-Rao lower bound for unbiased estimators. Equivalently, it is said that for white Gaussian noise process the least squares is a minimum variance unbiased estimate (MVUE).

Recursive Least Squares Estimation

Problem statement

- Given the set of input samples $\{u(1), u(2), \ldots, u(N)\}$ and the set of desired response $\{d(1), d(2), \ldots, d(N)\}$
- In the family of linear filters computing their output according to

$$y(n) = \sum_{k=0}^{M} w_k u(n-k), \quad n = 0, 1, 2, \ldots \tag{3}$$

- Find *recursively in time* the parameters $\{w_0(n), w_1(n), \ldots, w_{M-1}(n)\}$ such as to minimize the sum of error squares

$$\mathcal{E}(n) = \mathcal{E}(w_0(n), w_1(n), \ldots, w_{M-1}(n)) = \sum_{i=i_1}^{n} \beta(n,i)[e(i)^2] = \sum_{i=i_1}^{n} \beta(n,i)[d(i) - \sum_{k=0}^{M-1} w_k(n)u(i-k)]^2$$

where the error signal is

$$e(i) = d(i) - y(i) = d(i) - \sum_{k=0}^{M-1} w_k(n)u(i-k)$$

and the forgetting factor or weighting factor reduces the influence of old data

$$0 < \beta(n,i) \leq 1, \qquad i = 1, 2, \ldots, n$$

usually taking the form $(0 < \lambda < 1)$

$$\beta(n,i) = \lambda^{n-i}, \qquad i = 1, 2, \ldots, n$$

□

The exponentially weighted Least squares solution

- Writing the criterion with an exponential forgetting factor

$$\mathcal{E}(n) = \mathcal{E}(w_0(n), w_1(n), \ldots, w_{M-1}(n)) = \sum_{i=i_1}^{n} \lambda^{n-i}[e(i)^2] = \sum_{i=i_1}^{n} \lambda^{n-i}[d(i) - \sum_{k=0}^{M-1} w_k(n)u(i-k)]^2$$

Make the following variable changes:

$$u'(i) = \sqrt{\lambda^{n-i}}u(i); \qquad d'(i) = \sqrt{\lambda^{n-i}}d(i) \tag{4}$$

Then the criterion rewrites

$$\mathcal{E}(n) = \sum_{i=i_1}^{n} \lambda^{n-i}[d(i) - \sum_{k=0}^{M-1} w_k(n)u(i-k)]^2 = \sum_{i=i_1}^{n}[d'(i) - \sum_{k=0}^{M-1} w_k(n)u'(i-k)]^2$$

which is the standard LS criterion, in the new variables $u'(i), d'(i)$. The LS solution can be obtained as

$$\underline{w}(n) = (\sum_{i=i_1}^{n} \underline{u}'(i)\underline{u}'(i)^T)^{-1} \sum_{i=i_1}^{n} \underline{u}'(i)d'(i) = (\sum_{i=i_1}^{n} \lambda^{n-i}\underline{u}(i)\underline{u}(i)^T)^{-1} \sum_{i=i_1}^{n} \lambda^{n-i}\underline{u}(i)d(i) = [\Phi(n)]^{-1}\underline{\psi}(n) =$$

where we will denote (making use of Pre-windowing assumption, that data before $i = 1$ is zero)

$$\Phi(n) = \sum_{i=1}^{n} \lambda^{n-i}\underline{u}(i)\underline{u}(i)^T$$

$$\underline{\psi}(n) = \sum_{i=1}^{n} \lambda^{n-i}\underline{u}(i)d(i)$$

□

Recursive in time solution

- We want to find a recursive in time way to compute

$$\underline{w}(n) = [\Phi(n)]^{-1}\underline{\psi}(n)$$

using the information already available at time $n-1$, i.e.

$$\underline{w}(n-1) = [\Phi(n-1)]^{-1}\underline{\psi}(n-1)$$

We therefore will rewrite the variables $\Phi(n)$ and $\underline{\psi}(n)$ as functions of $\Phi(n-1)$ and $\underline{\psi}(n-1)$

$$\begin{aligned}
\Phi(n) &= \sum_{i=1}^{n} \lambda^{n-i}\underline{u}(i)\underline{u}(i)^T = \lambda\sum_{i=1}^{n-1} \lambda^{n-1-i}\underline{u}(i)\underline{u}(i)^T + \underline{u}(n)\underline{u}(n)^T = \lambda\Phi(n-1) + \underline{u}(n)\underline{u}(n)^T \\
\underline{\psi}(n) &= \sum_{i=1}^{n} \lambda^{n-i}\underline{u}(i)d(i) = \lambda\sum_{i=1}^{n-1} \lambda^{n-1-i}\underline{u}(i)d(i) + \underline{u}(n)d(n) = \lambda\underline{\psi}(n-1) + \underline{u}(n)d(n)
\end{aligned}$$

