Lecture Notes in Adaptive Filters

Third Edition

Jesper Kjær Nielsen
jkn@create.aau.dk
Aalborg University

Søren Holdt Jensen
shj@es.aau.dk
Aalborg University

Last revised: July 5, 2019

Nielsen, Jesper Kjær and Jensen, Søren Holdt
Lecture Notes in Adaptive Filters

1. edition: June 7, 2011
2. edition: September 19, 2012
3. edition: July 4, 2019

Audio Analysis Lab, CREATE
Aalborg University
Rendsburggade 14
DK-9000 Aalborg
Denmark

These lecture notes have been printed with Computer Modern 10pt and been typeset using LaTeX 2ε on a computer running the GNU/Linux operating system. All of the figures have been created using GNUPLOT, PGF and the macro packages Ti*k*Z and PGFPLOTS. Simulations have been run in MATLAB$^{\text{TM}}$.

Contents

Preface

```
\*
 * Your warranty is now void.
 *
 * By installing this knowledge onto your brain, you accept that we, the
 * authors of the present lecture notes, are not responsible for potential
 * confusion, wasted time, failed exams, or misguided missiles.
 *
 * You will most likely encounter unclear statements and remarks, poor English
 * usage, and errors when you read the present notes. Therefore, use these
 * notes at your own risk, and please help us correct these problems by giving
 * us some feedback.
 *\
```

The present lecture notes were written for the annual course on adaptive filters at Aalborg University. The content of the course is now a part of the annual course called *Array and Sensor Signal Processing*. The notes are written for the lecturer, but they may also be useful to the student as a supplement to his/her favourite textbook. Consequently, the notes are very concise and contain only what we believe to be the basics of adaptive filtering. Moreover, we have also made some important simplifications.

1. We use real-valued numbers and not complex-valued numbers. Although the latter is more general, it is less confusing and leads to fewer errors when real-valued numbers are used.
2. We only consider FIR adaptive filters.
3. The signals have zero mean. This is a standard assumption used in most textbooks.

Each of the lectures contains an amount of material suited for a lecture lasting for approximately 90 minutes. The appendices contain a summary and some supplementary material.

These lecture notes are always work in progress. Therefore, if you have found an error, have a suggestion for a better statement and/or explanation, or just want to give us some feedback, then do not hesitate to contact us.

Lecture 1

Introduction, Wiener-Hopf Equations, and Normal Equations

Review of the Basics

Linear Algebra

Notation

An N-dimensional vector is written as

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & \cdots & x_N \end{bmatrix}^T \tag{1.1}$$

where $(\cdot)^T$ denotes the transpose. An $N \times M$-dimensional matrix is written as

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & \cdots & a_{1M} \\ \vdots & \ddots & \vdots \\ a_{N1} & \cdots & a_{NM} \end{bmatrix} . \tag{1.2}$$

Matrix Properties

Let $\boldsymbol{A}$ be a square matrix. We then have that

$$\boldsymbol{A} \text{ is orthogonal} \iff \boldsymbol{A}^T = \boldsymbol{A}^{-1} \iff \boldsymbol{I} = \boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{A}^T\boldsymbol{A} \tag{1.3}$$

$$\boldsymbol{A} \text{ is symmetric} \iff \boldsymbol{A}^T = \boldsymbol{A} \tag{1.4}$$

$$\boldsymbol{A} \text{ is skew-symmetric} \iff \boldsymbol{A}^T = -\boldsymbol{A} . \tag{1.5}$$

Let $Q(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x}$ where $\boldsymbol{A}$ is a square and symmetric matrix. We then have that

$$\begin{aligned}
&\boldsymbol{A} \text{ is positive definite (p.d.)} && \iff Q(\boldsymbol{x}) > 0 && \forall\ \boldsymbol{x} \neq 0 \\
&\boldsymbol{A} \text{ is positive semidefinite (p.s.d.)} && \iff Q(\boldsymbol{x}) \geq 0 && \forall\ \boldsymbol{x} \\
&\boldsymbol{A} \text{ is negative definite (n.d.)} && \iff Q(\boldsymbol{x}) < 0 && \forall\ \boldsymbol{x} \neq 0 \\
&\boldsymbol{A} \text{ is negative semidefinite (n.s.d.)} && \iff Q(\boldsymbol{x}) \leq 0 && \forall\ \boldsymbol{x} \\
&\boldsymbol{A} \text{ is indefinite otherwise.}
\end{aligned}$$

The interested reader can find more on linear algebra in for example [1] and [2].

Optimisation

Gradiant and Hessian

Let $f(\boldsymbol{x})$ be a scalar function with continuous second-order partial derivatives. The gradient vector of $f(\boldsymbol{x})$ is defined as

$$\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{\nabla} f(\boldsymbol{x}) = \frac{\partial f}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \cdots & \frac{\partial f}{\partial x_N} \end{bmatrix}^T . \tag{1.6}$$

The Hessian matrix of $f(\boldsymbol{x})$ is symmetric and defined as

$$\begin{aligned}
\boldsymbol{H}(\boldsymbol{x}) = \boldsymbol{\nabla}[\boldsymbol{g}^T(\boldsymbol{x})] = \boldsymbol{\nabla}[\boldsymbol{\nabla}^T f(\boldsymbol{x})] = \frac{\partial^2 f}{\partial \boldsymbol{x} \partial \boldsymbol{x}^T} \\
= \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_N \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_N^2} \end{bmatrix} .
\end{aligned} \tag{1.7}$$

When $f(\boldsymbol{x})$ consists of a constant term, we have that

$$f(\boldsymbol{x}) = \boldsymbol{c} \implies \begin{cases} \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0} \\ \boldsymbol{H}(\boldsymbol{x}) = \boldsymbol{0} \end{cases} .$$

When $f(\boldsymbol{x})$ consists of a linear term, we have that

$$f(\boldsymbol{x}) = \boldsymbol{v}^T \boldsymbol{x} = \boldsymbol{x}^T \boldsymbol{v} \implies \begin{cases} \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{v} \\ \boldsymbol{H}(\boldsymbol{x}) = \boldsymbol{0} \end{cases} .$$

When $f(\boldsymbol{x})$ consists of a quadratic term, we have that

$$f(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} \implies \begin{cases} \boldsymbol{g}(\boldsymbol{x}) = (\boldsymbol{A} + \boldsymbol{A}^T)\boldsymbol{x} = 2\boldsymbol{A}\boldsymbol{x} \\ \boldsymbol{H}(\boldsymbol{x}) = \boldsymbol{A} + \boldsymbol{A}^T = 2\boldsymbol{A} \end{cases}$$

where the last equality for the gradient and the Hessian holds if and only if $\boldsymbol{A}$ is symmetric.

Unconstrained Optimisation

We can solve an unconstrained optimisation problem by following a five step recipe.

1. Construct the cost function $f(\boldsymbol{x})$.
2. Find the gradient $\boldsymbol{g}(\boldsymbol{x}) = \frac{\partial f}{\partial \boldsymbol{x}}$.
3. Solve $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0}$ for $\boldsymbol{x}$. The solutions $\{\boldsymbol{x}_i\}$ are called critical points.
4. Find the Hessian $\boldsymbol{H}(\boldsymbol{x}) = \frac{\partial^2 f}{\partial \boldsymbol{x} \partial \boldsymbol{x}^T}$ and compute it for all the critical points. If

$$\begin{aligned}
&\boldsymbol{H}(\boldsymbol{x}_i) \text{ is p.d} && \implies && f(\boldsymbol{x}_i) \text{ is a local minimum} \\
&\boldsymbol{H}(\boldsymbol{x}_i) \text{ is n.d} && \implies && f(\boldsymbol{x}_i) \text{ is a local maximum} \\
&\boldsymbol{H}(\boldsymbol{x}_i) \text{ is indefinite} && \iff && f(\boldsymbol{x}_i) \text{ is a saddle point} \\
&\boldsymbol{H}(\boldsymbol{x}_i) \text{ is p.s.d or n.s.d} && \iff && \text{further analysis is necessary [3, p. 41]}
\end{aligned}$$

5. Find the global minimiser/maximiser by evaluating $f(\boldsymbol{x}_i)$ for every critical point.

If $f(\boldsymbol{x})$ is a convex[1] function, we have the following useful fact [3, p. 57]

$$f(\boldsymbol{x}) \text{ is a convex function} \iff \boldsymbol{H}(\boldsymbol{x}) \text{ is p.d. } \forall\ \boldsymbol{x}\ .$$

Thus, there is only one critical point corresponding to a minimum if and only if $f(\boldsymbol{x})$ is a convex function.

The interested reader can find more on optimisation in for example [3] and [4].

Stochastic Processes

Let $X(n)$ for $n = n_0, n_0+1, \ldots, n_0+N-1$ be a stochastic process with joint probability density function (pdf) $p_X(x(n_0), x(n_0+1), \ldots, x(n_0+N-1))$.

Mean, Covariance and Correlation

The mean, covariance and correlations sequences are defined as

$$\text{Mean sequence:} \quad \mu_X(n) = E[X(n)] = \int x(n) p_X(x(n)) dx(n) \tag{1.8}$$

$$\begin{aligned}
\text{Covariance sequence:} \quad c_X(n, n+k) &= \operatorname{cov}(X(n), X(n+k)) \\
&= E[(X(n) - \mu_X(n))(X(n+k) - \mu_X(n+k))] \qquad (1.9)
\end{aligned}$$

$$\text{Correlation sequence:} \quad r_X(n, n+k) = E[X(n)X(n+k)]\ , \tag{1.10}$$

and they are related by

$$c_X(n, n+k) = r_X(n, n+k) - \mu_X(n)\mu_X(n+k)\ . \tag{1.11}$$

For a finite number of observation, we may define the mean vector

$$\boldsymbol{\mu}_X = \begin{bmatrix} \mu_X(n_0) & \mu_X(n_0+1) & \cdots & \mu_X(n_0+N-1) \end{bmatrix}^T\ , \tag{1.12}$$

the covariance matrix

$$\boldsymbol{C}_X = \begin{bmatrix} c_X(n_0, n_0) & \cdots & c_X(n_0, n_0+N-1) \\ \vdots & \ddots & \vdots \\ c_X(n_0+N-1, n_0) & \cdots & c_X(n_0+N-1, n_0+N-1) \end{bmatrix}\ , \tag{1.13}$$

[1] If we should be completely accurate, we should distinguish between weakly and strictly convex functions. However, in this note, we only use convexity in the strict sense.

and the correlation matrix

$$\boldsymbol{R}_X = \begin{bmatrix} r_X(n_0, n_0) & \cdots & r_X(n_0, n_0+N-1) \\ \vdots & \ddots & \vdots \\ r_X(n_0+N-1, n_0) & \cdots & r_X(n_0+N-1, n_0+N-1) \end{bmatrix} . \tag{1.14}$$

They are related by

$$\boldsymbol{C}_X = \boldsymbol{R}_X - \boldsymbol{\mu}_X \boldsymbol{\mu}_X^T . \tag{1.15}$$

$\boldsymbol{C}_X$ and $\boldsymbol{R}_X$ are symmetric and p.s.d. They are p.d. if $X(n)$ is *not* perfectly predictable.

Stationarity and Wide Sense Stationarity (WSS)

$$\begin{aligned} &X(n) \text{ is stationary} \iff p_X(x(n_0+m), x(n_0+1+m), \ldots, x(n_0+N-1+m)) \\ &\quad \Downarrow \qquad\qquad = p_X(x(n_0), x(n_0+1), \ldots, x(n_0+N-1)) \quad \forall\, n_0, m, N \\ &X(n) \text{ is WSS} \iff \begin{cases} \mu_X(n) = \mu_X & \text{(a constant)} \\ r_X(n, n+k) = r_X(0,k) & \text{(a function of } k \text{ not } n) \end{cases} \end{aligned}$$

Since $r_X(0,k)$ does not depend on n, it is often written as $r_X(k)$. A Venn diagram for all stochastic processes is shown in Fig. 1.1.

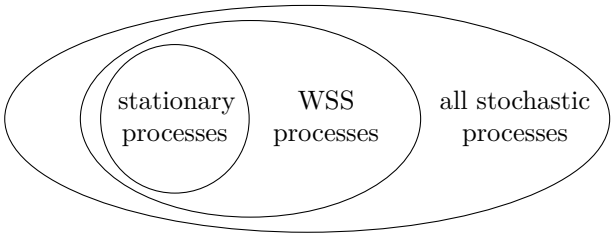

Figure 1.1: A Venn diagram for all stochastic processes.

Let $Y(n)$ be another stochastic process.

$$X(n) \text{ and } Y(n) \text{ are jointly WSS} \iff \begin{cases} X(n) \text{ is WSS} \\ Y(n) \text{ is WSS} \\ r_{X,Y}(n, n+k) = r_{X,Y}(0,k) \end{cases}$$

where $r_{X,Y}(n, n+k)$ is the cross-correlation sequence.

Estimation of Statistics

Often, we wish to estimate the statistics of $X(n)$ from a single realisation $x(n)$. If $X(n)$ is (wide sense) stationary and *ergodic*[2] in some sense (for example, in mean, in correlation, in power, or in distribution), we may estimate some or all of its statistics from a single realisation $x(n)$. Examples are the unbiased estimators of the mean

$$\hat{\mu}_X = \frac{1}{N} \sum_{n=n_0}^{n_0+N-1} x(n) \tag{1.16}$$

[2]Ergodicity basically means that we can infer something about the statistics of a stochastic process from a single realisation of it.

and the correlation sequence

$$\hat{r}_X(k) = \frac{1}{N-k} \sum_{n=n_0}^{n_0+N-1-k} x(n)x(n+k) \ . \tag{1.17}$$

The interested reader can find more on stochastic processes in for example [5] and [6].

Block Diagram of Adaptive Filtering

The adaptive filtering problem is shown in Fig. 1.2. From the figure, we have that

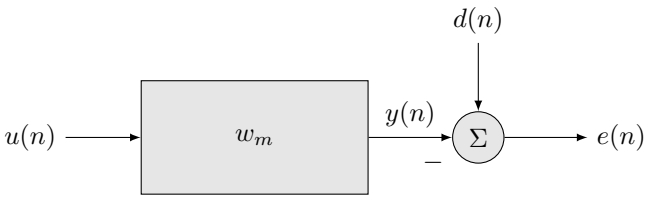

Figure 1.2: Block Diagram of Adaptive Filtering in a WSS environment.

