Santa Clara University

ELEN 299: Directed Research

Mathematical Development of the Least Squares, Wiener Filter, LMS, and RLS Algorithms

Author:
Yevgen Solodkyy

Instructor:
Dr. Tokunbo Ogunfunmi

Contents

March 2019

1 Introduction

The purpose of this independent study was to revisit the mathematical development of the Least Squares, Wiener filter, LMS, and RLS algorithms. The orthogonality principle and uniqueness of the solution were also reviewed, as they serve the key role in finding and understanding the optimal solution.
This paper may also be viewed as a summary of select topics form *Adaptive Filtering: Fundamentals of Least Mean Squares with MATLAB* by Alexander D. Poulakis, as this text served as the primary source of information. Other referenced works will be documented in the paper, but otherwise it may be assumed that all the information comes from the above mentioned text. There are four primary applications, topologies, in adaptive signal processing: (a) system identification,(b) echo cancellation, (c) system equalization, and (d) signal prediction (Figure 1). As long as a problem can be presented in terms of the four topologies, adaptive signal processing can be applied to obtain the solution.

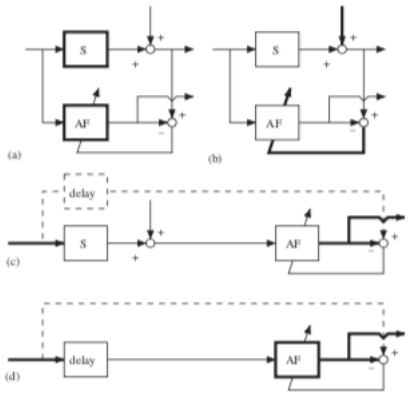

Figure 1: Fundamental Topologies

Adaptive Signal Processing is concerned with finding the optimal solution of filter weights $\mathbf{w}$(n) based on the input $\mathbf{x}$(n) and the desired reference output d(n), such that the Squared Error

$$e^2(n) = (d(n)–y(n))^2$$

is minimized. Equivalently, the objective of adaptive signal processing may be described as finding the optimal solution that corresponds to the minimum of the error surface (Figure 2) described by the cost function

$$J(\mathbf{w}) = \sum ||e(n)||^2 = \sum ||d(n)–y(n)||^2$$

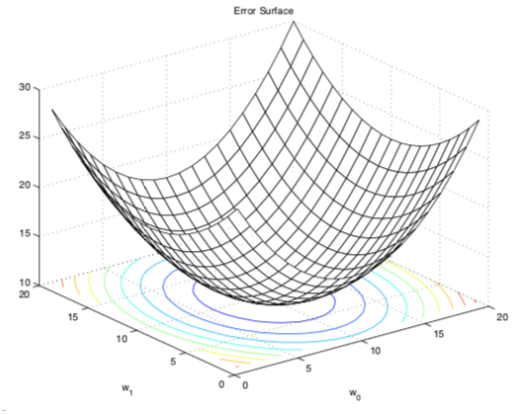

Figure 2: Error Surface

For an input vector

$$\mathbf{x}(n) = \begin{bmatrix} x_1(n) & x_2(n) & \cdots & x_M(n) \end{bmatrix}^T (M \times 1)$$

and an M-weight linear adaptive filter with a coefficients vector

$$\mathbf{w}(n) = \begin{bmatrix} w_1(n) & w_2(n) & \cdots & w_M(n) \end{bmatrix}^T (M \times 1),$$

the output of the filter is given by

$$y(n) = \mathbf{w}^T(n)\mathbf{x}(n) = \sum_{k=1}^{M} w_k(n)x_k(n).$$

For mathematical convenience, the input and output quantities may also be expressed in matrix form. Let

$$\mathbf{x}(n+1) = \begin{bmatrix} x_1(n+1) & x_2(n+1) & \cdots & x_M(n+1) \end{bmatrix}^T$$

and

$$\mathbf{x}(n+k) = \begin{bmatrix} x_1(n+k) & x_2(n+k) & \cdots & x_M(n+k) \end{bmatrix}^T.$$

For a total of N inputs, we can define an (N x M) input matrix X

$$X = \begin{bmatrix} x_1(n) & x_2(n) & \cdots & x_M(n) \\ x_1(n+1) & x_2(n+1) & \cdots & x_M(n+1) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(N-1) & x_2(N-1) & \cdots & x_M(N-1) \\ x_1(N) & x_2(N) & \cdots & x_M(N) \end{bmatrix}$$

The **X** matrix may be thought of as an aggregate of all inputs to the filter. The vector **x**(n) and matrix **X** imply an M-dimensional input. However, if the input is an ergodic process/signal, its time average is equal to its ensemble average. Therefore, we can treat the signal either as an array input or as a delayed sequence input. We may equivalently express the input to the filter as one dimensional delayed sequence with

$$\mathbf{x}(n) = \begin{bmatrix} x(M) & x(M-1) & x(M-2) & \cdots & x(1) \end{bmatrix}^T.$$

Assuming N>M, the X matrix is then defined as

$$X = \begin{bmatrix} x(M) & x(M-1) & x(M-2) & \cdots & x(1) \\ x(M+1) & x(M) & x(M-1) & \cdots & x(2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x(N-1) & x(N-2) & x(N-3) & \cdots & x(N-M) \\ x(N) & x(N-1) & x(N-2) & \cdots & x(N-M+1) \end{bmatrix}$$