The matrix inversion formula
If A and B are $M \times M$ positive definite matrices, D is a $N \times N$ matrix, and C is a $M \times N$ matrix which are related by

$$A = B^{-1} + CD^{-1}C^T$$

then

$$A^{-1} = B - BC(D + C^T BC)^{-1}C^T B$$

Proof Exercise.

□

Derivation of the algorithm

- Applying the matrix inversion formula to

$$\Phi(n) \quad = \quad \lambda\Phi(n-1) + \underline{u}(n)\underline{u}(n)^T$$

we obtain

$$\Phi^{-1}(n) \quad = \quad \lambda^{-1}\Phi^{-1}(n-1) - \frac{\lambda^{-2}\Phi^{-1}(n-1)\underline{u}(n)\underline{u}^T(n)\Phi^{-1}(n-1)}{1+\lambda^{-1}\underline{u}^T(n)\Phi^{-1}(n-1)\underline{u}(n)}$$

Denoting

$$P(n) = \Phi^{-1}(n)$$

and

$$\underline{k}(n) \quad = \quad \frac{\lambda^{-1}P(n-1)\underline{u}(n)}{1+\lambda^{-1}\underline{u}^T(n)P(n-1)\underline{u}(n)} = \text{Exercise} = P(n)\underline{u}(n)$$

we obtain

$$P(n) \quad = \quad \lambda^{-1}P(n-1) - \lambda^{-1}\underline{k}(n)\underline{u}^T(n)P(n-1)$$

□

Recursive in time solution

We are now able to derive the main time-update equation, that of $\underline{w}(n)$

$$\begin{aligned}
\underline{w}(n) &= [\Phi(n)]^{-1}\underline{\psi}(n) = P(n)\underline{\psi}(n) = P(n)(\lambda\underline{\psi}(n-1) + \underline{u}(n)d(n)) = P(n)(\lambda\Phi(n-1)\underline{w}(n-1) + \underline{u}(n)d(n)) \\
&= P(n)((\Phi(n) - \underline{u}(n)\underline{u}(n)^T)\underline{w}(n-1) + \underline{u}(n)d(n)) = \underline{w}(n-1) - P(n)\underline{u}(n)\underline{u}(n)^T\underline{w}(n-1) + P(n)\underline{u}(n)d(n) \\
&= \underline{w}(n-1) + P(n)\underline{u}(n)(d(n) - \underline{u}(n)^T\underline{w}(n-1)) = \underline{w}(n-1) + P(n)\underline{u}(n)\alpha(n) = \underline{w}(n-1) + \underline{k}(n)\alpha(n)
\end{aligned}$$

where

$$\alpha(n) = d(n) - \underline{u}(n)^T\underline{w}(n-1)$$

is the innovation process (apriori errors). Now we can collect all necessary equations to form the RLS algorithm:

$$\begin{aligned}
\underline{k}(n) &= \frac{\lambda^{-1}P(n-1)\underline{u}(n)}{1 + \lambda^{-1}\underline{u}^T(n)P(n-1)\underline{u}(n)} \\
\alpha(n) &= d(n) - \underline{u}(n)^T\underline{w}(n-1) \\
\underline{w}(n) &= \underline{w}(n-1) + \underline{k}(n)\alpha(n) \\
P(n) &= \lambda^{-1}P(n-1) - \lambda^{-1}\underline{k}(n)\underline{u}^T(n)P(n-1)
\end{aligned}$$

□

Initialization of RLS algorithm

- In RLS algorithm there are two variables involved in the recursions (those with time index $n-1$):

  $\underline{\hat{w}}(n-1)$, P_{n-1}. We must provide initial values for these variables in order to start the recursions :

  - $\underline{w}(0)$
    If we have some apriori information about the parameters $\underline{\hat{w}}$ this information will be used to initialize the algorithm.
    Otherwise, the typical initialization is

    $$\underline{w}(0) = 0$$

  - P_0

    1. Recalling the significance of $P(n)$

    $$P(n) = \Phi^{-1}(n) = [\sum_{i=i_1}^{n} \lambda^{n-i} \underline{u}(i)\underline{u}(i)^T]^{-1}$$

    the *exact initialization* of the recursions uses a small initial segment of the data $u(i_1), u(i_1+1), \ldots, u(0)$ to compute

    $$P(0) = \Phi^{-1}(0) = [\sum_{i=i_1}^{0} \lambda^{-i} \underline{u}(i)\underline{u}(i)^T]^{-1}$$

    However, it is not a simple matter to select the length of data required for ensuring invertibility of $\Phi(0)$!