$u(n)$: zero-mean, WSS input signal

w_m: M-tap FIR-filter with impulse response $w_0, w_1, \ldots, w_{M-1}$

$y(n)$: output signal given by

$$y(n) = \sum_{m=0}^{M-1} w_m u(n-m) \tag{1.18}$$

$d(n)$: zero-mean, WSS desired signal

$e(n)$: error signal

Moreover, $u(n)$ and $d(n)$ are assumed to be jointly WSS.

Mean-Squared Error and Squared Error cost functions

Define

$$\boldsymbol{w} = \begin{bmatrix} w_0 & w_1 & \cdots & w_{M-1} \end{bmatrix}^T \tag{1.19}$$

$$\boldsymbol{u}(n) = \begin{bmatrix} u(n) & u(n-1) & \cdots & u(n-M+1) \end{bmatrix}^T \tag{1.20}$$

$$\boldsymbol{R}_u = E[\boldsymbol{u}(n)\boldsymbol{u}^T(n)] \tag{1.21}$$

$$\boldsymbol{r}_{ud} = E[\boldsymbol{u}(n)d(n)] \ . \tag{1.22}$$

Here, $\boldsymbol{R}_u$ is the correlation matrix of the input signal vector $\boldsymbol{u}(n)$, and $\boldsymbol{r}_{ud}$ is the cross-correlation vector between the input signal vector $\boldsymbol{u}(n)$ and the desired signal $d(n)$.

For $n = n_0, n_0+1, \ldots, n_0+K-1$ with $K \geq M$, define

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{u}(n_0) & \boldsymbol{u}(n_0+1) & \cdots & \boldsymbol{u}(n_0+K-1) \end{bmatrix}^T \tag{1.23}$$

$$\boldsymbol{d} = \begin{bmatrix} d(n_0) & d(n_0+1) & \cdots & d(n_0+K-1) \end{bmatrix}^T \tag{1.24}$$

$$\boldsymbol{e} = \begin{bmatrix} e(n_0) & e(n_0+1) & \cdots & e(n_0+K-1) \end{bmatrix}^T . \tag{1.25}$$

Then

$$e(n) = d(n) - \boldsymbol{u}^T(n)\boldsymbol{w} \quad \Longleftrightarrow \quad d(n) = \boldsymbol{u}^T(n)\boldsymbol{w} + e(n) \tag{1.26}$$

and

$$\boldsymbol{e} = \boldsymbol{d} - \boldsymbol{A}\boldsymbol{w} \quad \Longleftrightarrow \quad \boldsymbol{d} = \boldsymbol{A}\boldsymbol{w} + \boldsymbol{e} . \tag{1.27}$$

We observe the input signal $u(n)$ and the desired signal $d(n)$, and we wish to find the filter vector $\boldsymbol{w}$ which minimises the error in some sense. Two popular choices for the cost function of this problem are

$$\text{Mean-squared error:} \qquad J_1(\boldsymbol{w}) = E[e(n)^2] \qquad \text{(statistics is known)} \tag{1.28}$$

$$\text{Squared error:} \qquad J_2(\boldsymbol{w}) = \sum_{n=n_0}^{n_0+K-1} e^2(n) = \boldsymbol{e}^T\boldsymbol{e} \qquad \text{(statistics is unknown)} \tag{1.29}$$

If $\text{rank}(\boldsymbol{A}) > M$, Eq. (1.27) constitutes an overdetermined system of equations which do not have an exact solution that makes $J_2(\boldsymbol{w}) = 0$. The problem of minimising the squared error is often referred to as *the method of least-squares*, and the argument minimising the squared error is referred to as *the least-squares solution*. Moreover, the squared error divided by the number of elements in the error vector $\boldsymbol{e}$ can be seen as an unbiased estimate of the mean-squared error when $u(n)$ and $d(n)$ are assumed to be jointly WSS. That is,

$$E\left[\frac{1}{K}J_2(\boldsymbol{w})\right] = J_1(\boldsymbol{w}) . \tag{1.30}$$

The Wiener-Hopf Equations

We want to solve the following unconstrained optimisation problem

$$\boldsymbol{w}_o = \arg\min_{\boldsymbol{w}} J_1(\boldsymbol{w}) \tag{1.31}$$

where $\boldsymbol{w}_o$ is the vector containing the optimal filter coefficients. We use the five step recipe in Sec. 1.1.2 to solve the optimisation problem.

1. Construct the cost function

$$\begin{aligned} J_1(\boldsymbol{w}) &= E[e(n)^2] = E[(d(n) - \boldsymbol{u}^T(n)\boldsymbol{w})^2] = E[(d(n) - \boldsymbol{u}^T(n)\boldsymbol{w})^T(d(n) - \boldsymbol{u}^T(n)\boldsymbol{w})] \\ &= E[d(n)^2] + E[\boldsymbol{w}^T\boldsymbol{u}(n)\boldsymbol{u}^T(n)\boldsymbol{w}] - E[d(n)\boldsymbol{u}^T(n)\boldsymbol{w}] - E[\boldsymbol{w}^T\boldsymbol{u}(n)d(n)] \\ &= E[d(n)^2] + \boldsymbol{w}^T E[\boldsymbol{u}(n)\boldsymbol{u}^T(n)]\boldsymbol{w} - 2\boldsymbol{w}^T E[\boldsymbol{u}(n)d(n)] \\ &= \sigma_d^2 + \boldsymbol{w}^T\boldsymbol{R}_u\boldsymbol{w} - 2\boldsymbol{w}^T\boldsymbol{r}_{ud} \quad \text{(quadratic cost function)} \end{aligned} \tag{1.32}$$

2. Find the gradient

$$\boldsymbol{g}(\boldsymbol{w}) = (\boldsymbol{R}_u + \boldsymbol{R}_u^T)\boldsymbol{w} - 2\boldsymbol{r}_{ud} = 2\boldsymbol{R}_u\boldsymbol{w} - 2\boldsymbol{r}_{ud} \tag{1.33}$$

3. Solve $\boldsymbol{g}(\boldsymbol{w}) = \boldsymbol{0}$ for $\boldsymbol{w}$

$$\boldsymbol{g}(\boldsymbol{w}) = 2\boldsymbol{R}_u\boldsymbol{w} - 2\boldsymbol{r}_{ud} = \boldsymbol{0}$$

$$\Updownarrow$$

$$\boxed{\boldsymbol{R}_u\boldsymbol{w} = \boldsymbol{r}_{ud}} \qquad \text{(Wiener-Hopf Equations)} \tag{1.34}$$

$$\Updownarrow$$

$$\boldsymbol{w} = \boldsymbol{R}_u^{-1}\boldsymbol{r}_{ud} \qquad \text{(If } \boldsymbol{R}_u \text{ is invertible)} \tag{1.35}$$

4. Find the Hessian

$$\boldsymbol{H}(\boldsymbol{w}) = 2\boldsymbol{R}_u \tag{1.36}$$

which is p.d. for all $\boldsymbol{w}$ if $u(n)$ is not perfectly predictable.

5. This implies that

- $J_1(\boldsymbol{w})$ is a convex function,
- $\boldsymbol{R}_u$ is invertible,
- $\boldsymbol{w}_o = \boldsymbol{R}_u^{-1}\boldsymbol{r}_{ud}$ is the global minimiser, and
- $J_1(\boldsymbol{w}_o) = \sigma_d^2 - \boldsymbol{r}_{ud}^T\boldsymbol{R}_u^{-1}\boldsymbol{r}_{ud}$.

The solution $\boldsymbol{w}_o$ is often referred to as the least-mean-squares solution.

Principle of Orthogonality

The Wiener-Hopf equations can also be derived from the principle of orthogonality. For $m = 0, 1, \ldots, M-1$, the principle of orthogonality states

$$0 = E[u(n-m)e_o(n)] \iff 0 = E[y_o(n)e_o(n)] \iff \boldsymbol{w}_o = \boldsymbol{R}_u^{-1}\boldsymbol{r}_{ud} \ . \tag{1.37}$$

where $e_o(n)$ and $y_o(n)$ are the error and the filter output, respectively, when the filter is optimised in the mean-squared sense.

The Modified Yule-Walker Equations

The Wiener-Hopf equations are closely related to the Modified Yule-Walker equations. To see this, let $d(n) = u(n+1)$. Then

$$\boldsymbol{r}_{ud} = E[\boldsymbol{u}(n)d(n)] = E[\boldsymbol{u}(n)u(n+1)] = \boldsymbol{p}_u \tag{1.38}$$

which implies that

$$\boldsymbol{R}_u\boldsymbol{w} = \boldsymbol{p}_u \qquad \text{(Modified Yule-Walker Equations)} \ . \tag{1.39}$$

The Normal Equations

We want to solve the following unconstrained optimisation problem

$$\boldsymbol{w}_o = \arg\min_{\boldsymbol{w}} J_2(\boldsymbol{w}) \tag{1.40}$$

where $\boldsymbol{w}_o$ is the vector containing the optimal filter coefficients. We use the five step recipe in Sec. 1.1.2 to solve the optimisation problem.

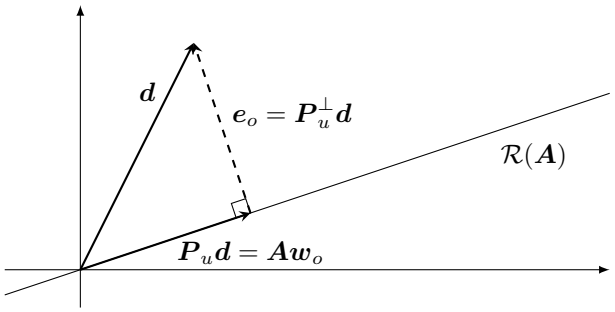

Figure 1.3: The principle of orthogonality for the squared error cost function.

1. Construct the cost function

$$\begin{aligned} J_2(\boldsymbol{w}) &= \boldsymbol{e}^T\boldsymbol{e} = (\boldsymbol{d}-\boldsymbol{A}\boldsymbol{w})^T(\boldsymbol{d}-\boldsymbol{A}\boldsymbol{w}) \\ &= \boldsymbol{d}^T\boldsymbol{d} + \boldsymbol{w}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{w} - 2\boldsymbol{w}^T\boldsymbol{A}^T\boldsymbol{d} \quad \text{(quadratic cost function)} \end{aligned} \tag{1.41}$$

2. Find the gradient

$$\boldsymbol{g}(\boldsymbol{w}) = (\boldsymbol{A}^T\boldsymbol{A} + \boldsymbol{A}^T\boldsymbol{A})\boldsymbol{w} - 2\boldsymbol{A}^T\boldsymbol{d} = 2\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{w} - 2\boldsymbol{A}^T\boldsymbol{d} \tag{1.42}$$

3. Solve $\boldsymbol{g}(\boldsymbol{w}) = \boldsymbol{0}$ for $\boldsymbol{w}$

$$\boldsymbol{g}(\boldsymbol{w}) = 2\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{w} - 2\boldsymbol{A}^T\boldsymbol{d} = \boldsymbol{0}$$

$$\Updownarrow$$

$$\boxed{\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{w} = \boldsymbol{A}^T\boldsymbol{d}} \qquad \text{(Normal Equations)} \tag{1.43}$$

$$\Updownarrow$$

$$\boldsymbol{w} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{d} \qquad \text{(If } \boldsymbol{A} \text{ has full rank)} \tag{1.44}$$

4. Find the Hessian

$$\boldsymbol{H}(\boldsymbol{w}) = 2\boldsymbol{A}^T\boldsymbol{A} \tag{1.45}$$

   which is p.d. for all $\boldsymbol{w}$ if $\boldsymbol{A}$ has full rank.

5. This implies that

   - $J_2(\boldsymbol{w})$ is a convex function,
   - $\boldsymbol{w}_o = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{d}$ is the global minimiser, and
   - $J_2(\boldsymbol{w}_o) = \boldsymbol{d}^T(\boldsymbol{I} - \boldsymbol{P}_u)\boldsymbol{d} = \boldsymbol{d}^T\boldsymbol{P}_u^{\perp}\boldsymbol{d}$ where $\boldsymbol{P}_u = \boldsymbol{A}(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T$ is the orthogonal projection matrix and $\boldsymbol{P}_u^{\perp}$ is the complementary projection matrix to $\boldsymbol{P}_u$.

The solution $\boldsymbol{w}_o$ is often referred to as the least-squares solution.

Principle of Orthogonality

The normal equations can also be derived from the principle of orthogonality

$$\boldsymbol{0} = \boldsymbol{A}^T\boldsymbol{e}_o \iff \boldsymbol{0} = \boldsymbol{y}_o^T\boldsymbol{e}_o \iff \boldsymbol{w}_o = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{d}\ . \tag{1.46}$$

where $\boldsymbol{e}_o$ and $\boldsymbol{y}_o$ are the error vector and the filter output vector, respectively, when the filter is optimised in the squared sense. Fig. 1.3 illustrates the principle of orthogonality.

Estimation of the (Cross-)Correlation

Comparing the normal equations with the Wiener-Hopf equations, we see that

$$\hat{\boldsymbol{R}}_u = \frac{1}{K}\boldsymbol{A}^T\boldsymbol{A} \tag{1.47}$$

$$\hat{\boldsymbol{r}}_{ud} = \frac{1}{K}\boldsymbol{A}^T\boldsymbol{d} \tag{1.48}$$

are the build in estimates of the correlation matrix and the cross-correlation vector, respectively, in the method of least-squares.