Using the X matrix we can describe the output of the filter in vector form as

$$\mathbf{y} = \mathbf{X}\mathbf{w},$$

where

$$\mathbf{y}(n) = \begin{bmatrix} y(1) & y(2) & y(3) & \cdots & y(N) \end{bmatrix}^T,$$

which represents all the outputs of the filter. Then the error vector is defined as

$$\mathbf{e} = \mathbf{d}–\mathbf{y} = \mathbf{d}–\mathbf{X}\mathbf{w},$$

where

$$\mathbf{d}(n) = \begin{bmatrix} d(1) & d(2) & d(3) & \cdots & d(N) \end{bmatrix}^T,$$

and

$$\mathbf{e}(n) = \begin{bmatrix} e(1) & e(2) & e(3) & \cdots & e(N) \end{bmatrix}^T.$$

Consequently

$$J = \mathbf{e}^T\mathbf{e} = (\mathbf{d}–\mathbf{y})^T(\mathbf{d}–\mathbf{y}) = (\mathbf{d}–\mathbf{X}\mathbf{w})^T(\mathbf{d}–\mathbf{X}\mathbf{w}) = (\mathbf{d}^T–\mathbf{w}^T\mathbf{X}^T)(\mathbf{d}–\mathbf{X}\mathbf{w})$$

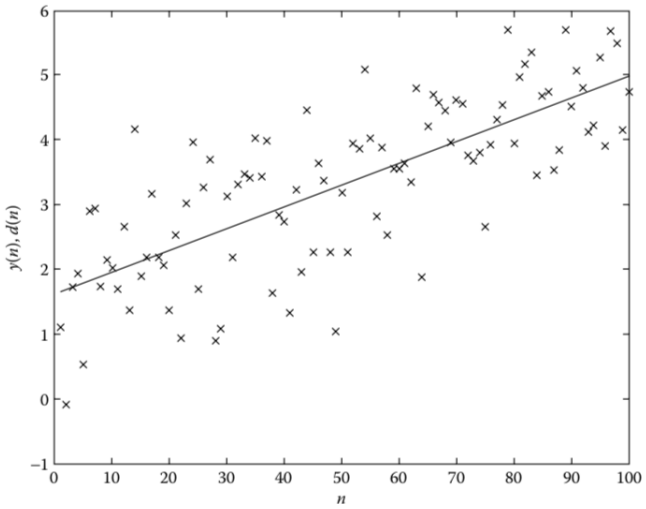

Figure 3: Least Squares

2 Least Squares

The Least Squares solution is a deterministic approach used to find the solution to an over-determined system

$$\mathbf{y} = \mathbf{X}\mathbf{w}$$

, $N > M$, i.e. where the number of samples is greater than the number of filter weights, with the aim of producing a line of best fit to the observed data(Figure 3), which is achieved by minimizing the cost function

$$J(\mathbf{w}) = \mathbf{e}^T\mathbf{e} = ||\mathbf{d} - \mathbf{y}||^2$$

where $\mathbf{y}$ is the filter output vector and $\mathbf{d}$ is the desired reference value vector. Expanding the cost function, we get

$$J(\mathbf{w}) = (\mathbf{d} - \mathbf{y})^T(\mathbf{d} - \mathbf{y}) = (\mathbf{d} - \mathbf{X}\mathbf{w})^T(\mathbf{d} - \mathbf{X}\mathbf{w})$$

$$= E_d - 2\mathbf{p}^T\mathbf{w} + \mathbf{w}^T\mathbf{R}\mathbf{w}$$

where $\mathbf{R} = \mathbf{X}^T\mathbf{X}$ is the sample correlation matrix, and $\mathbf{p} = \mathbf{d}\mathbf{X}$ is the output correlation matrix. For a Wide Sense Stationary (WSS) process $\mathbf{R} = \mathbf{R}^T$. In this paper all inputs $\mathbf{x}(n)$ are assumed to be (WSS). $E_d = \mathbf{d}^T\mathbf{d}$ is the energy of the signal.

2.1 LS Solution

The Least Squares solution may be obtained using either the Calculus approach or the Orthogonality Principle approach.

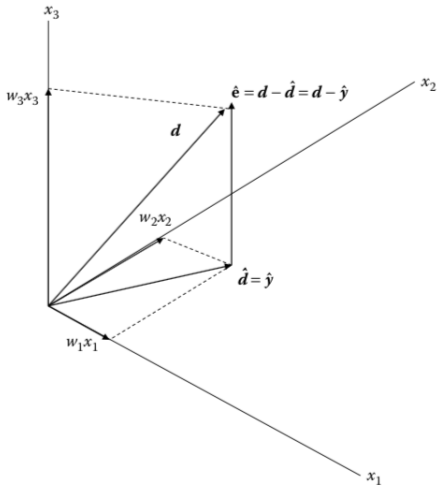

Figure 4: Orthogonality and Uniqueness

2.1.1 Calculus Approach

In the Calculus approach, we take the derivative $\frac{dJ(\mathbf{w})}{d\mathbf{w}}$ and set it to zero:

$$\frac{dJ}{d\mathbf{w}} = E_d - 2\mathbf{p}^T\mathbf{w} + \mathbf{w}^T\mathbf{R}\mathbf{w} = -2\mathbf{p}^T + 2\mathbf{w}^T\mathbf{R}$$