Initialization of RLS algorithm

- The *approximate initialization* is commonly used, it doesn't require matrix inversion:

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

  - Consider first an intuitive explanation of the initialization. The significance

$$P(n) = \Phi^{-1}(n) \approx const. \cdot E(\underline{w}(n) - \underline{\hat{w}})(\underline{w}(n) - \underline{\hat{w}})^T$$

    can be proven. Thus, $P(n)$ is proportional to the covariance matrix of the parameters $\underline{w}(n)$. Since our knowledge of these parameters at $n = 0$ is very vague, a very high covariance matrix of the parameters is to be expected, and thus we must assign a high value to δ^{-1}.
  - When the variance of the input and the variance of the noise are known (known SNR case) Haykin's textbook provides several recommendations for δ, according to the SNR regim (low,medium, or high SNR). One can get a good choice of δ by simulating the specific environment and analyzing the transient behaviour of the estimated parameters (small δ^{-1} provides fast convergence but may involves undesirable overshoot in the estimates).
  - For large data length, the initial values assigned at $n = 0$ are not important, since they are forgotten due to exponential forgetting factor λ.

□

Summary of the RLS algorithm

Summary of the RLS algorithm

Given data $u(1), u(2), u(3), \ldots, u(N)$ and $d(1), d(2), d(3), \ldots, d(N)$

1. Initialize $\underline{w}(0) = 0, \quad P_0 = \delta^{-1} I$
2. For each time instant, $n = 1, \ldots, N$, Compute

2.1	$\underline{\pi} = \underline{u}^T(n) P(n-1)$	M^2 *flops*
2.2	$\gamma = \lambda + \underline{\pi}\, \underline{u}$	M *flops*
2.3	$\underline{k}(n) = \frac{\underline{\pi}^T}{\gamma}$	M *flops*
2.4	$\alpha(n) = d(n) - \underline{w}^T(n-1)\underline{u}(n)$	M *flops*
2.5	$\underline{w}(n) = \underline{w}(n-1) + \underline{k}(n)\alpha(n)$	M *flops*
2.5	$P' = \underline{k}(n)\underline{\pi}$	M^2 *flops*
2.5	$P(n) = \frac{1}{\lambda}(P(n-1) - P')$	M^2 *flops*

We used flop as an abbreviation for one addition (subtraction) + one multiplication (floating point operations). The overall complexity of the algorithm is $\mathcal{O}(M^2)$ operations (flops) per time iteration.

Recursion for MSE criterion

Suppose we have at moment $n-1$ the MSE criterion value

$$\mathcal{E}(\underline{w}(n-1)) \quad = \quad \sum_{i=i_1}^{n-1} \lambda^{n-1-i}(d(i) - \underline{w}(n-1)^T \underline{u}(i))^2$$

and we want to know what is the MSE of the new filter $\underline{w}(n)$

$$\mathcal{E}(\underline{w}(n)) \quad = \quad \sum_{i=i_1}^{n} \lambda^{n-i}(d(i) - \underline{w}(n)^T \underline{u}(i))^2 = \sum_{i=i_1}^{n}(d(i))^2 - \underline{w}(n)^T \psi(n)$$

One remarkable recursion holds:

$$\mathcal{E}(\underline{w}(n)) \quad = \quad \lambda \mathcal{E}(\underline{w}(n-1)) + \alpha(n)e(n)$$

where $\alpha(n)$ is the apriori error

$$\alpha(n) = d(n) - \underline{w}(n-1)^T \underline{u}(n)$$

and $e(n)$ is the aposteriori error

$$e(n) = d(n) - \underline{w}(n)^T \underline{u}(n)$$

Applications of Recursive LS filtering: ANC

1. Adaptive noise canceller

Single weight, dual-input adaptive noise canceller

The filter order is $M = 1$ thus the filter output is

$$y(n) = \underline{w}(n)^T \underline{u}(n) = w(n)u(n)$$

Denoting $P^{-1}(n) = \sigma^2(n)$, the Recursive Least Squares filtering algorithm can be rearranged as follows:

RLS

Given data

$u(1), u(2), u(3), \ldots, u(N)$ and
$d(1), d(2), d(3), \ldots, d(N)$

1. Initialize $w(0) = 0, \quad P_0 = \delta^{-1}$
2. For each time instant, $n = 1, \ldots, N$
   - 2.1 $k(n) = \dfrac{1}{\lambda\sigma(n-1)^2 + u(n)^2}\, u(n)$
   - 2.2 $\alpha(n) = d(n) - w(n-1)u(n)$
   - 2.3 $w(n) = w(n-1) + \alpha(n)k(n)$
   - 2.4 $\sigma^2(n) = \lambda\sigma(n-1)^2 + u(n)^2$

Normalized LMS

Given data

$u(1), u(2), u(3), \ldots, u(N)$ and
$d(1), d(2), d(3), \ldots, d(N)$

1. Initialize $w(0) = 0$
2. For each time instant, $n = 1, \ldots, N$
   - 2.1 $k(n) = \dfrac{1}{a + u(n)^2}\, u(n)$
   - 2.2 $\alpha(n) = d(n) - w(n-1)u(n)$
   - 2.3 $w(n) = w(n-1) + \alpha(n)k(n)$

The normalized LMS algorithm can be obtained from RLS algorithm replacing the time varying term $\lambda\sigma(n-1)^2$ with the constant a.

Applications of Recursive LS filtering: ACE

2. Adaptive Channel Equalization
Modelling the communication channel
We assume the impulse response of the channel in the form

$$h(n) = \begin{cases} \frac{1}{2}\left[1 + cos(\frac{2\pi}{W}(n-2))\right], & n = 1, 2, 3 \\ 0, & otherwise \end{cases}$$

The filter input signal will be

$$u(n) = (h * a)(n) = \sum_{k=1}^{3} h(k)a(n-k) + v(n)$$

where $\sigma_v^2 = 0.001$
Selecting the filter structure
The filter has $M = 11$ delays units (taps).

$$y(n) = \underline{w}(n)^T \underline{u}(n) = w_0(n)u(n) + w_1(n)u(n-1) + \ldots + w_{10}(n)u(n-10)$$

The channel input is delayed 7 units to provide the desired response to the equalizer.

$$d(n) = a(n-7)$$

Applications of Recursive LS filtering: ACE

Two recursive (adaptive) filtering algorithms are compared: Recursive Least Squares (RLS) and (LMS).
RLS algorithm has higher computational requirement than LMS , but behaves much better in terms of steady state MSE and transient time. For a picture of major differences between RLS and LMS, the main recursive equation are rewritten:

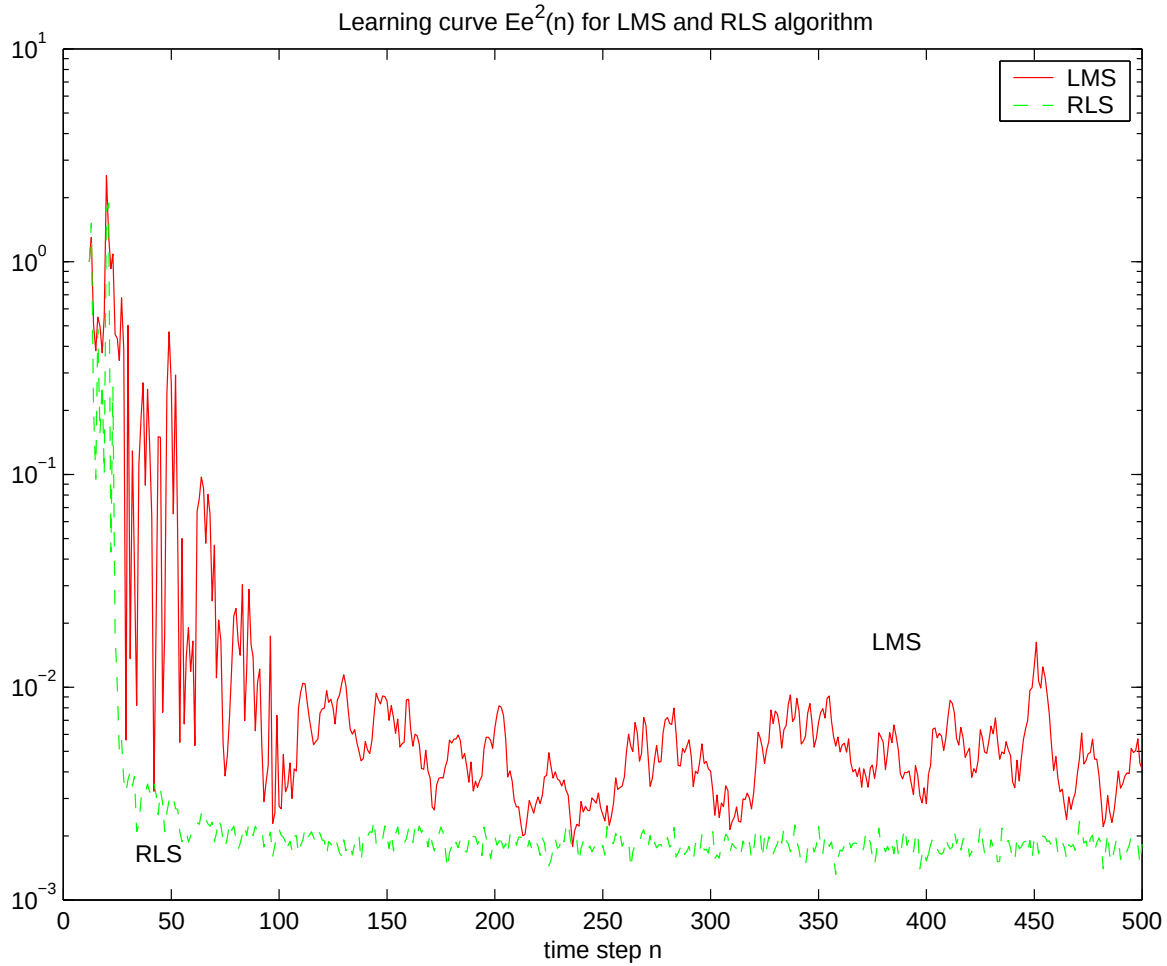
RLS algorithm

1. Initialize $\underline{w}(0) = 0, \quad P_0 = \delta^{-1} I$
2. For each time instant, $n = 1, \ldots, N$
   2.1 $\underline{w}(n) = \underline{w}(n-1) + \underline{P}(n)\underline{u}(n)(d(n) - \underline{w}^T(n-1)\underline{u}(n))$
   2.2 $P(n) = \frac{1}{\lambda + \underline{u}(n)^T P(n-1)\underline{u}(n)} (P(n-1) - P(n-1)\underline{u}(n)\underline{u}(n)^T P(n-1))$

LMS algorithm

1. Initialize $\underline{w}(0) = 0$
2. For each time instant, $n = 1, \ldots, N$
   2.1 $\underline{w}(n) = \underline{w}(n-1) + \mu \underline{u}(n)(d(n) - \underline{w}^T(n-1)\underline{u}(n))$