Data Windowing

Recall the definition of $\boldsymbol{A}^T$ in Eq. (1.23) with $n_0 = 1$ and $K = N + M - 1$.

$$\left[\begin{array}{ccc|ccc|ccc} u(1) & \cdots & u(M-1) & u(M) & \cdots & u(N) & u(N+1) & \cdots & u(N+M-1) \\ u(0) & \cdots & u(M-2) & u(M-1) & \cdots & u(N-1) & u(N) & \cdots & u(N+M-2) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ u(2-M) & \cdots & u(0) & u(1) & \cdots & u(N-M+1) & u(N-M+2) & \cdots & u(N) \end{array}\right]$$

In real-world applications, we only observe a finite amount of data. Assume that we observe $u(n)$ for $n = 1, \ldots, N$ while the remaining data for $n \leq 0$ and $n > N$ are unobserved. If we set the unobserved data equal to zero, we obtain the structure for $\boldsymbol{A}$ shown in Fig 1.4. The various

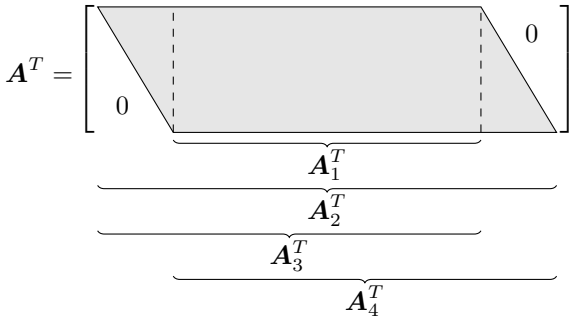

Figure 1.4: Four different data windowing methods.

choices for $\boldsymbol{A}$ are

$\boldsymbol{A}_1$ - covariance method:	$n_0 = M$	$K = N - M + 1$
$\boldsymbol{A}_2$ - autocorrelation method:	$n_0 = 1$	$K = N + M - 1$
$\boldsymbol{A}_3$ - prewindowing method:	$n_0 = 1$	$K = N$
$\boldsymbol{A}_4$ - postwindowing method:	$n_0 = M$	$K = N$

The cost function and the length of $\boldsymbol{e}$ in Eq. (1.25) and $\boldsymbol{d}$ in Eq. (1.24) must be adjusted according to the windowing method. The covariance method leads to unbiased estimates of $\boldsymbol{R}_u$ and $\boldsymbol{r}_{ud}$ in Eq. (1.47) and Eq. (1.48), respectively.

Lecture 2

Steepest Descent and Least-Mean-Square Adaptive Filters

Review of the Basics

The Eigenvalue Decomposition

If a square $M \times M$ matrix $\boldsymbol{A}$ has an eigenvalue decomposition (EVD), it can be written as

$$\boldsymbol{A} = \boldsymbol{X}\boldsymbol{\Lambda}\boldsymbol{X}^{-1} \qquad \Longleftrightarrow \qquad \boldsymbol{A}\boldsymbol{X} = \boldsymbol{X}\boldsymbol{\Lambda} \tag{2.1}$$

where $\boldsymbol{X}$ contains the M linearly independent eigenvectors of $\boldsymbol{A}$, and the diagonal matrix $\boldsymbol{\Lambda}$ contains the M eigenvalues $\{\lambda_m\}_{m=1}^{M}$ of $\boldsymbol{A}$. If $\boldsymbol{A}$ does not have an eigenvalue decomposition, it is said to be *defective*.

Matrix Powers

Let $\boldsymbol{A}$ have an EVD. Then

$$\boldsymbol{A} = \boldsymbol{X}\boldsymbol{\Lambda}\boldsymbol{X}^{-1} \tag{2.2}$$

$$\boldsymbol{A}^2 = \boldsymbol{X}\boldsymbol{\Lambda}\boldsymbol{X}^{-1}\boldsymbol{X}\boldsymbol{\Lambda}\boldsymbol{X}^{-1} = \boldsymbol{X}\boldsymbol{\Lambda}^2\boldsymbol{X}^{-1} \tag{2.3}$$

$$\vdots$$

$$\boldsymbol{A}^n = \boldsymbol{X}\boldsymbol{\Lambda}^n\boldsymbol{X}^{-1}\,. \tag{2.4}$$

EVD of Special Matrices

Let $\boldsymbol{A}$ be a symmetric matrix. Then

$\boldsymbol{A}$ has an EVD,

$\boldsymbol{A} = \boldsymbol{X}\boldsymbol{\Lambda}\boldsymbol{X}^T$ ($\boldsymbol{X}$ is an orthogonal matrix), and

$\lambda_m \in \mathbb{R}$ for $m = 1, \ldots, M$

Let $\boldsymbol{A}$ be a p.d. matrix. Then

$$\lambda_m \in \mathbb{R}^+ \text{ for } m = 1, \ldots, M.$$

where $\mathbb{R}^+$ is the set of all positive real numbers.

Matrix Trace

The trace of $\boldsymbol{A}$ is

$$\text{tr}(\boldsymbol{A}) = \sum_{m=1}^{M} a_{mm} = \sum_{m=1}^{M} \lambda_m = \text{tr}(\boldsymbol{\Lambda}) \ . \tag{2.5}$$

Condition Number of a Normal Matrix

If $\boldsymbol{A}$ is a normal matrix (i.e., $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{A}\boldsymbol{A}^T$), then the condition number of $\boldsymbol{A}$ is

$$\kappa(\boldsymbol{A}) = \left| \frac{\lambda_{\max}}{\lambda_{\min}} \right| = \chi(\boldsymbol{A}) \tag{2.6}$$

where $\chi(\boldsymbol{A})$ is the eigenvalue spread of $\boldsymbol{A}$.

The Wiener-Hopf Equations

The adaptive filtering problem with time-varying filter coefficients is shown in Fig. 2.1. From

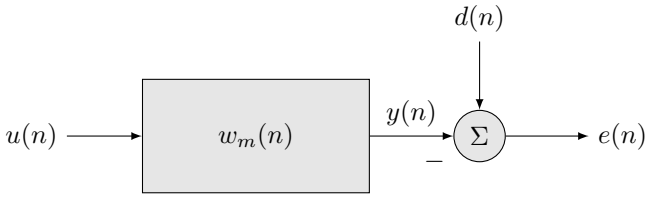

Figure 2.1: Block diagram of adaptive filtering in a non-WSS environment.

the figure, we have that

$u(n)$: zero-mean input signal

$w_m(n)$: M-tap FIR-filter with impulse response $w_0(n), w_1(n), \ldots, w_{M-1}(n)$

$y(n)$: output signal given by $y_n = \sum_{m=0}^{M-1} w_m(n)u(n-m)$

$d(n)$: zero-mean desired signal

$e(n)$: error signal

WSS Signals

When $u(n)$ and $d(n)$ are jointly WSS, the filter coefficients are not time-varying. Define

$$\boldsymbol{w} = \begin{bmatrix} w_0 & w_1 & \cdots & w_{M-1} \end{bmatrix}^T \tag{2.7}$$

$$\boldsymbol{u}(n) = \begin{bmatrix} u(n) & u(n-1) & \cdots & u(n-M+1) \end{bmatrix}^T \ . \tag{2.8}$$

Then

$$e(n) = d(n) - y(n) = d(n) - \boldsymbol{u}^T(n)\boldsymbol{w} \ . \tag{2.9}$$

We wish to minimise

$$\begin{aligned} J_1(\boldsymbol{w}) &= E[e(n)^2] = E[d(n)^2] + \boldsymbol{w}^T E[\boldsymbol{u}(n)\boldsymbol{u}^T(n)]\boldsymbol{w} - 2\boldsymbol{w}^T E[\boldsymbol{u}(n)d(n)] \\ &= \sigma_d^2 + \boldsymbol{w}^T \boldsymbol{R}_u \boldsymbol{w} - 2\boldsymbol{w}^T \boldsymbol{r}_{ud} \end{aligned} \tag{2.10}$$

w.r.t. $\boldsymbol{w}$. The minimiser

$$\boldsymbol{w}_o = \boldsymbol{R}_u^{-1} \boldsymbol{r}_{ud} \tag{2.11}$$

is the unique solution to the Wiener-Hopf equations

$$\boldsymbol{R}_u \boldsymbol{w} = \boldsymbol{r}_{ud} \ , \tag{2.12}$$

provided that $\boldsymbol{R}_u$ is p.d.

Non-WSS Signals

If $u(n)$ and $d(n)$ are *not* jointly WSS, the optimal filter coefficients are time-varying, and we have that

$$\boldsymbol{R}_u(n)\boldsymbol{w}(n) = \boldsymbol{r}_{ud}(n) \tag{2.13}$$

where

$$\boldsymbol{R}_u(n) = \begin{bmatrix} r_u(n,n) & \cdots & r_u(n,n-M+1) \\ \vdots & \ddots & \vdots \\ r_u(n-M+1,n) & \cdots & r_u(n-M+1,n-M+1) \end{bmatrix} \tag{2.14}$$

$$\boldsymbol{r}_{ud}(n) = \begin{bmatrix} r_{ud}(n,n) & \cdots & r_{ud}(n-M+1,n) \end{bmatrix}^T \ . \tag{2.15}$$

We could find the optimal solution by calculating

$$\boldsymbol{w}_o(n) = \boldsymbol{R}_u^{-1}(n) \boldsymbol{r}_{ud}(n) \tag{2.16}$$

for every time index n. However, this approach may suffer from problems such as that

1. the computational complexity is high, and
2. the statistics is unknown.

The steepest descent algorithm solves the first problem, and the least-mean-square (LMS) adaptive filter solves both problems.

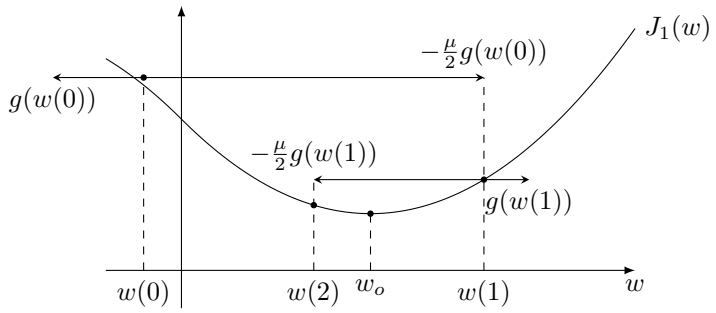

Figure 2.2: The first three iterations of the steepest descent algorithm for a one-dimensional filter vector.

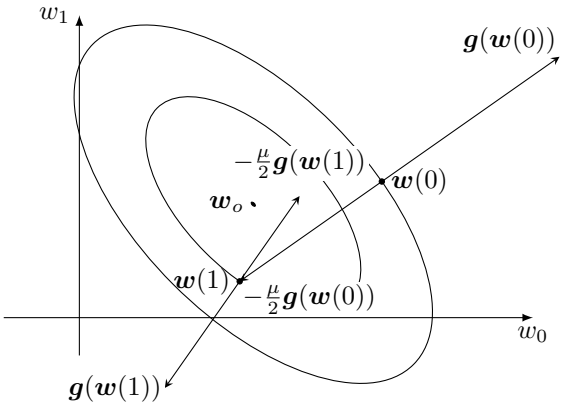

Figure 2.3: The first three iterations of the steepest descent algorithm for a two-dimensional filter vector.

The Method of Steepest Descent

Basic Idea

The basic idea of the steepest descent (SD) algorithm is to find the unique solution $\boldsymbol{w}_o$ of the Wiener-Hopf equations through a series of steps, starting from some point $\boldsymbol{w}(0)$. The steps are taken in the opposite direction of the gradient $\boldsymbol{g}(\boldsymbol{w}(n))$. This idea is illustrated in Fig. 2.2 and Fig. 2.3. We update the filter coefficients by a recursive update equation given by

$$\boxed{\boldsymbol{w}(n+1) = \boldsymbol{w}(n) - \frac{\mu}{2}\boldsymbol{g}(\boldsymbol{w}(n))} \tag{2.17}$$

where μ is the step-size and $\boldsymbol{g}(\boldsymbol{w}(n))$ is the gradient of the cost function $J_1(\boldsymbol{w}(n))$. The gradient is

$$\boldsymbol{g}(\boldsymbol{w}(n)) = 2\boldsymbol{R}_u(n)\boldsymbol{w}(n) - 2\boldsymbol{r}_{ud}(n) \ . \tag{2.18}$$

Since the gradient is a deterministic function, the evolution of the filter coefficient vector is also a deterministic function.

Define the weight error

$$\Delta\boldsymbol{w}(n) = \boldsymbol{w}_o - \boldsymbol{w}(n) \ . \tag{2.19}$$

In order for the SD algorithm to converge to the solution $\boldsymbol{w}_o$, we must require that the step-size μ is selected such that

$$\lim_{n\to\infty} \Delta\boldsymbol{w}(n) = 0 \tag{2.20}$$

when $u(n)$ and $d(n)$ are jointly WSS. If this is true, the SD algorithm is said to be *stable*. Moreover, we would like to select μ such that the $\Delta\boldsymbol{w}(n)$ becomes small as fast as possible.