Hence

$$-2\mathbf{p}^T + 2\widehat{\mathbf{w}}^T\mathbf{R} = 0$$
$$\widehat{\mathbf{w}}^T\mathbf{R} = \mathbf{p}^T$$

Because $\mathbf{R} = \mathbf{R}^T$, we get

$$\mathbf{R}\widehat{\mathbf{w}} = \mathbf{p},$$

and

$$\boxed{\widehat{\mathbf{w}} = \mathbf{R}^{-1}\mathbf{p}},$$

which is the optimal solution, such that

$$J(\widehat{\mathbf{w}}) = J_{min} = E_d - 2\mathbf{p}^T\widehat{\mathbf{w}} + \widehat{\mathbf{w}}^T\mathbf{R}\widehat{\mathbf{w}} = E_d - \mathbf{p}^T\mathbf{R}\mathbf{p}.$$

2.1.2 Orthogonality Principle Approach

Alternatively, we can employ the orthogonality principle. If $\widehat{\mathbf{w}}$ is the optimal solution then, the filter output $\widehat{\mathbf{y}} = \mathbf{X}\widehat{\mathbf{w}}$ must be perpendicular to the optimal error

$$\widehat{\mathbf{e}} = \mathbf{d}–\mathbf{X}\widehat{\mathbf{w}}$$

As a result, we may describe the problem in terms of inner product

$$\widehat{\mathbf{y}} \perp \widehat{\mathbf{e}} \Leftrightarrow < \widehat{\mathbf{y}}, \widehat{\mathbf{e}} >= 0 = \widehat{\mathbf{w}}^T \mathbf{X}^T (\mathbf{d}–\mathbf{X}\widehat{\mathbf{w}}).$$

Then the minimum cost function value is given as

$$J_{min} = (\mathbf{d}–\mathbf{X}\widehat{\mathbf{w}})^T(\mathbf{d}–\mathbf{X}\widehat{\mathbf{w}}) = \mathbf{d}^T(\mathbf{d}–\mathbf{X}\widehat{\mathbf{w}}) – \widehat{\mathbf{w}}^T\mathbf{X}^T(\mathbf{d}–\mathbf{X}\widehat{\mathbf{w}}) = \mathbf{d}^T(\mathbf{d}–\mathbf{X}\widehat{\mathbf{w}}) – < \widehat{\mathbf{y}}, \widehat{\mathbf{e}} >$$

With $\widehat{\mathbf{w}}^T\mathbf{X}^T(\mathbf{d}–\mathbf{X}\widehat{\mathbf{w}}) = 0$, we get

$$\mathbf{X}^T\mathbf{X}\widehat{\mathbf{w}} = \mathbf{X}^T\mathbf{d}$$

$$\boxed{\widehat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{d} = \mathbf{R}^{-1}\mathbf{p}}$$

2.1.3 Orthogonality and Uniqueness

The equation above indicates that when the filter weights are optimal, the output of the filter $\widehat{\mathbf{y}}$ is the projection of the desired output vector $\mathbf{d}$ onto the column space of $\mathbf{X}$ (Figure 4)

$$\widehat{\mathbf{y}} = \mathbf{X}\widehat{\mathbf{w}} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{d}$$

Because there can only be one projection of vector d onto the column space of X, the optimal solution is guaranteed to be unique. Likewise the uniqueness of the optimal solution is guaranteed because the error surfaces of Least Squares problems are parabolic, which means there is only one minimum (Figure 2). Therefore, the orthogonality principle presents two interesting aspects of viewing the optimal solution: (1) The optimal solution is the projection of the desired value onto the 'sample space;' (2) a solution is guaranteed to be optimal if the error is perpendicular to the sample space.

3 Wiener Filter

The Wiener Filter is a stochastic filter that produces an estimate $y(n)$ of the desired real-valued signal $d(n)$, using linear combinations of stationary process $\mathbf{x}(n)$ with the objective of minimizing the Mean Square Error

$$J = E\{e^2(n)\}$$

. The processes $\mathbf{x}(n)$, and $d(n)$ are assumed to be real-valued, Wide Sense Stationary (WSS), and have zero mean values. Depending on how the data $\mathbf{x}(n)$, and desired signal $d(n)$ are related, the Wiener filter can be used to obtain solutions for four basic problems: Filtering, Smoothing, Prediction, and Deconvolution." [2] Given the input vector

$$\mathbf{x}(n) = \begin{bmatrix} x(n) & x(n-1) & x(n-2) & \cdots & x(n-M+1) \end{bmatrix}^T$$

and filter coefficients vector

$$\mathbf{w} = \begin{bmatrix} w_0 & w_1 & w_2 & \cdots & w_{M-1} \end{bmatrix}^T,$$

the filter output is

$$y(n) = \sum_{m=0}^{M} w_m x(n-m) = \mathbf{w}^T \mathbf{x}(n) = \mathbf{x}^T(n)\mathbf{w},$$

where M is the number of filter coefficients. Then the cost function is given as

$$\begin{aligned} J(\mathbf{w}) &= E\{e^2(n)\} = E\{[d(n) - \mathbf{w}^T\mathbf{x}(n)]^2\} \\ &= E\{[d(n) - \mathbf{w}^T\mathbf{x}(n)][d(n) - \mathbf{w}^T\mathbf{x}(n)]^T\} \\ &= E\{d^2(n) - \mathbf{w}^T\mathbf{x}(n)d(n) - d(n)\mathbf{x}^T(n)\mathbf{w} + \mathbf{w}^T\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{w}\} \\ &= E\{d^2(n)\} - 2\mathbf{w}^T E\{d(n)\mathbf{x}(n)\} + \mathbf{w}^T E\{\mathbf{x}(n)\mathbf{x}^T(n)\}\mathbf{w} \\ &= \sigma_d^2 - 2\mathbf{w}^T\mathbf{p} + \mathbf{w}^T\mathbf{R}\mathbf{w} \end{aligned}$$