Applications of Recursive LS filtering: ACE

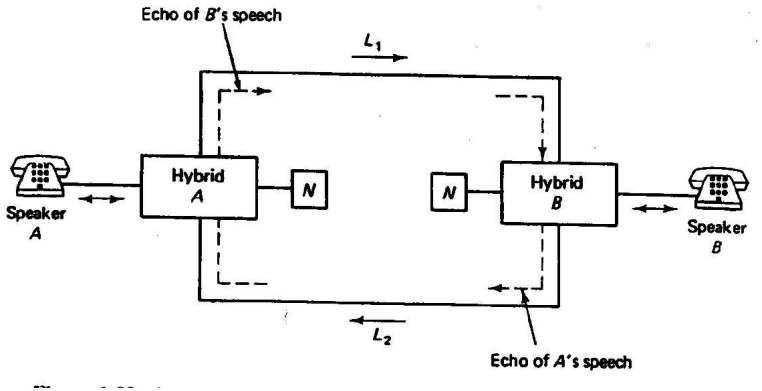


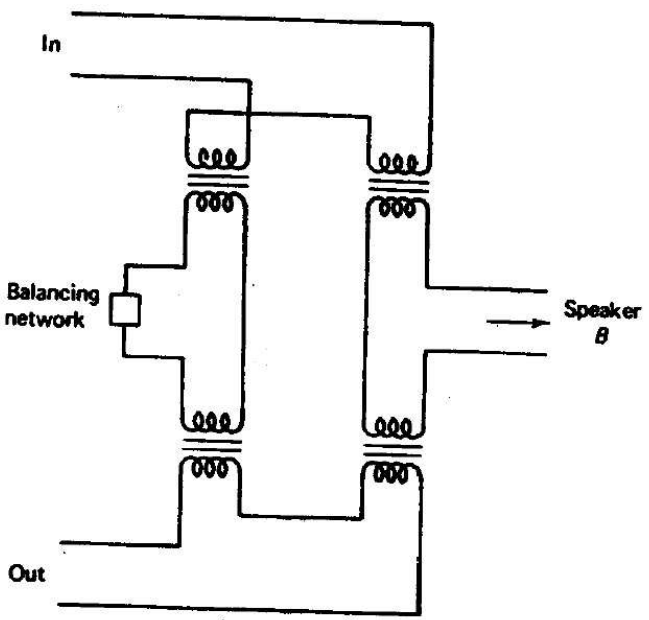
The parameters: $W = 3.1$ (i.e eigenvalue spread $\xi(R) = 11.1$). RLS: $\lambda = 1$, $\delta = 250$. LMS: $\mu = 0.075$

Echo Cancellation

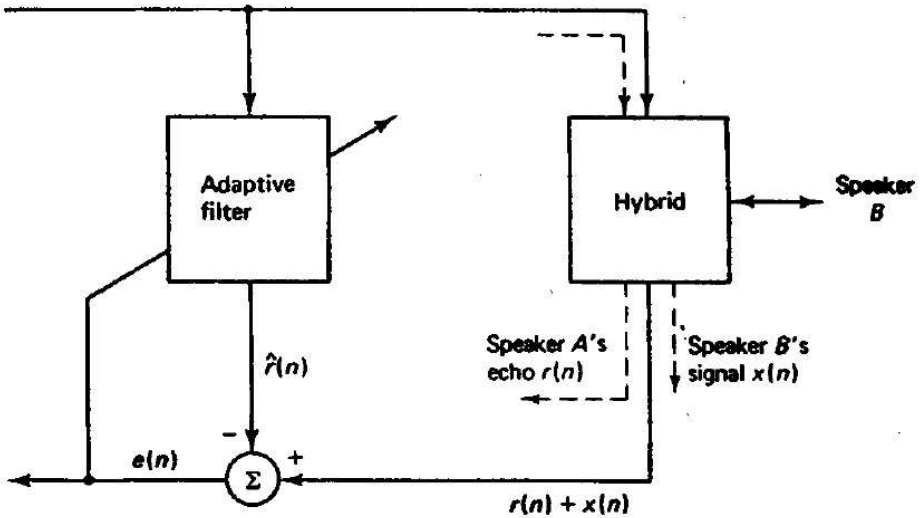
- Echo in telephone networks (PSTN = public switched telephone network);
- Subjectively echo is extremely annoying, the same as a too low volume or too high noise;
- The echo is produced at the connection of four-wire circuits with two-wire circuits;
  - The customer loop (two-wire line, maximum 60 km) is connected to the central office by four-wire lines.
  - In the central office a hybrid transformer connects to four-wire lines.
  - A hybrid transformer has three ports: IN, OUT and the two-wire line.
  - If the bridge is not balanced, the signal from IN port will appear at the OUT port (distorted and delayed).
  - Speaker A will receive its own speech, do to the leakage at Speaker B hybrid, delayed by the round trip time.
  - if the round trip time is very high (in satellite communications it may be 600msec) the echo is very annoying.
  - The larger the delay, the attenuation must be more important, to alleviate the subjective discomfort.

Echo Cancellation





Echo Cancellation



Cancelling the effect of noise: Adaptive filter

- The input signal in the adaptive filter: $u(n)$ speech signal from Speaker A
- The output signal in the adaptive filter: a replica of speech signal from Speaker A, $y(n) = \hat{r}(n)$
- The desired signal for the adaptive filter: the signal coming from OUT port of hybrid, i.e the sum $d(n) = r(n) + x(n)$ of the echo of speaker A $r(n)$ and of the speech from speaker B, $x(n)$
- The error signal in the adaptive filter is $e(n) = d(n) - y(n) = r(n) + x(n) - \hat{r}(n)$
- The adaptive filter changes its parameters such that the variance of $e(n)$ is minimized. This variance is composed of two terms:
  - The variance of $r(n) - \hat{r}(n)$, which depends on the weight vector, and therefore can be reduced.
  - The variance of $x(n)$, which don't depend on the weight vector, and therefore can't be reduced.
- the length M of adaptive filter impulse response must be larger than the assumed echo path length, e.g.
  - The sampling rate in ADPCM is $8kHz$ and then sampling period is $T_s = 0.125ms$
  - If the echo path has a delay $\tau = 30ms$, then M must be selected such that $M > 240$