Transient Analysis

Assume that $u(n)$ and $d(n)$ are jointly WSS. We then have that

$$\begin{aligned}
\Delta\boldsymbol{w}(n) &= \boldsymbol{w}_o - \Big(\boldsymbol{w}(n-1) - \frac{\mu}{2}\boldsymbol{g}(\boldsymbol{w}(n-1))\Big) && (2.21)\\
&= \Delta\boldsymbol{w}(n-1) + \mu(\boldsymbol{R}_u\boldsymbol{w}(n-1) - \boldsymbol{R}_u\boldsymbol{w}_o) && (2.22)\\
&= (\boldsymbol{I} - \mu\boldsymbol{R}_u)\Delta\boldsymbol{w}(n-1) && (2.23)\\
&= (\boldsymbol{I} - \mu\boldsymbol{R}_u)(\boldsymbol{I} - \mu\boldsymbol{R}_u)\Delta\boldsymbol{w}(n-2) && (2.24)\\
&= (\boldsymbol{I} - \mu\boldsymbol{R}_u)^n\Delta\boldsymbol{w}(0) \ . && (2.25)
\end{aligned}$$

Since the correlation matrix $\boldsymbol{R}_u$ is symmetric, it has an eigenvalue decomposition $\boldsymbol{R}_u = \boldsymbol{X}\boldsymbol{\Lambda}\boldsymbol{X}^T$ where $\boldsymbol{X}$ is an orthogonal matrix. That is, we may write

$$\Delta\boldsymbol{w}(n) = \boldsymbol{X}(\boldsymbol{I} - \mu\boldsymbol{\Lambda})^n\boldsymbol{X}^T\Delta\boldsymbol{w}(0) \tag{2.26}$$

where $\boldsymbol{I} - \mu\boldsymbol{\Lambda}$ is a diagonal matrix with the m'th diagonal given by the mode $1 - \mu\lambda_{\mathrm{m}}$. For Eq. (2.20) to be fulfilled from any starting point $\boldsymbol{w}(0)$, we must therefore require that

$$\lim_{n\to\infty}(1 - \mu\lambda_{\mathrm{m}})^n = 0 \ , \quad \text{for } m = 1, 2, \ldots, M \ . \tag{2.27}$$

Thus

$$\begin{aligned}
&-1 < 1-\mu\lambda_{\mathrm{m}} < 1 \ , \quad \text{for } m = 1, 2, \ldots, M && (2.28)\\
\Updownarrow\\
&-1 < 1-\mu\lambda_{\max} < 1 && (2.29)
\end{aligned}$$

Solving for μ leads to that the SD algorithm is stable if

$$\boxed{0 < \mu < \frac{2}{\lambda_{\max}}} \,. \tag{2.30}$$

From Eq. (2.20), we see that

$$\begin{aligned} |1-\mu\lambda_m| \text{ is close to } 0 \quad &\Longleftrightarrow \quad \text{mode with fast convergence} \\ |1-\mu\lambda_m| \text{ is close to } 1 \quad &\Longleftrightarrow \quad \text{mode with slow convergence} \end{aligned}$$

The optimal step-size μ_o is therefore given by

$$\mu_o = \arg\min_{\mu} \max_{\lambda_m} |1-\mu\lambda_m| \tag{2.31}$$

$$\text{subject to} \qquad |1-\mu\lambda_m| < 1 \;, \text{ for } m = 1, \ldots, M \;. \tag{2.32}$$

That is, μ_o minimises the value of the largest (slowest) mode. We solve this optimisation problem by the use of Fig 2.4. From Fig 2.4, we see that the optimal step-size μ_o satisfies

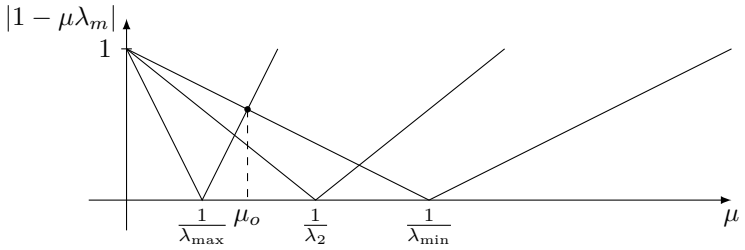

Figure 2.4: Finding the optimal step-size.

$$1-\mu_o\lambda_{\min} = -(1-\mu_o\lambda_{\max}) \tag{2.33}$$

$$\Updownarrow$$

$$\boxed{\mu_o = \frac{2}{\lambda_{\max}+\lambda_{\min}}} \,. \tag{2.34}$$

For the optimal step-size, the slowest modes $\pm(1-\mu_o\lambda_{\max})$ and $\pm(1-\mu_o\lambda_{\min})$ are therefore given by

$$\pm\frac{\lambda_{\max}-\lambda_{\min}}{\lambda_{\max}+\lambda_{\min}} = \pm\frac{\kappa(\boldsymbol{R}_u)-1}{\kappa(\boldsymbol{R}_u)+1} \tag{2.35}$$

where $\kappa(\boldsymbol{R}_u)$ is the condition number of the correlation matrix $\boldsymbol{R}_u$. Thus, if the condition number is large, the slowest modes are close to one, and the convergence of the SD algorithm is slow. Conversely, if the condition number is close to one, the slowest modes are close to zero, and the convergence of the SD algorithm is fast.

Learning Curve

The cost function can be written as

$$J_1(\boldsymbol{w}(n)) = J_1(\boldsymbol{w}_o) + \Delta\boldsymbol{w}^T(n)\boldsymbol{R}_u\Delta\boldsymbol{w}(n) \tag{2.36}$$

$$= J_1(\boldsymbol{w}_o) + \Delta\boldsymbol{w}^T(0)\boldsymbol{X}(\boldsymbol{I}-\mu\boldsymbol{\Lambda})^n\boldsymbol{\Lambda}(\boldsymbol{I}-\mu\boldsymbol{\Lambda})^n\boldsymbol{X}^T\Delta\boldsymbol{w}(0) \tag{2.37}$$

$$= J_1(\boldsymbol{w}_o) + \sum_{m=1}^{M} \lambda_m(1-\mu\lambda_m)^{2n}\left(\boldsymbol{x}_m^T\Delta\boldsymbol{w}(0)\right)^2 \;. \tag{2.38}$$

The plot of $J_1(\boldsymbol{w}(n))$ as a function of n is called the learning curve, and a sketch of it is shown in Fig. 2.5.

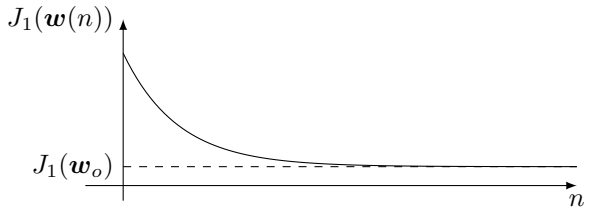

Figure 2.5: The learning curve of the steepest descent algorithm.

Least-Mean-Square Adaptive Filters

Basic Idea

Recall, that the gradient of the mean-squared error cost function $J_1(\boldsymbol{w}(n))$ is

$$\boldsymbol{g}(\boldsymbol{w}(n)) = \frac{\partial E[e^2(n)]}{\partial \boldsymbol{w}(n)} = 2E\left[\frac{\partial e(n)}{\partial \boldsymbol{w}(n)} e(n)\right] = -2E[\boldsymbol{u}(n)e(n)] \ . \tag{2.39}$$

Often, we do not know the gradient, and we therefore have to estimate it. A simple estimate is

$$\hat{\boldsymbol{g}}(\boldsymbol{w}(n)) = -2\boldsymbol{u}(n)e(n) \ . \tag{2.40}$$

If we replace the gradient in the steepest descent algorithm with this estimate, we obtain

$$\boxed{\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu \boldsymbol{u}(n)e(n)} \tag{2.41}$$

which is called the least-mean-square (LMS) algorithm. A block diagram of the LMS filter is depicted in Fig. 2.6. The LMS algorithm is the simplest *stochastic gradient method* (SGM). The

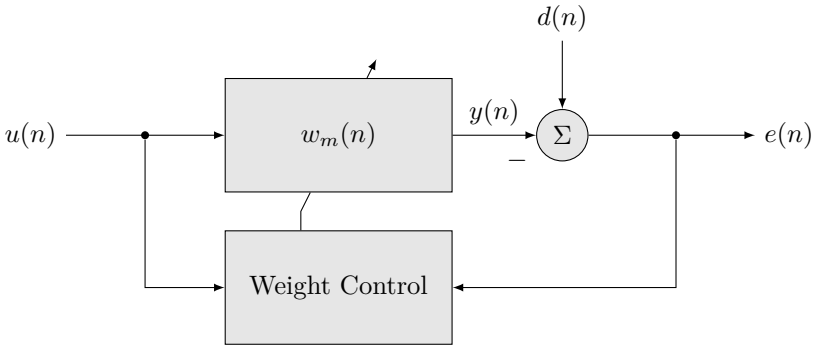

Figure 2.6: Typical block diagram of an adaptive filter in the case where the statistics is unknown.

naming of the SGM refers to that the gradient is a stochastic process. This has the following consequences.

1. The filter coefficient vector $\boldsymbol{w}(n)$ is also a stochastic process. This makes the analysis of the SGMs difficult. To see this, consider the cost function

$$\begin{aligned} J_1(\boldsymbol{w}(n)) &= E[e^2(n)] \\ &= \sigma_d^2(n) + E[\boldsymbol{w}^T(n)\boldsymbol{u}(n)\boldsymbol{u}^T(n)\boldsymbol{w}(n)] - 2E[\boldsymbol{w}^T(n)\boldsymbol{u}(n)d(n)] \ . \end{aligned} \tag{2.42}$$

   For the stochastic gradient methods, it is not easy to evaluate the expectations in the cost function since $\boldsymbol{w}(n)$ and $\boldsymbol{u}(n)$ are not independent (unless $M = 1$ and $u(n)$ is a white process).

2. The weight error $\Delta\boldsymbol{w}(n) = \boldsymbol{w}_o - \boldsymbol{w}(n)$ never goes permanently to zero. That is, in steady-state ($n \to \infty$), the filter coefficient vector $\boldsymbol{w}(\infty)$ fluctuates randomly around the optimum $\boldsymbol{w}_o$. Consequently, the mean-square error (MSE) $J_1(\boldsymbol{w}(\infty))$ in steady-state is larger than $J_{\min} = J_1(\boldsymbol{w}_o)$ by an amount referred to as the *excess mean-square* error (EMSE)

$$J_{\text{ex}} = J_1(\boldsymbol{w}(\infty)) - J_{\min} \ . \tag{2.43}$$

   The ratio of the EMSE to the MSE is called the *misadjustment*

$$\mathcal{M} = \frac{J_{\text{ex}}}{J_{\min}} = \frac{J_1(\boldsymbol{w}(\infty))}{J_{\min}} - 1 \ . \tag{2.44}$$

In Lecture 3, we say more about these parameters and the analysis of adaptive filters.

Transient Analysis

Assume that $d(n)$ is given by[1]

$$d(n) = \boldsymbol{u}^T(n)\boldsymbol{w}_o + v(n) \tag{2.45}$$

where $v(n)$ is white Gaussian noise with variance σ_v^2 and uncorrelated with $u(n)$. Moreover, assume that the random variables of the stochastic process $\boldsymbol{u}(n)$ are independent and identically distributed (IID) with a Gaussian distribution and correlation matrix $\boldsymbol{R}_u$. This assumption is infeasible unless $M = 1$. Nevertheless, we make this assumption anyway in order to make the transient analysis as simple as possible. We are concerned with

$$\begin{aligned} \lim_{n\to\infty} E[\Delta\boldsymbol{w}(n)] &= 0 && \text{(Convergence in mean)} \\ \lim_{n\to\infty} E[\|\Delta\boldsymbol{w}(n)\|^2] &= c < \infty && \text{(Convergence in mean-square)} \end{aligned}$$

where $\|\Delta\boldsymbol{w}(n)\|^2 = \Delta\boldsymbol{w}^T(n)\Delta\boldsymbol{w}(n)$ is the vector 2-norm, and c in some positive constant[2].

Convergence in the Mean

By subtracting the LMS recursion from the optimal filter coefficients, we obtain

$$\Delta\boldsymbol{w}(n) = \boldsymbol{w}_o - [\boldsymbol{w}(n-1) + \mu\boldsymbol{u}(n-1)e(n-1)] \tag{2.46}$$

$$= \Delta\boldsymbol{w}(n-1) - \mu\boldsymbol{u}(n-1)\left[d(n-1) - \boldsymbol{u}^T(n-1)\boldsymbol{w}(n-1)\right] \tag{2.47}$$

$$= \left[\boldsymbol{I} - \mu\boldsymbol{u}(n-1)\boldsymbol{u}^T(n-1)\right]\Delta\boldsymbol{w}(n-1) - \mu\boldsymbol{u}(n-1)v(n-1) \tag{2.48}$$

[1] This model is quite popular for the analysis of adaptive filters. We say more about this model in Sec. 3.2.1.

[2] Apparently, the definition of convergence in mean-square used for adaptive filtering analysis is different from the usual definition where $c = 0$.

Taking the expectation of both sides and using the above assumptions, we obtain

$$E[\Delta \boldsymbol{w}(n)] = (\boldsymbol{I} - \mu \boldsymbol{R}_u) E[\Delta \boldsymbol{w}(n-1)] \tag{2.49}$$

$$= \boldsymbol{X}(\boldsymbol{I} - \mu \boldsymbol{\Lambda})^n \boldsymbol{X}^T E[\Delta \boldsymbol{w}(0)] \ . \tag{2.50}$$

This recursion is the same as in the transient analysis of the steepest descent algorithm. We therefore have that the LMS algorithm is stable in the mean if

$$\boxed{0 < \mu < \frac{2}{\lambda_{\max}}} \ . \tag{2.51}$$

The bound given above ensures convergence in the mean, but places no constraint on the variance of $\Delta \boldsymbol{w}(n)$. Furthermore, since $\boldsymbol{R}_u$ is unknown, $\lambda_{\max}$ is unknown, and we have to estimate or upper bound it. We have that

$$\lambda_{\max} \leq \mathrm{tr}(\boldsymbol{R}_u) = \sum_{m=1}^{M} r_u(0) = M r_u(0) = M E[u^2(n)] \ . \tag{2.52}$$

The expected power $E[u^2(n)]$ can be estimated as

$$\hat{E}[u^2(n)] = \frac{1}{M} \boldsymbol{u}^T(n) \boldsymbol{u}(n) \ . \tag{2.53}$$

Note that the estimation accuracy increases with the filter length M. From Parseval's Theorem, we also have that

$$E[u^2(n)] = \frac{1}{2\pi} \int_{-\pi}^{\pi} S_u(\omega) d\omega \leq \frac{1}{2\pi} \int_{-\pi}^{\pi} S_{\max} d\omega = S_{\max} \tag{2.54}$$

where $S_u(\omega)$ is the power spectral density of $u(n)$. Thus, we have that

$$\boxed{\frac{2}{M S_{\max}} \leq \frac{2}{\mathrm{tr}(\boldsymbol{R}_u)} = \frac{2}{M E[u^2(n)]} \leq \frac{2}{\lambda_{\max}}} \ , \tag{2.55}$$

provided that the desired signal model is given by Eq. (2.45) and that $\boldsymbol{u}(n)$ is an IID random process.