$\mathbf{R} = E\{\mathbf{x}(n)\mathbf{x}^T(n)\}$ is the correlation matrix of the input data. $\mathbf{R} = \mathbf{R}^T$, because the random process x(n) is assumed to be Wide Sense Stationary. $\sigma_d^2 = E\{d(n)^2\}$ is the variance of the signal. $\mathbf{p} = E\{d(n)\mathbf{x}(n)\}$

3.1 Wiener Solution

While the LS solution is concerned with minimizing the Squared Error $J = \mathbf{e}^T\mathbf{e}$, the Wiener solution minimizes the Mean Squared Error (MSE). Hence, the two filters differ in the definition of the cost function $J(\mathbf{w})$, where for the Wiener Filter the cost function is defined as

$$J(\mathbf{w}) = E\{e^2(n)\} = E\{[d(n) - \mathbf{w}^T\mathbf{x}(n)]^2\}$$

The approach to finding the Wiener solution is analogous to that of the Least Squares approach, except that here we want to minimize the MSE.

3.1.1 Calculus Approach

Taking the derivative $\frac{dJ(\mathbf{w})}{d\mathbf{w}}$ and setting it to zero yields the optimal solution:

$$\frac{dJ(\mathbf{w})}{d\mathbf{w}} = \frac{d}{d\mathbf{w}}(\sigma_d^2 - 2\mathbf{p}^T\mathbf{w} + \mathbf{w}^T\mathbf{R}\mathbf{w}) = -2\mathbf{p}^T + 2\mathbf{w}^T\mathbf{R}$$

$$\left.\frac{dJ(\mathbf{w})}{d\mathbf{w}}\right|_{\widehat{\mathbf{w}}} = -2\mathbf{p}^T + 2\mathbf{w}^T\mathbf{R} = 0$$

$$\mathbf{R}\widehat{\mathbf{w}} = \mathbf{p}$$

$$\widehat{\mathbf{w}} = \mathbf{R}^{-1}\mathbf{p}$$

Which results in the minimum value of the cost function $J(\mathbf{w})$:

$$J(\widehat{\mathbf{w}}) = J_{min} = \sigma_d^2 - 2\mathbf{p}^T\widehat{\mathbf{w}} + \widehat{\mathbf{w}}^T\mathbf{R}\widehat{\mathbf{w}} = \sigma_d^2 - \mathbf{p}^T\mathbf{R}\mathbf{p}$$

3.1.2 Orthogonality Principle Approach

As in the case with Least Squares, since the Wiener filter is an optimal filter, we can employ the Orthogonality Principle to get the optimal solution $\widehat{\mathbf{w}}$, treating it as an inner product problem, such that

$$E\{\widehat{y}, \widehat{e}\} = 0$$

which means that the error and the optimal estimate of the signal are uncorrelated. As a result, we have

$$J_{min} = J(\widehat{\mathbf{w}}) = E\{[d(n) - \widehat{\mathbf{w}}^T\mathbf{x}(n)][d(n) - \widehat{\mathbf{w}}^T\mathbf{x}(n)]^T\}$$

$$= E\{e(n)[d^T(n) - \mathbf{x}^T(n)\widehat{\mathbf{w}}]\} = E\{(d(n) - \widehat{\mathbf{w}}^T\mathbf{x}(n))d^T(n)\} - E\{e(n)\mathbf{x}^T(n)\widehat{\mathbf{w}}\},$$

where

$$E\{e(n)\mathbf{x}^T(n)\}\widehat{\mathbf{w}} = E\{(d(n) - \widehat{\mathbf{w}}^T\mathbf{x}(n))\mathbf{x}^T(n)\}\widehat{\mathbf{w}} = 0$$

then

$$E\{d(n)\mathbf{x}^T(n)\} - E\{\widehat{\mathbf{w}}^T\mathbf{x}(n)\mathbf{x}^T(n)\} = \mathbf{p}^T - \widehat{\mathbf{w}}^T\mathbf{R} = 0$$

and

$$\boxed{\widehat{\mathbf{w}} = \mathbf{R}^{-1}\mathbf{p}}$$

4 LMS Algorithm

The LMS algorithm is a stochastic gradient descent algorithm that provides an iterative approximation to the Wiener solution. The method does away with finding the auto-correlation $\mathbf{R} = E\{\mathbf{x}(n)\mathbf{x}^T(n)\}$ and cross-correlation $\mathbf{p} = E\{d(n)\mathbf{x}^T(n)\}$ of the signal, and uses the exact values $\mathbf{x}(n)\mathbf{x}^T(n)$ and $d(n)\mathbf{x}^T(n)$, instead, to calculate the value of $\widehat{\mathbf{w}}(n)$. The iterative solution utilizes the steepest descent algorithm