Convergence in the Mean-Square

From Eq. (2.48), we have that

$$\begin{aligned} \|\Delta \boldsymbol{w}(n)\|^2 &= \| \left[\boldsymbol{I} - \mu \boldsymbol{u}(n-1) \boldsymbol{u}^T(n-1)\right] \Delta \boldsymbol{w}(n-1) - \mu \boldsymbol{u}(n-1) v(n-1) \|^2 \\ &= \| \left[\boldsymbol{I} - \mu \boldsymbol{u}(n-1) \boldsymbol{u}^T(n-1)\right] \Delta \boldsymbol{w}(n-1) \|^2 \\ &\quad + \mu^2 v^2(n-1) \|\boldsymbol{u}(n-1)\|^2 \\ &\quad - 2\mu v(n) \boldsymbol{u}^T(n-1) \left[\boldsymbol{I} - \mu \boldsymbol{u}(n-1) \boldsymbol{u}^T(n-1)\right] \Delta \boldsymbol{w}(n-1) \end{aligned} \tag{2.56}$$

Taking the expectation on both sides and using the above assumptions, we obtain

$$E[\|\Delta \boldsymbol{w}(n)\|^2] = E\left[\| \left[\boldsymbol{I} - \mu \boldsymbol{u}(n-1) \boldsymbol{u}^T(n-1)\right] \Delta \boldsymbol{w}(n-1) \|^2\right] + \mu^2 J_{\min} \mathrm{tr}(\boldsymbol{\Lambda}) \tag{2.57}$$

where $J_{\min} = \sigma_v^2$. Evaluating the expected value of the first term and finding the values of the step-size μ for which $E[\|\Delta \boldsymbol{w}(n)\|^2]$ converges in the mean-square require a lot of work. In

Appendix B, we show how this can be done. Alternatively, a derivation can also be found in [7, pp. 452–465]. The final result is that the LMS filter is mean-square stable (and thus stable in the mean) if and only if the step-size satisfies [7, pp. 462]

$$\boxed{f(\mu) = \frac{\mu}{2}\sum_{m=1}^{M}\frac{\lambda_m}{1-\mu\lambda_m} = \frac{1}{2}\text{tr}(\boldsymbol{\Lambda}(\mu^{-1}\boldsymbol{I}-\boldsymbol{\Lambda})^{-1}) < 1} \ . \tag{2.58}$$

For small step-sizes $\mu \ll 1/\lambda_{\max}$, we have that

$$f(\mu) \approx \frac{\mu}{2}\sum_{m=1}^{M}\lambda_m = \frac{\mu}{2}\text{tr}(\boldsymbol{\Lambda}) < 1 \ . \tag{2.59}$$

which leads to that the LMS algorithm is mean-square stable if

$$\boxed{0 < \mu < \frac{2}{\text{tr}(\boldsymbol{R}_u)}} \ . \tag{2.60}$$

Learning Curve

The learning curve is also difficult to find. In Appendix B, we show that the learning curve is given by

$$\boxed{J_1(\boldsymbol{w}(n)) = \boldsymbol{\Delta w}^T(0)\boldsymbol{X}\boldsymbol{S}(n)\boldsymbol{X}^T\boldsymbol{\Delta w}(0) + \mu^2 J_{\min}\sum_{i=0}^{n-1}\text{tr}(\boldsymbol{S}(i)\boldsymbol{\Lambda}) + J_{\min}} \tag{2.61}$$

where

$$\boldsymbol{S}(n) = \boldsymbol{S}(n-1) - 2\mu\boldsymbol{\Lambda}\boldsymbol{S}(n-1) + \mu^2[\boldsymbol{\Lambda}\text{tr}(\boldsymbol{S}(n-1)\boldsymbol{\Lambda}) + 2\boldsymbol{\Lambda}\boldsymbol{S}(n-1)\boldsymbol{\Lambda}] \tag{2.62}$$

with $\boldsymbol{S}(0) = \boldsymbol{\Lambda}$.

Steady-State Analysis

When the LMS algorithm is operating in steady-state and is mean-square stable, the EMSE, the mean-square deviation (MSD), and the misadjustment are [7, pp. 465]

$$J_{\text{ex}} = J_{\min}\frac{f(\mu)}{1-f(\mu)} \tag{2.63}$$

$$\mathcal{M} = \frac{f(\mu)}{1-f(\mu)} \tag{2.64}$$

$$E[\|\boldsymbol{\Delta w}(\infty)\|^2] = \frac{J_{\min}}{1-f(\mu)}\frac{\mu}{2}\sum_{m=1}^{M}\frac{1}{1-\mu\lambda_m} \ . \tag{2.65}$$

These results are also derived in Appendix B. For small step-sizes $\mu \ll 1/\lambda_{\max}$, the EMSE, MSD, and misadjustment simplify to [7, pp. 465]

$$J_{\text{ex}} \approx \mu J_{\min}\frac{\text{tr}(\boldsymbol{R}_u)}{2-\mu\text{tr}(\boldsymbol{R}_u)} \approx \frac{\mu}{2}J_{\min}\text{tr}(\boldsymbol{R}_u) \tag{2.66}$$

$$\mathcal{M} \approx \mu\frac{\text{tr}(\boldsymbol{R}_u)}{2-\mu\text{tr}(\boldsymbol{R}_u)} \approx \frac{\mu}{2}\text{tr}(\boldsymbol{R}_u) = \frac{\mu}{2}ME[u^2(n)] \tag{2.67}$$

$$E[\|\Delta\boldsymbol{w}(\infty)\|^2] \approx \mu J_{\min}\frac{M}{2-\mu\text{tr}(\boldsymbol{R}_u)} \approx \frac{\mu}{2}J_{\min}M \ . \tag{2.68}$$

The approximated expression for the misadjustment shows that misadjustment is approximately proportional to the input power. This is undesirable and referred to as *gradient noise amplification*. In Lecture 3, we consider the normalised LMS algorithm which solves this problem.

Computational Cost

Table 2.1 shows the computational cost of the LMS algorithm in terms of the number of multiplications and additions or subtractions. From the table, we see that the total number of flops is $4M + 1$. Thus, the LMS algorithm has a linear complexity $\mathcal{O}(M)$ in the filter length M.

Term	$\times$	$+$ or $-$
$\boldsymbol{u}^T(n)\boldsymbol{w}(n)$	M	$M-1$
$e(n) = d(n) - \boldsymbol{u}^T(n)\boldsymbol{w}(n)$		1
$\mu e(n)$	1	
$\mu \boldsymbol{u}(n) e(n)$	M	
$\boldsymbol{w}(n) + \mu \boldsymbol{u}(n) e(n)$		M
Total	$2M+1$	$2M$

Table 2.1: Computational cost of the LMS algorithm.

Lecture 3

Normalised LMS and Affine Projection Algorithm

Review of the Basics

Inversion of a 2 by 2 Block Matrix

Let $\boldsymbol{A}$ and $\boldsymbol{M}_1 = \boldsymbol{D} - \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B}$ be non-singular matrices. Then

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{A}^{-1} + \boldsymbol{A}^{-1}\boldsymbol{B}\boldsymbol{M}_1^{-1}\boldsymbol{C}\boldsymbol{A}^{-1} & -\boldsymbol{A}^{-1}\boldsymbol{B}\boldsymbol{M}_1^{-1} \\ -\boldsymbol{M}_1^{-1}\boldsymbol{C}\boldsymbol{A}^{-1} & \boldsymbol{M}_1^{-1} \end{bmatrix} . \tag{3.1}$$

Let instead $\boldsymbol{D}$ and $\boldsymbol{M}_2 = \boldsymbol{A} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C}$ be non-singular matrices. Then

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{M}_2^{-1} & -\boldsymbol{M}_2^{-1}\boldsymbol{B}\boldsymbol{D}^{-1} \\ -\boldsymbol{D}^{-1}\boldsymbol{C}\boldsymbol{M}_2^{-1} & \boldsymbol{D}^{-1} + \boldsymbol{D}^{-1}\boldsymbol{C}\boldsymbol{M}_2^{-1}\boldsymbol{B}\boldsymbol{D}^{-1} \end{bmatrix} . \tag{3.2}$$

The matrices $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ are said to be the *Schur complement* of $\boldsymbol{A}$ and $\boldsymbol{D}$, respectively.

The Method of Lagrange Multipliers

The method of Lagrange multipliers can be used to solve optimisation problems with equality constraints[1]

$$\begin{aligned} &\underset{\boldsymbol{x}}{\text{optimise}} && f(\boldsymbol{x}) \\ &\text{subject to} && \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0} \end{aligned} \tag{3.3}$$

Consider the optimisation problem in Fig. 3.1 with the cost function $f(\boldsymbol{x})$ and the constraints $\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0}$. At the constrained optimum $\boldsymbol{x}_c$, the gradients of $f(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$ are parallel. Thus,

$$\nabla f(\boldsymbol{x}) = \lambda \nabla h(\boldsymbol{x}) \qquad \text{(single constraint)}$$

$$\nabla f(\boldsymbol{x}) = \sum_{m=1}^{M} \lambda_m \nabla h_m(\boldsymbol{x}) = \nabla \boldsymbol{h}(\boldsymbol{x})\boldsymbol{\lambda} \qquad \text{(multiple constraints)}$$

[1] The method of Lagrange multipliers can also be used to solve the more general problem with inequality constraints, but we do not consider this here.

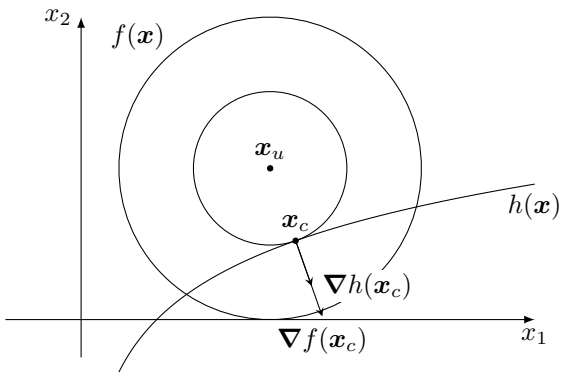

Figure 3.1: Optimisation problem with a single equality constraint. The points $\boldsymbol{x}_u$ and $\boldsymbol{x}_c$ are the unconstrained and the constrained minimum, respectively.

The vector $\boldsymbol{\lambda}$ contains the Lagrange multipliers, and they ensure that the gradients have the same direction and length.

If we define the Lagrangian function

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \boldsymbol{\lambda}^T \boldsymbol{h}(\boldsymbol{x}) \ , \tag{3.4}$$

then its critical points satisfy

$$\nabla_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda}) = \boldsymbol{0} \quad \Longleftrightarrow \quad \nabla f(\boldsymbol{x}) = \nabla \boldsymbol{h}(\boldsymbol{x}) \boldsymbol{\lambda} \tag{3.5}$$

$$\nabla_{\boldsymbol{\lambda}} L(\boldsymbol{x}, \boldsymbol{\lambda}) = \boldsymbol{0} \quad \Longleftrightarrow \quad \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0} \ . \tag{3.6}$$

Eq. (3.5) is the same as what we previously derived from Fig. 3.1, and Eq. (3.6) is simply the constraints of our optimisation problem. Thus, the solutions to the system of equations given by Eq. (3.5) and Eq. (3.6) are indeed the critical points of our optimisation problems. In order to find the optimum, we need to classify the critical points as either minimums, maximums, or saddle points. The classification of the critical points for any pair of cost function and constraints is beyond the scope of these lecture notes. We refer the interested reader to [3, pp. 294–308]. Alternatively, we may formulate the constrained optimisation problem in Eq. (3.3) as the sequential unconstrained optimisation problem [4, ch. 5]

$$\max_{\boldsymbol{\lambda}} \left\{ \underset{\boldsymbol{x}}{\text{optimise}} \ L(\boldsymbol{x}, \boldsymbol{\lambda}) \right\} \ . \tag{3.7}$$

We could solve this optimisation problem by applying the five step recipe for unconstrained optimisation in Sec. 1.1.2 to first the inner optimisation problem and then the outer optimisation problem.

We are here not concerned with these general methods. Instead, we consider the simpler problem of minimising a quadratic cost function with linear equality constraints.

Quadratic Cost Function with Linear Equality Constraints

We consider the following equality constrained minimisation problem

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) = \frac{1}{2} \boldsymbol{x}^T \boldsymbol{P} \boldsymbol{x} + \boldsymbol{q}^T \boldsymbol{x} + r \\ \text{subject to} \quad & \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} = \boldsymbol{0} \end{aligned} \tag{3.8}$$

in which the cost function is quadratic and the equality constraints are linear. We assume that $\boldsymbol{P}$ is an $N \times N$ p.d. matrix so that $f(\boldsymbol{x})$ is a convex function. Moreover, we assume that $\boldsymbol{A}$ is a full rank $M \times N$ matrix with $M \leq N$. The Langrangian function is

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x} + \boldsymbol{q}^T\boldsymbol{x} + r - \boldsymbol{\lambda}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \tag{3.9}$$

and its gradient w.r.t. $\boldsymbol{x}$ is

$$\boldsymbol{\nabla}_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda}) = \boldsymbol{P}\boldsymbol{x} + \boldsymbol{q} - \boldsymbol{A}^T\boldsymbol{\lambda} \, . \tag{3.10}$$

The system of equations given by Eq. (3.5) and Eq. (3.6) are therefore

$$\begin{bmatrix} \boldsymbol{P} & -\boldsymbol{A}^T \\ -\boldsymbol{A} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{q} \\ -\boldsymbol{b} \end{bmatrix} \, . \tag{3.11}$$

Since the cost function $f(\boldsymbol{x})$ is assumed to be convex, there is only one point satisfying this system of equations. By using the 2×2 block matrix inversion rule in Eq. (3.1), we obtain that this point $(\boldsymbol{x}_c, \boldsymbol{\lambda}_c)$ is

$$\boldsymbol{\lambda}_c = (\boldsymbol{A}\boldsymbol{P}^{-1}\boldsymbol{A}^T)^{-1}(\boldsymbol{A}\boldsymbol{P}^{-1}\boldsymbol{q} + \boldsymbol{b}) \tag{3.12}$$

$$\boldsymbol{x}_c = \boldsymbol{P}^{-1}(\boldsymbol{A}^T\boldsymbol{\lambda}_c - \boldsymbol{q}) \, . \tag{3.13}$$

Thus, the solution to the constrained minimisation problem in Eq. (3.8) is $\boldsymbol{x}_c$ given by Eq. (3.13).