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu \cdot \nabla J(\mathbf{w}), 0 < \mu < 1,$$

where μ is the learning rate. Using the Steepest Descent Algorithm, we calculate the value of $\widehat{\mathbf{w}}(n)$ as follows:

(1) Start with an initial, guessed, value of $\mathbf{w(n)}$
(2) Compute $\nabla J(\mathbf{w})|_{\mathbf{w = w(0)}}$
(3) Compute $\mathbf{w}(1) = \mathbf{w}(0) – \mu \cdot \nabla J(\mathbf{w})|_{\mathbf{w = w(0)}}$
(4) Return to step (2) and continue iterating

4.0.1 Development

With the Steepest Descent Algorithm as the starting point, we may derive the updated filter weights values

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu\mathbf{x}(n)e(n)$$

obtained via the LMS algorithm as follows:

(1) $J(\mathbf{w}) = \sigma_d^2 - 2\mathbf{w}^T\mathbf{p} + \mathbf{w}^T\mathbf{R}\mathbf{w}$

(2) $\mathbf{w}(n+1) = \mathbf{w}(n) - \mu \cdot \nabla J(\mathbf{w})$

(3) $\nabla J(\mathbf{w}) = -2\mathbf{p} + 2\mathbf{R}\widehat{\mathbf{w}}$

(4) Now approximate using instantaneous values instead of statistics as

$$\mathbf{R} = E\{\mathbf{x}(n)\mathbf{x}^T(n)\} \approx \mathbf{x}(n)\mathbf{x}^T(n)$$

$$\mathbf{p} = E\{d(n)\mathbf{x}(n)\} \approx d(n)\mathbf{x}(n)$$

(5) $\mathbf{w}(n+1) = \mathbf{w}(n) - 2\mu(\mathbf{R}\mathbf{w}(n) - \mathbf{p}) = \mathbf{w}(n) - 2\mu[\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{w}(n) - d(n)\mathbf{x}(n)]$

$$\boxed{\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu\mathbf{x}(n)e(n)},$$

where

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

is the *posteriori* estimator error. Since the LMS algorithm replaces the statistical quantities $\mathbf{R}$ and $\mathbf{p}$ of the signal with instantaneous values, it is robust and has low computational complexity, as it requires only present values of input $\mathbf{x}(n)$ and reference value $d(n)$. Because the expectation operator is not present, the filter never terminates at the Wiener solution, and the coefficients of the calculated vector $\mathbf{w}(n)$ experience sharp variations around the optimal value during the iteration process.

4.0.2 Convergence

The convergence of the LMS algorithm is subject to appropriate selection of the learning rate μ, such that $\mu < 1/\lambda_{max}$, where λ_{max} is the largest eigenvalue of $\mathbf{R}$. The convergence of the algorithms is depicted by the learning curve $J(n)$, where the cost function $J(\mathbf{w}(n))$ is plotted versus the number of iterations n of the LMS algorithm. The curve exhibits exponentially decaying behavior for a properly chosen learning rate μ. Since the convergence of the LMS algorithm is primarily determined by the chosen learning rate μ, the learning curve serves as an indicator whether the learning rate μ has been selected properly.

The trade off in selecting between higher and lower learning rates is that the higher the rate, the faster the algorithm will converge, but the final solution will have greater ripple than with a lower learning rate. In practice typically $\mu < 1/2$.

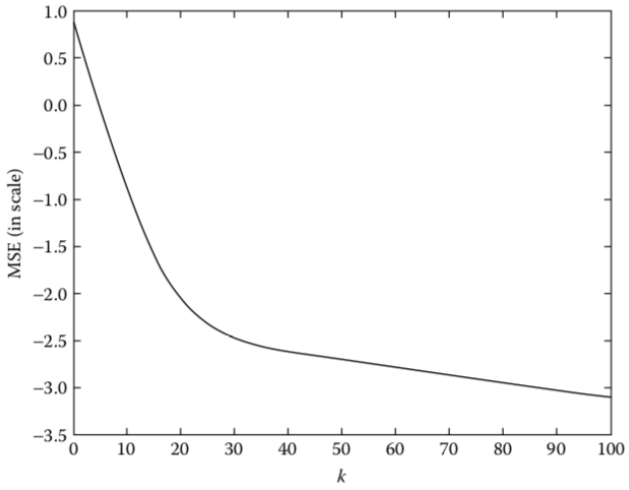

Figure 5: Learning Curve

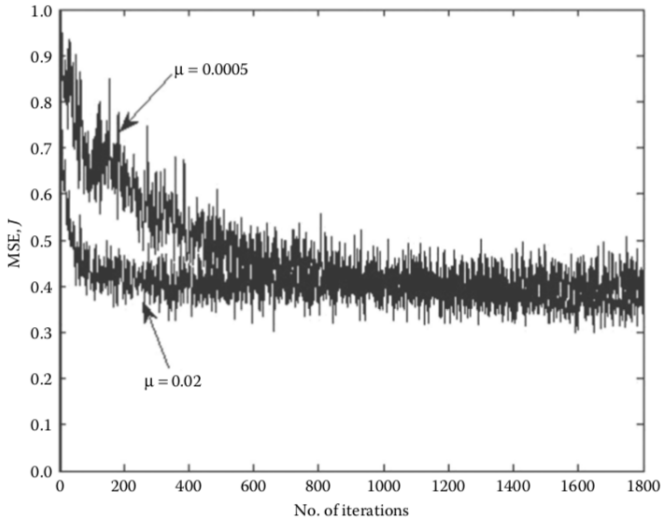

Figure 6: LMS Convergence learning rate $\mu = .02$ and $\mu = .0005$

5 RLS Algorithm

The optimal solution obtained using the Least Squares solution $\widehat{\mathbf{w}} = \mathbf{R}^{-1}\mathbf{p}$ isn't practical because it requires the knowledge of all the past values of the input signal $\mathbf{x}(n)$. The RLS algorithm uses the estimate of filter weights $\widehat{\mathbf{w}}(n-1)$ at iteration , and computes the new estimate $\widehat{\mathbf{w}}(n)$ using new incoming data. As a

result, the algorithm only requires the knowledge of the previous estimate and new input data. No knowledge of past data is needed [2].

The cost function of the RLS algorithm is of the form

$$J(n) = \sum_{k=0}^{n} \lambda^{(n-k)} e^2(k),$$

where $\lambda, 0 < \lambda < 1$, is the forgetting factor with a typical value of $\lambda = .98$. When $\lambda = 1$ we have an infinite memory system since all input data is preserved.

5.0.1 Development

The derivation of the RLS algorithm takes advantage of the matrix inversion lemma to obviate the need for calculating $\mathbf{R}^{-1}(n)$.

$$(\lambda \mathbf{A} + \mathbf{x}\mathbf{x}^T)^{-1} = \lambda^{-1}\mathbf{A}^{-1} - \frac{(\lambda^{-1}\mathbf{A}^{-1})\mathbf{x}\mathbf{x}^T(\lambda^{-1}\mathbf{A}^{-1})}{1 + \lambda^{-1}\mathbf{x}^T\mathbf{A}^{-1}\mathbf{x}}$$

Starting with the definitions of $\mathbf{R}_\lambda(n)$ and $\mathbf{p}_\lambda$(n)

$$\mathbf{R}_\lambda(n) = \mathbf{X}^T \Lambda \mathbf{X} = \sum_{k=1}^{n} \lambda^{(n-k)} \mathbf{x}(k)\mathbf{x}^T(k) (9.7.8)$$

$$\mathbf{p}_\lambda(n) = \mathbf{X}^T \Lambda d = \sum_{k=1}^{n} \lambda^{(n-k)} \mathbf{x}(k)d(k) (9.7.9)$$

Where, again, we ultimately want to calculate (9.7.7)

$$\widehat{\mathbf{w}_\lambda}(n) = \mathbf{R}_\lambda^{-1}(n)\mathbf{p}_\lambda(n)$$

Observe that

$$\mathbf{R}_\lambda(n) = \sum_{k=1}^{n} \lambda^{(n-k)} \mathbf{x}(k)\mathbf{x}^T(k) = \sum_{k=1}^{n-1} \lambda^{(n-k)} \mathbf{x}(k)\mathbf{x}^T(k) + \mathbf{x}(n)\mathbf{x}^T(n)$$

$$\mathbf{R}_\lambda(n) = \mathbf{X}^T \Lambda \mathbf{X} = \sum_{k=1}^{n-1} \lambda^{(n-k)} \mathbf{x}(k)\mathbf{x}^T(k) + \mathbf{x}(n)\mathbf{x}^T(n)$$

Similarly

$$\mathbf{p}_\lambda(n) = \sum_{k=1}^{n} \lambda^{(n-k)} \mathbf{x}(k)d(k) = \sum_{k=1}^{n-1} \lambda^{(n-k)} \mathbf{x}(k)d(k) + \mathbf{x}(n)d(n)$$

By the matrix inversion lemma

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

$$= \lambda^{-1}\mathbf{R}_\lambda^{-1}(n-1) - \frac{\lambda^{-1}\mathbf{R}_\lambda^{-1}(n-1)\mathbf{x}(n)\mathbf{x}^T(n)\lambda^{-1}\mathbf{R}_\lambda^{-1}(n-1)}{1+\lambda^{-1}\mathbf{x}^T(n)\mathbf{R}_\lambda^{-1}(n-1)\mathbf{x}(n)}$$

$$= \lambda^{-1}\mathbf{R}_\lambda^{-1}(n-1) - \frac{\lambda^{-2}\mathbf{R}_\lambda^{-1}(n-1)\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{R}_\lambda^{-1}(n-1)}{1+\lambda^{-1}\mathbf{x}^T(n)\mathbf{R}_\lambda^{-1}(n-1)\mathbf{x}(n)}.$$

Define

$$\mathbf{g}(n) = \frac{\lambda^{-1}\mathbf{R}_\lambda^{-1}(n-1)\mathbf{x}(n)}{1+\lambda^{-1}\mathbf{x}^T(n)\mathbf{R}_\lambda^{-1}(n-1)\mathbf{x}(n)}$$