Overview over Adaptive Filters based on the Mean-Squared Error Cost Function

An overview over the adaptive filters is shown in Fig. A.1. Recall that we have the following important properties for the steepest descent algorithm and the stochastic gradient methods.

Steepest Descent (SD)

- Since the statistics is assumed known, the gradient and filter coefficients are deterministic functions.
- SD is of limited practical usage, but illustrates nicely how adaptive filters function an how they are analysed.
- SD is the "ideal" (first-order) adaptive filter.

Stochastic Gradient Methods (SGMs)

- Since the statistics is unknown, the gradient and filter coefficients are stochastic processes.
- They are generally easy to implement and hard to analyse.
- They are approximations to the steepest descent algorithm.

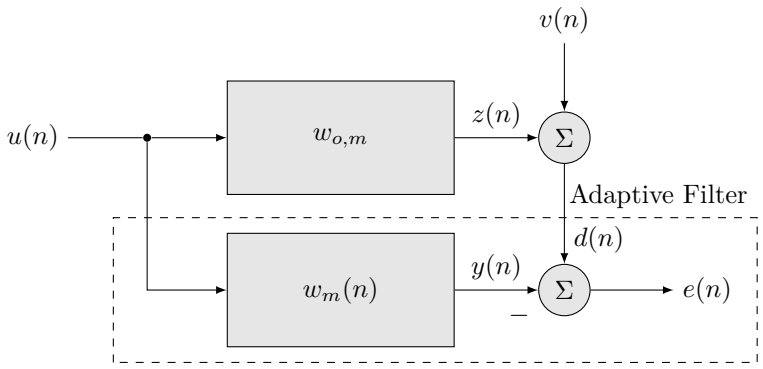

Figure 3.2: Block diagram showing the model used to analyse adaptive filters.

Model for the Analysis of SGMs

In order to make the SGMs easier to analyse, a model is typically assumed for the relationship between the input signal $u(n)$ and the desired signal $d(n)$. Fig. 3.2 shows one of the most popular models used for the analysis of adaptive filters. Define

$$\boldsymbol{w}_o = \begin{bmatrix} w_{o,0} & \cdots & w_{o,M-1} \end{bmatrix}^T \tag{3.14}$$

$$\boldsymbol{w}(n) = \begin{bmatrix} w_0(n) & \cdots & w_{M-1}(n) \end{bmatrix}^T \tag{3.15}$$

$$\boldsymbol{u}(n) = \begin{bmatrix} u(n) & \cdots & u(n-M+1) \end{bmatrix}^T \tag{3.16}$$

$$\boldsymbol{\Delta w}(n) = \boldsymbol{w}_o - \boldsymbol{w}(n) \tag{3.17}$$

where $\boldsymbol{\Delta w}(n)$ is referred to as the weight error.

Assumptions

From Fig. 3.2, we have that

$$d(n) = z(n) + v(n) = \boldsymbol{u}^T(n)\boldsymbol{w}_o + v(n) \ . \tag{3.18}$$

Moreover, we assume that

1. $v(n)$ is white noise with variance σ_v^2.
2. $v(n)$ and $u(n)$ are uncorrelated.
3. $\boldsymbol{u}(n)$ is a white process with correlation matrix $\boldsymbol{R}_u$. This assumption implies that $\boldsymbol{u}(n)$ and $\boldsymbol{w}(n)$ are uncorrelated.

Equations

We have that

$$e(n) = d(n) - y(n) = v(n) + \boldsymbol{u}^T(n)\boldsymbol{w}_o - \boldsymbol{u}^T(n)\boldsymbol{w}(n) \tag{3.19}$$

$$= \boldsymbol{u}^T(n)\boldsymbol{\Delta w}(n) + v(n) \tag{3.20}$$

Thus, at the optimum where $\boldsymbol{\Delta w}(n) = \mathbf{0}$, we have that

$$J_1(\boldsymbol{w}_o) = J_{\min} = J_1(v^2(n)) = \sigma_v^2 \ . \tag{3.21}$$

The cost function can be written as

$$\begin{aligned} J_1(\boldsymbol{w}(n)) &= E[e^2(n)] = E[v^2(n)] + E[\boldsymbol{\Delta w}^T(n)\boldsymbol{u}(n)\boldsymbol{u}^T(n)\boldsymbol{\Delta w}(n)] \\ &= \sigma_v^2 + E[E[\boldsymbol{\Delta w}^T(n)\boldsymbol{u}(n)\boldsymbol{u}^T(n)\boldsymbol{\Delta w}(n)|\boldsymbol{\Delta w}(n)]] \\ &= J_{\min} + E[\boldsymbol{\Delta w}^T(n)\boldsymbol{R}_u\boldsymbol{\Delta w}(n)] \ . \end{aligned} \tag{3.22}$$

When viewed as a function of n, the cost function is called the learning curve.

How to Analyse Adaptive Filters

Adaptive filters can be analysed in several ways.

Transient Performance How does the filter handle an abrupt change in the statistics of $u(n)$, and how fast does it converge to steady-state?

Steady-state Performance How does the filter perform in a WSS environment after all transients have died out?

Tracking Performance How does the filter handle slow variations in the statistics of $u(n)$ and/or $d(n)$?

Numerical Precision Effects What happens when the filter is implemented on a finite precision computer?

Computational Complexity How much processing time does the algorithm require?

Important Values and Functions

When analysing adaptive filters, we typically quantify their performance in terms of the following values and functions.

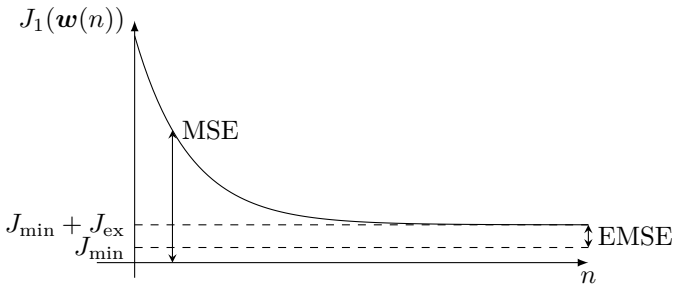

Figure 3.3: The cost function (same as the MSE), the minimum MSE, and the excess MSE.

$$J_1(\boldsymbol{w}(n)) \quad \text{(Mean-Square Error (MSE))} \tag{3.23}$$
$$J_1(\boldsymbol{w}_o) = J_{\min} \quad \text{(Minimum MSE (MMSE))} \tag{3.24}$$
$$J_{\text{ex}} = J_1(\boldsymbol{w}(\infty)) - J_{\min} \quad \text{(Excess MSE (EMSE))} \tag{3.25}$$
$$\mathcal{M} = \frac{J_{\text{ex}}}{J_{\min}} \quad \text{(Misadjustment)} \tag{3.26}$$
$$E[\|\boldsymbol{\Delta w}(\infty)\|^2] \quad \text{(Mean-Square Deviation (MSD))} \tag{3.27}$$
$$\lambda_{\max} \quad \text{(Maximum Eigenvalue of } \boldsymbol{R}_u\text{)} \tag{3.28}$$
$$\lambda_{\min} \quad \text{(Minimum Eigenvalue of } \boldsymbol{R}_u\text{)} \tag{3.29}$$
$$\kappa(\boldsymbol{R}_u) = \frac{\lambda_{\max}}{\lambda_{\min}} \quad \text{(Condition number of } \boldsymbol{R}_u\text{)} \tag{3.30}$$

In adaptive filtering, some people use the term eigenvalue spread $\chi(\boldsymbol{R}_u)$ instead of the condition number. The MSE, the EMSE, the MSD, and the misadjustment depend on the adaptive filtering algorithm. Conversely, the remaining parameters depend on the desired signal and the input signal and are thus independent of the adaptive filtering algorithm. In Fig. 3.3, some of the quantities are shown.

LMS Revisited

Recall the steepest descent algorithm

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) - \frac{\mu}{2}\boldsymbol{g}(\boldsymbol{w}(n)) \tag{3.31}$$

where

$$\boldsymbol{g}(\boldsymbol{w}(n)) = 2\boldsymbol{R}_u\boldsymbol{w}(n) - 2\boldsymbol{r}_{ud} = -2E[\boldsymbol{u}(n)e(n)] \tag{3.32}$$

is the gradient of $J_1(\boldsymbol{w}(n))$. Replacing the gradient with the simple estimate

$$\hat{\boldsymbol{g}}(\boldsymbol{w}(n)) = -2\boldsymbol{u}(n)e(n) \tag{3.33}$$

leads to the LMS algorithm

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu\boldsymbol{u}(n)e(n) \tag{3.34}$$
$$= (\boldsymbol{I} - \mu\boldsymbol{u}(n)\boldsymbol{u}^T(n))\boldsymbol{w}(n) + \mu\boldsymbol{u}(n)d(n)\ . \tag{3.35}$$

In order to analyse the LMS algorithm, we use the analysis model in Sec. 3.2.1.

Transient Analysis

We would like that

$$\lim_{n\to\infty} E[\Delta\boldsymbol{w}(n)] = 0 \qquad \text{(Convergence in mean)}$$
$$\lim_{n\to\infty} E[\|\Delta\boldsymbol{w}(n)\|^2] = c < \infty \qquad \text{(Convergence in mean-square)}$$

Convergence in the Mean

We have that

$$\boldsymbol{\Delta w}(n+1) = (\boldsymbol{I} - \mu\boldsymbol{u}(n)\boldsymbol{u}^T(n))\boldsymbol{\Delta w}(n) + \mu\boldsymbol{u}(n)v(n) \tag{3.36}$$

from which we get

$$E[\boldsymbol{\Delta w}(n+1)] = (\boldsymbol{I} - \mu \boldsymbol{R}_u) E[\boldsymbol{\Delta w}(n)] \ . \tag{3.37}$$

Thus, the LMS algorithm is stable in the mean if (same as for SD)

$$0 < \mu < \frac{2}{\lambda_{\max}} \ . \tag{3.38}$$

Since $\lambda_{\max}$ is unknown and hard to estimate, we can bound it by

$$\lambda_{\max} \leq \mathrm{tr}(\boldsymbol{R}_u) = M E[u^2(n)] \leq M S_{\max} \ . \tag{3.39}$$

Estimating the power $E[u^2(n)]$ or the maximum is the power spectral density $S_{\max}$ is fairly easy.

Convergence in the Mean-Square

We assume that $\boldsymbol{u}(n)$ has a Gaussian distribution with zero-mean and covariance matrix $\boldsymbol{R}_u$. Then, as we have shown in Appendix B, we get that the LMS algorithm is stable in the mean-square if the step-size satisfies

$$f(\mu) = \frac{\mu}{2} \sum_{m=1}^{M} \frac{\lambda_m}{1 - \mu \lambda_m} < 1 \ . \tag{3.40}$$

For small step-sizes $\mu \ll 1/\lambda_{\max}$, we have

$$f(\mu) \approx \frac{\mu}{2} \sum_{m=1}^{M} \lambda_m = \frac{\mu}{2} \mathrm{tr}(\boldsymbol{R}_u) < 1 \ . \tag{3.41}$$

Thus, stability in the mean-square requires that the step-size satisfies

$$0 < \mu < \frac{2}{\mathrm{tr}(\boldsymbol{R}_u)} \ . \tag{3.42}$$

Steady-State Analysis

From Appendix B, we have that

EMSE:
$$J_{\mathrm{ex}} = J_1(\boldsymbol{w}(\infty)) - J_{\min} = J_{\min} \frac{f(\mu)}{1 - f(\mu)} \tag{3.43}$$
$$\approx \frac{\mu}{2} J_{\min} \mathrm{tr}(\boldsymbol{R}_u) \tag{3.44}$$

Misadjustment:
$$\mathcal{M} = \frac{J_{\mathrm{ex}}}{J_{\min}} = \frac{f(\mu)}{1 - f(\mu)} \tag{3.45}$$
$$\approx \frac{\mu}{2} \mathrm{tr}(\boldsymbol{R}_u) \tag{3.46}$$

MSD:
$$E[\|\boldsymbol{\Delta w}(\infty)\|^2] \approx \frac{\mu}{2} J_{\min} M \tag{3.47}$$

Here, the approximations are valid for small step-sizes. We see that the EMSE is approximately proportional to the input power. This is a problem and referred to as *gradient noise amplification*.

Normalised LMS Adaptive Filters

Recall that we can bound the maximum eigenvalue by

$$\lambda_{\max} \leq \mathrm{tr}(\boldsymbol{R}_u) = ME[u^2(n)] \ . \tag{3.48}$$

An estimate of the power could be

$$\hat{E}[u^2(n)] = \frac{1}{M}\boldsymbol{u}^T(n)\boldsymbol{u}(n) = \frac{1}{M}\|\boldsymbol{u}(n)\|^2 \ . \tag{3.49}$$

This leads to the following bound on the step-size

$$0 < \mu < \frac{2}{\|\boldsymbol{u}(n)\|^2} \quad \Longleftrightarrow \quad 0 < \mu\|\boldsymbol{u}(n)\|^2 < 2 \ . \tag{3.50}$$

We may define the time-varying step-size

$$\mu(n) = \frac{\beta}{\|\boldsymbol{u}(n)\|^2} \ , \qquad 0 < \beta < 2 \ . \tag{3.51}$$

Then, the LMS algorithm is

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \frac{\beta}{\|\boldsymbol{u}(n)\|^2}\boldsymbol{u}(n)e(n) \tag{3.52}$$

which is called the normalised LMS (NLMS) algorithm. The NLMS bypasses the problem of gradient noise amplification by normalising the gradient estimate by the input power. However, this normalisation causes numerical problems when the input power is close to zero. In order to avoid this, we introduce the regularisation parameter ϵ, which is a small positive constant, into the NLMS algorithm

$$\boxed{\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \frac{\beta}{\epsilon + \|\boldsymbol{u}(n)\|^2}\boldsymbol{u}(n)e(n)} \ . \tag{3.53}$$

Fast Computation

The power estimate may be computed very efficiently by writing it in a recursive manner

$$\begin{aligned}\|\boldsymbol{u}(n)\|^2 &= \sum_{m=0}^{M-1} u^2(n-m) = u^2(n) - u^2(n-M) + \sum_{m=1}^{M} u^2(n-m) \\ &= u^2(n) - u^2(n-M) + \|\boldsymbol{u}(n-1)\|^2 \ .\end{aligned} \tag{3.54}$$

It should be noted that the use of this recursion can be problematic in practice due to accumulated rounding errors [7, p. 227]. These rounding errors may potentially cause the norm to be negative.