Manipulating $\mathbf{g}(n)$

$$\mathbf{g}(n) + \mathbf{g}(n)\lambda^{-1}\mathbf{x}^T(n)\mathbf{R}_\lambda^{-1}(n-1)\mathbf{x}(n) = \lambda^{-1}\mathbf{R}_\lambda^{-1}(n-1)\mathbf{x}(n)$$

$$\mathbf{g}(n) = [1-\mathbf{g}(n)\mathbf{x}^T(n)]\lambda^{-1}\mathbf{R}_\lambda^{-1}(n-1)\mathbf{x}(n)$$

Expressing $\mathbf{R}_\lambda^{-1}(n)$ in terms of $\mathbf{g}(n)$

$$\mathbf{R}_\lambda^{-1}(n) = \lambda^{-1}\mathbf{R}_\lambda^{-1}(n-1) - \lambda^{-1}\mathbf{g}(n)\mathbf{x}^T(n)\mathbf{R}_\lambda^{-1}(n-1)$$

$$= \lambda^{-1}[1-\mathbf{g}(n)\mathbf{x}^T(n)]\mathbf{R}_\lambda^{-1}(n-1)$$

$$\mathbf{g}(n) = \mathbf{R}_\lambda^{-1}(n)\mathbf{x}(n)$$

Substitute to obtain

$$\widehat{\mathbf{w}}_\lambda(n) = \mathbf{R}_\lambda^{-1}(n)\mathbf{p}_\lambda(n) = \mathbf{R}_\lambda^{-1}(n)[\lambda\mathbf{p}_\lambda^{-1}(n-1) + \mathbf{x}(n)d(n)]$$

$$= \lambda\mathbf{R}_\lambda^{-1}(n)\mathbf{p}_\lambda^{-1}(n-1) + \underbrace{\mathbf{R}_\lambda^{-1}(n)\mathbf{x}(n)}_{\mathbf{g}(n)}d(n)$$

$$= \lambda[\lambda^{-1}(1-\mathbf{g}(n)\mathbf{x}^T(n))\mathbf{R}_\lambda^{-1}(n-1)]\mathbf{p}_\lambda(n-1) + \mathbf{g}(n)d(n)$$

$$= (1-\mathbf{g}(n)\mathbf{x}^T(n))]\underbrace{\mathbf{R}_\lambda^{-1}(n-1)\mathbf{p}_\lambda(n-1)}_{\widehat{\mathbf{w}}_\lambda(n-1)} + \mathbf{g}(n)d(n)$$

$$= (1-\mathbf{g}(n)\mathbf{x}^T(n))\widehat{\mathbf{w}}_\lambda(n-1) + \mathbf{g}(n)d(n)$$

$$= \widehat{\mathbf{w}}_\lambda(n-1) + \mathbf{g}(n)(d(n) - \mathbf{x}^T(n)\widehat{\mathbf{w}}_\lambda(n-1))$$

$$\boxed{\widehat{\mathbf{w}}_\lambda(n) = \widehat{\mathbf{w}}_\lambda(n-1) + \mathbf{g}(n)\widehat{e}_{n-1}(n))},$$

where

$$\widehat{e}_{n-1}(n) = d(n) - \mathbf{x}^T(n)\widehat{\mathbf{w}}_\lambda(n-1)$$

is known as the *a priori* error. It is the difference between the estimate of the desired value $d(n)$ using the previously calculated filter weight $\widehat{\mathbf{w}}_\lambda(n-1)$. Explicitly substituting

$$\mathbf{g}(n) = \frac{\lambda^{-1}\mathbf{R}_\lambda^{-1}(n-1)\mathbf{x}(n)}{1+\lambda^{-1}\mathbf{x}^T(n)\mathbf{R}_\lambda^{-1}(n-1)\mathbf{x}(n)},$$

we get

$$\widehat{\mathbf{w}}_\lambda(n) = \widehat{\mathbf{w}}_\lambda(n-1) + (\frac{\lambda^{-1}\mathbf{R}_\lambda^{-1}(n-1)\mathbf{x}(n)}{1+\lambda^{-1}\mathbf{x}^T(n)\mathbf{R}_\lambda^{-1}(n-1)\mathbf{x}(n)})(d(n) - \mathbf{x}^T(n)\widehat{\mathbf{w}}_\lambda(n-1)).$$

Then using the update function,

$$\mathbf{R}_\lambda^{-1}(n) = \lambda^{-1}\Big(\mathbf{R}_\lambda^{-1}(n-1) - \mathbf{g}(n)\big[\mathbf{x}^T(n)\mathbf{R}_\lambda^{-1}(n-1)\big]\Big)$$

$$= \lambda^{-1}\Big(\mathbf{R}_\lambda^{-1}(n-1) - \big[\frac{\lambda^{-1}\mathbf{R}_\lambda^{-1}(n-1)\mathbf{x}(n)}{1+\lambda^{-1}\mathbf{x}^T(n)\mathbf{R}_\lambda^{-1}(n-1)\mathbf{x}(n)}\big]\big[\mathbf{x}^T(n)\mathbf{R}_\lambda^{-1}(n-1)\big]\Big),$$

one can be reassured that the RLS algorithm doesn't require the computation of $\mathbf{R}_\lambda^{-1}(n)$.

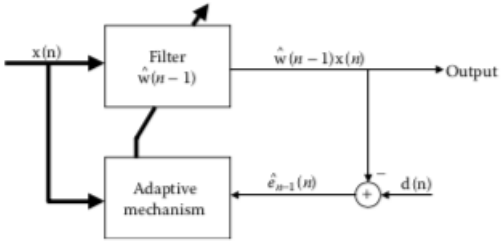

Figure 7: RLS Algorithm Block Diagram

5.0.2 Initialization Concerns

For $n < M$, $\mathbf{R}_\lambda(n)$ is singular, as it isn't full rank. Therefore to remedy the issue the definition is modified to

$$\mathbf{R}_\lambda(n) = \mathbf{X}^T\Lambda\mathbf{X} + \delta\lambda^n\mathbf{I}$$

with a suggested value of $\delta < 0.01\sigma_x^2$, then

$$\mathbf{R}_\lambda(0) = \delta\mathbf{I},$$

Setting $\mathbf{R}_\lambda(0) = \delta\mathbf{I}$ ensures that $\mathbf{R}_\lambda(n)$ is non-singular at all times, with the first estimate of $\widehat{\mathbf{w}}_\lambda(n)$ being

$$\widehat{\mathbf{w}}_\lambda(1) = (\frac{\lambda^{-1}(\delta\mathbf{I})\mathbf{x}(n)}{1+\lambda^{-1}\mathbf{x}^T(n)(\delta\mathbf{I})\mathbf{x}(n)})d(n).$$

5.0.3 Implementation Concerns

Although the RLS algorithm is always guaranteed to converge because it is an optimal filter, in practice it may suffer from accumulation of round off errors in calculating

$$\mathbf{R}_{\lambda}^{-1}(n) = \lambda^{-1}\Big(\mathbf{R}_{\lambda}^{-1}(n-1) - \mathbf{g}(n)\big[\mathbf{x}^{T}(n)\mathbf{R}_{\lambda}^{-1}(n-1)\big]\Big),$$

which makes $\mathbf{R}_{\lambda}^{-1}(n)$ non-symmetric. As a result, since by definition $\mathbf{R}_{\lambda}^{-1}(n)$ is supposed to be symmetric, only either the lower or upper triangular part of $\mathbf{R}_{\lambda}^{-1}(n)$ is calculated, and then copied to the other triangular part of the matrix. This alleviates the computational error issue and results in significant computational savings[3].

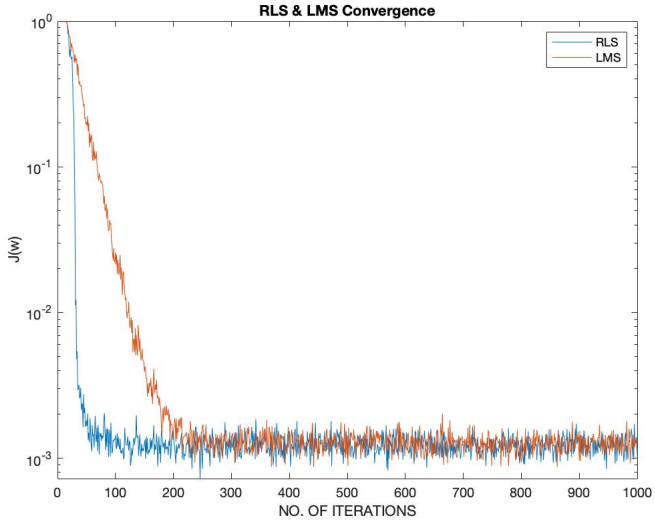

Figure 8: Convergence RLS vs LMS

6 Conclusion

The intent of the paper is to review the mathematical development of the Least Squares, Wiener filter, LMS and RLS algorithms, with the focus on gaining an intuitive view of what the algorithms are, and how they are derived. As a result, topics such as misadjustment, LMS variations, or fast RLS algorithms were purposely omitted. We start with Least Squares and Wiener filter and progress toward LMS and RLS algorithms as practical, recursive implementation of each. The RLS algorithm converges much faster than the LMS algorithm, but it is not as robust. It doesn't perform well for noisy signals. Computational intensity

is the primary disadvantage of the RLS algorithm. Unlike the RLS algorithm, the LMS algorithm is a stochastic filter, so it will have stochastic ripple around the calculated optimal value of $\mathbf{w}$. It is robust and performs well for signals with high noise. The Least Squares, Wiener Filter, and RLS Algorithm, are all exact/optimal solutions, while the LMS algorithm is an iterative approximation to the Wiener filter. As such, it is a compromise that sacrifices accuracy for implementation efficiency and robustness.

References

[1] Adaptive Filtering: Fundamentals of Least Mean Squares with MATLAB®, Alexander D. Poularikas

[2] ADAPTIVE FILTERING PRIMER with MATLAB®, Alexander D. Poularikas, Zayed M. Ramadan

[3] ADAPTIVE FILTERS THEORY AND APPLICATIONS Second Edition, Behrouz Farhang-Boroujeny

[4] The Kalman Filter in the context of adaptive Filter theory, Dani Lippuner; and George S. Moschytz

Figures: Figure 1: The Kalman Filter in the context of adaptive Filter theory, Dani Lippuner; and George S. Moschytz

Figure 2: Open Source

Figure 3-6: Adaptive Filtering: Fundamentals of Least Mean Squares with MATLAB®, Alexander D. Poularikas Figure 7: ADAPTIVE FILTERS THEORY AND APPLICATIONS Second Edition, Behrouz Farhang-Boroujeny