Transient Analysis

In order to analyse the NLMS algorithm, we use the analysis model in Sec. 3.2.1. If $\sigma_v^2 = 0$, the NLMS algorithm converges in the mean and the mean-square if [8, p. 325]

$$\boxed{0 < \beta < 2} \ . \tag{3.55}$$

Moreover, the optimal step-size is

$$\beta_o = 1 \ . \tag{3.56}$$

If $\sigma_v^2 > 0$, the optimal step-size is more complicated and can be found in [8, p. 325].

Steady-State Analysis

Assuming the analysis model in Sec. 3.2.1, it can be shown that [7, pp. 300–302,p. 474]

$$\text{EMSE:} \quad J_{\text{ex}} = J_1(\boldsymbol{w}(\infty)) - J_{\min} \approx \frac{\beta}{2} J_{\min} \text{tr}(\boldsymbol{R}_u) E\left[\frac{1}{\|\boldsymbol{u}(n)\|^2}\right] \geq \frac{\beta}{2} J_{\min} \tag{3.57}$$

$$\text{Misadjustment:} \quad \mathcal{M} = \frac{J_{\text{ex}}}{J_{\min}} \approx \frac{\beta}{2} \text{tr}(\boldsymbol{R}_u) E\left[\frac{1}{\|\boldsymbol{u}(n)\|^2}\right] \geq \frac{\beta}{2} \tag{3.58}$$

$$\text{MSD:} \quad E[\|\boldsymbol{\Delta w}(\infty)\|^2] \approx \frac{\beta}{2} J_{\min} E\left[\frac{1}{\|\boldsymbol{u}(n)\|^2}\right] \geq \frac{\beta J_{\min}}{2\text{tr}(\boldsymbol{R}_u)} \tag{3.59}$$

where the inequality follows from that $E[x^{-1}] \geq E[x]^{-1}$. The approximations are valid for small values of β and ϵ. Note that the approximation of the misadjustment no longer depends on the input power. This is a consequence of the normalisation of the LMS gradient with $\|\boldsymbol{u}(n)\|^2$. Moreover, it should be noted that there exist several other approximations to the EMSE and the MSD than presented above [7, p. 474].

Computational Cost

Table 3.1 shows the computational cost of the NLMS algorithm in terms of the number of multiplications, additions or subtractions, and divisions. From the table, we see that the total number of flops is $6M + 2$. If we use the fast computation of the normalisation constant, the computational cost is reduced to $4M + 7$ flops as shown in Table 3.2. Thus, the NLMS algorithm has a linear complexity in the filter length $\mathcal{O}(M)$ for both versions of the NLMS algorithm.

Term	$\times$	$+$ or $-$	$/$
$\|\boldsymbol{u}(n)\|^2$	M	$M-1$	
$\boldsymbol{u}^T(n)\boldsymbol{w}(n)$	M	$M-1$	
$e(n) = d(n) - \boldsymbol{u}^T(n)\boldsymbol{w}(n)$		1	
$\beta e(n)/(\epsilon + \|\boldsymbol{u}(n)\|^2)$	1	1	1
$\beta \boldsymbol{u}(n) e(n)/(\epsilon + \|\boldsymbol{u}(n)\|^2)$	M		
$\boldsymbol{w}(n) + \beta \boldsymbol{u}(n) e(n)/(\epsilon + \|\boldsymbol{u}(n)\|^2)$		M	
Total	$3M+1$	$3M$	1

Table 3.1: Computational cost of the NLMS algorithm.

Another Derivation of the NLMS Algorithm

We have two requirements to the adaptive filter.

1. If we filter the input sample at time n through the filter at time $n+1$, the error should be zero. That is,
$$d(n) - \boldsymbol{u}^T(n)\boldsymbol{w}(n+1) = 0 \ . \tag{3.60}$$
There are an infinite number of solutions that satisfy this requirement if $M > 1$.

2. Among all the filter coefficient vectors $\boldsymbol{w}(n+1)$ satisfying the first requirements, we select the vector resulting in the smallest change from iteration n to $n+1$. That is, $\|\boldsymbol{w}(n+1) - \boldsymbol{w}(n)\|^2$ should be as small as possible.

Term	$\times$	$+$ or $-$	$/$
$\|\boldsymbol{u}(n)\|^2$	2	2	
$\boldsymbol{u}^T(n)\boldsymbol{w}(n)$	M	$M-1$	
$e(n) = d(n) - \boldsymbol{u}^T(n)\boldsymbol{w}(n)$		1	
$\beta e(n)/(\epsilon + \|\boldsymbol{u}(n)\|^2)$	1	1	1
$\beta \boldsymbol{u}(n)e(n)/(\epsilon + \|\boldsymbol{u}(n)\|^2)$	M		
$\boldsymbol{w}(n) + \beta \boldsymbol{u}(n)e(n)/(\epsilon + \|\boldsymbol{u}(n)\|^2)$		M	
Total	$2M+3$	$2M+3$	1

Table 3.2: Computational cost of the NLMS algorithm with fast update of the normalisation factor.

These two requirements lead to the following constrained minimisation problem

$$\begin{aligned} \min_{\boldsymbol{w}(n+1)} \quad & f(\boldsymbol{w}(n+1)) = \|\boldsymbol{w}(n+1) - \boldsymbol{w}(n)\|^2 \\ \text{s.t.} \quad & h(\boldsymbol{w}(n+1)) = d(n) - \boldsymbol{u}^T(n)\boldsymbol{w}(n+1) = 0 \end{aligned} \tag{3.61}$$

This optimisation problem has a quadratic cost function and a single linear equality constraint, and it is therefore on the same form as Eq. (3.8) with

$$\boldsymbol{x} = \boldsymbol{w}(n+1) \tag{3.62}$$
$$\boldsymbol{A} = -\boldsymbol{u}^T(n) \tag{3.63}$$
$$\boldsymbol{b} = -d(n) \tag{3.64}$$
$$\boldsymbol{P} = 2\boldsymbol{I} \tag{3.65}$$
$$\boldsymbol{q} = -2\boldsymbol{w}(n) \tag{3.66}$$
$$\boldsymbol{r} = \|\boldsymbol{w}(n)\|^2 \ . \tag{3.67}$$

Thus, from Eq. (3.12) and Eq. (3.13), we readily obtain the solution to the optimisation problem in Eq. (3.61). The solution is given by

$$\lambda = -\frac{2}{\|\boldsymbol{u}(n)\|^2} e(n) \tag{3.68}$$
$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \frac{1}{\|\boldsymbol{u}(n)\|^2} \boldsymbol{u}(n)e(n) \ . \tag{3.69}$$

The last equation is identical to the NLMS algorithm with $\beta = 1$ and $\epsilon = 0$.

Affine Projection Adaptive Filters

We may extend the requirements to the adaptive filter in the following way.

1. If we filter the input sample at time $n-k$ for $k = 0, 1, \ldots, K-1 < M$ through the filter at time $n+1$, the corresponding errors should be zero. That is,

$$d(n-k) - \boldsymbol{u}^T(n-k)\boldsymbol{w}(n+1) = 0 \ , \quad k = 0, 1, \ldots, K-1 < M \ . \tag{3.70}$$

   If we define

$$\boldsymbol{U}(n) = \begin{bmatrix} \boldsymbol{u}(n) & \boldsymbol{u}(n-1) & \cdots & \boldsymbol{u}(n-K+1) \end{bmatrix} \tag{3.71}$$
$$\boldsymbol{d}(n) = \begin{bmatrix} d(n) & d(n-1) & \cdots & d(n-K+1) \end{bmatrix}^T \ , \tag{3.72}$$

we may write

$$\boldsymbol{d}(n) - \boldsymbol{U}^T(n)\boldsymbol{w}(n+1) = \boldsymbol{0} \ . \tag{3.73}$$

If $K = M$ and $\boldsymbol{U}(n)$ has full rank, there is only one filter coefficient vector $\boldsymbol{w}(n+1)$ satisfying this requirement. It is given by

$$\boldsymbol{w}(n+1) = \boldsymbol{U}^{-T}(n)\boldsymbol{d}(n) \ . \tag{3.74}$$

When this is not the case, there are an infinite number of solutions that satisfy this requirement, and we therefore need a second requirement.

2. Among all the filter coefficient vectors $\boldsymbol{w}(n+1)$ satisfying the first requirements, we select the vector resulting in the smallest change from iteration n to $n+1$. That is, $\|\boldsymbol{w}(n+1) - \boldsymbol{w}(n)\|^2$ should be as small as possible.

Note that for $K = 1$, the requirements are identical to the requirements leading to the NLMS algorithm. The affine projections algorithm (APA) can therefore be viewed as a generalisation of the NLMS algorithm. We may formulate the requirements as a constrained optimisation problem

$$\begin{aligned} \min_{\boldsymbol{w}(n+1)} \quad & f(\boldsymbol{w}(n+1)) = \|\boldsymbol{w}(n+1) - \boldsymbol{w}(n)\|^2 \\ \text{s.t.} \quad & \boldsymbol{h}(\boldsymbol{w}(n+1)) = \boldsymbol{d}(n) - \boldsymbol{U}^T(n)\boldsymbol{w}(n+1) = \boldsymbol{0} \end{aligned} \tag{3.75}$$

This optimisation problem has a quadratic cost function and $K \leq M$ linear equality constraints, and it is therefore on the same form as Eq. (3.8) with

$$\boldsymbol{x} = \boldsymbol{w}(n+1) \tag{3.76}$$
$$\boldsymbol{A} = -\boldsymbol{U}^T(n) \tag{3.77}$$
$$\boldsymbol{b} = -\boldsymbol{d}(n) \tag{3.78}$$
$$\boldsymbol{P} = 2\boldsymbol{I} \tag{3.79}$$
$$\boldsymbol{q} = -2\boldsymbol{w}(n) \tag{3.80}$$
$$\boldsymbol{r} = \|\boldsymbol{w}(n)\|^2 \ . \tag{3.81}$$

Thus, from Eq. (3.12) and Eq. (3.13), we readily obtain the solution to the optimisation problem in Eq. (3.75). The solution is given by

$$\boldsymbol{\lambda} = -2\left(\boldsymbol{U}^T(n)\boldsymbol{U}(n)\right)^{-1}\boldsymbol{e}(n) \tag{3.82}$$
$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \boldsymbol{U}(n)\left(\boldsymbol{U}^T(n)\boldsymbol{U}(n)\right)^{-1}\boldsymbol{e}(n) \ . \tag{3.83}$$

From this result, we see that the APA reduces to the NLMS algorithm for $K = 1$. Usually, we add a regularisation parameter ϵ and a step-size parameter β to the algorithm and obtain

$$\boxed{\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \beta\boldsymbol{U}(n)\left(\epsilon\boldsymbol{I} + \boldsymbol{U}^T(n)\boldsymbol{U}(n)\right)^{-1}\boldsymbol{e}(n)} \ . \tag{3.84}$$

As for the NLMS algorithm, ϵ is a small positive value which bypasses numerical problems when $\boldsymbol{U}^T(n)\boldsymbol{U}(n)$ is ill-conditioned.

For $K > 1$, The computational complexity of the APA is higher than that of the NLMS algorithm since we have to invert a $K \times K$ matrix. Although there exist fast ways of doing this [8, pp. 339–340], the APA is more expensive than the NLMS algorithm.

Transient Analysis

In order to analyse the APA, we use the analysis model in Sec. 3.2.1. If $\sigma_v^2 = 0$, the APA converges in the mean and the mean-square if [8, p. 337]

$$\boxed{0 < \beta < 2} \,. \tag{3.85}$$

Moreover, the optimal step-size is

$$\beta_o = 1 \,. \tag{3.86}$$

If $\sigma_v^2 > 0$, the optimal step-size is more complicated and can be found in [8, p. 337].

Steady-State Analysis

Assuming the analysis model in Sec. 3.2.1, it can be shown that [7, p. 327]

$$\begin{aligned} \text{EMSE:} \qquad J_{\text{ex}} &= J_1(\boldsymbol{w}(\infty)) - J_{\min} \\ &\approx \frac{\beta}{2} J_{\min} \text{tr}(\boldsymbol{R}_u) E\left[\frac{K}{\|\boldsymbol{u}(n)\|^2}\right] \geq \frac{\beta}{2} J_{\min} K \end{aligned} \tag{3.87}$$

$$\text{Misadjustment:} \qquad \mathcal{M} = \frac{J_{\text{ex}}}{J_{\min}} \approx \frac{\beta}{2} \text{tr}(\boldsymbol{R}_u) E\left[\frac{K}{\|\boldsymbol{u}(n)\|^2}\right] \geq \frac{\beta K}{2} \tag{3.88}$$

where the inequality follows from that $E[x^{-1}] \geq E[x]^{-1}$. The approximations are valid for small values of β and ϵ. It should be noted that there exist several other approximations to the EMSE than presented above [7, p. 325]. For more accurate expressions and for an expression for the mean-square deviation, see [7, pp. 510–512] and [9].

Computational Cost

Table 3.3 shows the computational cost of the APA algorithm in terms of the number of multiplications and additions or subtractions. We have assumed that the cost of inverting a $K \times K$ matrix is K^3 multiplications and additions [7, p. 240]. From the table, we see that the total number of flops is $2M(K^2+2K)+2K^3+K^2+M$. Thus, the APA has a complexity of $\mathcal{O}(MK^2)$. Note that there exist faster ways of implementing the APA [8, pp. 339-340].

Term	$\times$	$+$ or $-$
$\boldsymbol{U}^T(n)\boldsymbol{w}(n)$	MK	$(M-1)K$
$\boldsymbol{e}(n) = \boldsymbol{d}(n) - \boldsymbol{U}^T(n)\boldsymbol{w}(n)$		K
$\boldsymbol{U}^T(n)\boldsymbol{U}(n)$	MK^2	$(M-1)K^2$
$\epsilon\boldsymbol{I} + \boldsymbol{U}^T(n)\boldsymbol{U}(n)$		K
$(\epsilon\boldsymbol{I} + \boldsymbol{U}^T(n)\boldsymbol{U}(n))^{-1}$	K^3	K^3
$(\epsilon\boldsymbol{I} + \boldsymbol{U}^T(n)\boldsymbol{U}(n))^{-1}\boldsymbol{e}(n)$	K^2	$K(K-1)$
$\boldsymbol{U}(n)(\epsilon\boldsymbol{I} + \boldsymbol{U}^T(n)\boldsymbol{U}(n))^{-1}\boldsymbol{e}(n)$	MK	$M(K-1)$
$\beta\boldsymbol{U}(n)(\epsilon\boldsymbol{I} + \boldsymbol{U}^T(n)\boldsymbol{U}(n))^{-1}\boldsymbol{e}(n)$	M	
$\boldsymbol{w}(n) + \beta\boldsymbol{U}(n)(\epsilon\boldsymbol{I} + \boldsymbol{U}^T(n)\boldsymbol{U}(n))^{-1}\boldsymbol{e}(n)$		M
Total	$M(K^2+2K+1)+K^3+K^2$	$M(K^2+2K)+K^3$

Table 3.3: Computational cost of the APA.

Lecture 4

Recursive Least-Squares Adaptive Filters

Review of the Basics

The Matrix Inversion Lemma

Let $\boldsymbol{X}$, $\boldsymbol{Y}$, $\boldsymbol{X}+\boldsymbol{UYV}$, and $\boldsymbol{Y}^{-1}+\boldsymbol{VX}^{-1}\boldsymbol{U}$ all be non-singular matrices. By equating element $(1,1)$ of the two block matrices in Eq. (3.1) and Eq. (3.2) and setting

$$\boldsymbol{X} = \boldsymbol{A} \tag{4.1}$$
$$\boldsymbol{U} = \boldsymbol{B} \tag{4.2}$$
$$\boldsymbol{V} = \boldsymbol{C} \tag{4.3}$$
$$\boldsymbol{Y} = -\boldsymbol{D}^{-1} \ , \tag{4.4}$$

we obtain the matrix inversion lemma

$$(\boldsymbol{X}+\boldsymbol{UYV})^{-1} = \boldsymbol{X}^{-1} - \boldsymbol{X}^{-1}\boldsymbol{U}(\boldsymbol{Y}^{-1}+\boldsymbol{VX}^{-1}\boldsymbol{U})^{-1}\boldsymbol{VX}^{-1} \ . \tag{4.5}$$

The matrix inversion lemma is also sometimes called the *Woodbury matrix identity* or the *Woodbury's identity*.

Method of Least-Squares

The adaptive filtering problem is shown in Fig. 4.1. From the figure, we have that

$u(n)$: zero-mean input signal

$w_m(n)$: M-tap FIR-filter with impulse response $w_0(n), w_1(n), \ldots, w_{M-1}(n)$

$y(n)$: output signal given by $y(n) = \sum_{m=0}^{M-1} w_m(n)u(n-m)$

$d(n)$: zero-mean desired signal

$e(n)$: error signal

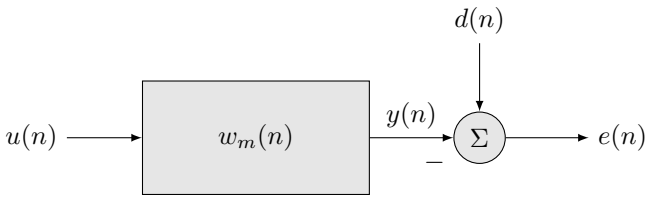

Figure 4.1: Block diagram of adaptive filtering in a non-WSS environment.

Define

$$\boldsymbol{w}(n) = \begin{bmatrix} w_0(n) & w_1(n) & \cdots & w_{M-1}(n) \end{bmatrix}^T \tag{4.6}$$

$$\boldsymbol{u}(n) = \begin{bmatrix} u(n) & u(n-1) & \cdots & u(n-M+1) \end{bmatrix}^T \ . \tag{4.7}$$

For $i = 1, 2, \ldots, n$ with $n \geq M$, define

$$\boldsymbol{A}(n) = \begin{bmatrix} \boldsymbol{u}(1) & \boldsymbol{u}(2) & \cdots & \boldsymbol{u}(n) \end{bmatrix}^T \tag{4.8}$$

$$\boldsymbol{d}(n) = \begin{bmatrix} d(1) & d(2) & \cdots & d(n) \end{bmatrix}^T \tag{4.9}$$

$$\boldsymbol{e}(n) = \begin{bmatrix} e(1) & e(2) & \cdots & e(n) \end{bmatrix}^T \ . \tag{4.10}$$

Then

$$\boldsymbol{e}(n) = \boldsymbol{d}(n) - \boldsymbol{A}(n)\boldsymbol{w}(n) \quad \Longleftrightarrow \quad \boldsymbol{d}(n) = \boldsymbol{A}(n)\boldsymbol{w}(n) + \boldsymbol{e}(n) \ . \tag{4.11}$$

Note that we have made $\boldsymbol{A}(n)$, $\boldsymbol{d}(n)$, and $\boldsymbol{e}(n)$ time-dependent in order to emphasise that we are here concerned with an online algorithm. Therefore, we also formulate the squared error cost function $J_2(\boldsymbol{w}(n))$ in a slightly different way compared to Eq. (1.29). Here, we define it as

$$J_2(\boldsymbol{w}(n)) = \sum_{i=1}^{n} e^2(i) = \boldsymbol{e}^T(n)\boldsymbol{e}(n) \ . \tag{4.12}$$

In the method of least-squares, we wish to minimise $J_2(\boldsymbol{w}(n))$ which we can write as

$$\begin{aligned} J_2(\boldsymbol{w}(n)) &= \boldsymbol{e}^T(n)\boldsymbol{e}(n) = (\boldsymbol{d}(n) - \boldsymbol{A}(n)\boldsymbol{w}(n))^T(\boldsymbol{d}(n) - \boldsymbol{A}(n)\boldsymbol{w}(n)) \\ &= \boldsymbol{d}^T(n)\boldsymbol{d}(n) + \boldsymbol{w}^T(n)\boldsymbol{A}^T(n)\boldsymbol{A}(n)\boldsymbol{w}(n) - 2\boldsymbol{w}^T(n)\boldsymbol{A}^T(n)\boldsymbol{d}(n) \ . \end{aligned} \tag{4.13}$$

The minimiser

$$\boldsymbol{w}_o(n) = (\boldsymbol{A}^T(n)\boldsymbol{A}(n))^{-1}\boldsymbol{A}^T(n)\boldsymbol{d}(n) \tag{4.14}$$

is referred to as the least-squares solution, and it is the unique solution to the normal equations

$$\boldsymbol{A}^T(n)\boldsymbol{A}(n)\boldsymbol{w}(n) = \boldsymbol{A}^T(n)\boldsymbol{d}(n) \ , \tag{4.15}$$

provided that $\boldsymbol{A}(n)$ has full rank.

Weighted Least-Squares

When the statistics of $u(n)$ and/or $d(n)$ is time dependent, the minimisation of the squared error $J_2(\boldsymbol{w}(n))$ may fail to give a good estimate at time n since all data affect the value of $J_2(\boldsymbol{w}(n))$

with the same weight. Ideally, we would like that the new data are assigned a larger weight than the old data. In order to do this, we reformulate $J_2(\boldsymbol{w}(n))$ as a weighted cost function

$$J_\beta(\boldsymbol{w}(n)) = \sum_{i=1}^{n} \beta(n,i)e^2(i) = \boldsymbol{e}^T(n)\boldsymbol{B}(n)\boldsymbol{e}(n) \tag{4.16}$$

where $\beta(n,i)$ contains the weight pertaining to the i'th error at time n, and $\boldsymbol{B}(n)$ is a diagonal matrix given by

$$\boldsymbol{B}(n) = \text{diag}(\beta(n,1), \beta(n,2), \ldots, \beta(n,n)) \ . \tag{4.17}$$

We use the five step recipe in Sec. 1.1.2 to minimise $J_\beta(\boldsymbol{w}(n))$ w.r.t $\boldsymbol{w}(n)$.

1. Construct the cost function

$$\begin{aligned} J_\beta(\boldsymbol{w}(n)) &= \boldsymbol{e}^T(n)\boldsymbol{B}(n)\boldsymbol{e}(n) = (\boldsymbol{d}(n) - \boldsymbol{A}(n)\boldsymbol{w}(n))^T\boldsymbol{B}(n)(\boldsymbol{d}(n) - \boldsymbol{A}(n)\boldsymbol{w}(n)) \\ &= \boldsymbol{d}(n)^T\boldsymbol{B}(n)\boldsymbol{d}(n) + \boldsymbol{w}^T(n)\boldsymbol{\Phi}(n)\boldsymbol{w}(n) - 2\boldsymbol{w}^T(n)\boldsymbol{\varphi}(n) \end{aligned} \tag{4.18}$$

   where we have defined $\boldsymbol{\Phi}(n)$ and $\boldsymbol{\varphi}(n)$ as

$$\boldsymbol{\Phi}(n) = \boldsymbol{A}^T(n)\boldsymbol{B}(n)\boldsymbol{A}(n) \tag{4.19}$$

$$\boldsymbol{\varphi}(n) = \boldsymbol{A}^T(n)\boldsymbol{B}(n)\boldsymbol{d}(n) \ . \tag{4.20}$$

   We refer to $\boldsymbol{\Phi}(n)$ and $\boldsymbol{\varphi}(n)$ as the correlation matrix and the cross-correlation vector, respectively, since they are scaled and weighted estimates of $\boldsymbol{R}_u$ and $\boldsymbol{r}_{ud}$.

2. Find the gradient

$$\boldsymbol{g}(\boldsymbol{w}(n)) = (\boldsymbol{\Phi}(n) + \boldsymbol{\Phi}^T(n))\boldsymbol{w}(n) - 2\boldsymbol{\varphi}(n) = 2\boldsymbol{\Phi}(n)\boldsymbol{w}(n) - 2\boldsymbol{\varphi}(n) \tag{4.21}$$

3. Solve $\boldsymbol{g}(\boldsymbol{w}(n)) = \boldsymbol{0}$ for $\boldsymbol{w}(n)$

$$\begin{aligned} &\boldsymbol{g}(\boldsymbol{w}(n)) = 2\boldsymbol{\Phi}(n)\boldsymbol{w}(n) - 2\boldsymbol{\varphi}(n) \\ &\Updownarrow \\ &\boxed{\boldsymbol{\Phi}(n)\boldsymbol{w}(n) = \boldsymbol{\varphi}(n)} \qquad \text{(Weighted Normal Equations)} \qquad (4.22) \\ &\Updownarrow \\ &\boldsymbol{w}_o(n) = \boldsymbol{\Phi}^{-1}(n)\boldsymbol{\varphi}(n) \qquad \text{(If } \boldsymbol{\Phi}(n) \text{ is invertible)} \qquad (4.23) \end{aligned}$$

4. Find the Hessian

$$\boldsymbol{H}(\boldsymbol{w}) = 2\boldsymbol{\Phi}(n) \tag{4.24}$$

   which is p.d. for all $\boldsymbol{w}(n)$ if $\boldsymbol{A}(n)$ has full rank and $\beta(n,i) > 0$ for all $n \geq i > 0$.

5. This implies that

   - $J_\beta(\boldsymbol{w}(n))$ is a convex function, and
   - $\boldsymbol{w}_o(n) = \boldsymbol{\Phi}^{-1}(n)\boldsymbol{\varphi}(n)$ is the global minimiser.

The solution $\boldsymbol{w}_o(n)$ is often referred to as the weighted least-squares solution.

Estimation of the (Cross-)Correlation

Comparing the weighted normal equations with the Wiener-Hopf equations, we see that

$$\hat{\boldsymbol{R}}_u(n) = c(n,\beta)\boldsymbol{\Phi}(n) = c(n,\beta)\boldsymbol{A}^T(n)\boldsymbol{B}(n)\boldsymbol{A}(n) = c(n,\beta)\sum_{i=1}^{n}\beta(n,i)\boldsymbol{u}(i)\boldsymbol{u}^T(i) \tag{4.25}$$

$$\hat{\boldsymbol{r}}_{ud}(n) = c(n,\beta)\boldsymbol{\varphi}(n) = c(n,\beta)\boldsymbol{A}^T(n)\boldsymbol{B}(n)\boldsymbol{d}(n) = c(n,\beta)\sum_{i=1}^{n}\beta(n,i)\boldsymbol{u}(i)d(i) \tag{4.26}$$

are the estimates of the correlation matrix and the cross-correlation vector, respectively. The constant $c(n,\beta)$ depends on n and the weighting function $\beta(n,i)$, and it can be selected such that $\hat{\boldsymbol{R}}_u(n)$ and $\hat{\boldsymbol{r}}_{ud}(n)$ are unbiased estimates of $\boldsymbol{R}_u(n)$ and $\boldsymbol{r}_{ud}(n)$.

Weight Functions

We consider three simple weight functions.

Growing Window Weight Function

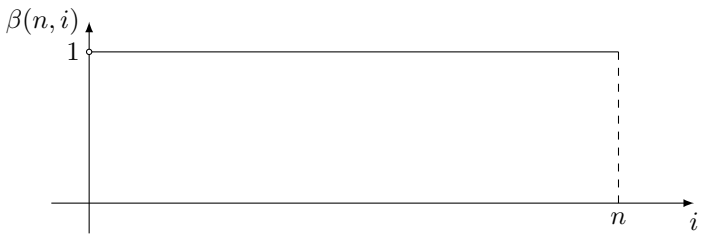

Figure 4.2: The growing window weight function.

If we select the weight function as

$$\beta(n,i) = \begin{cases} 1 & 0 < i \leq n \\ 0 & \text{otherwise} \end{cases}, \tag{4.27}$$

we obtain the growing window weight function, and it is sketched in Fig. 4.2. Selecting the growing window weight function reduces the weighted least-squares problem to the standard least-squares problem. In order to obtain unbiased estimates of $\boldsymbol{R}_u(n)$ and $\boldsymbol{r}_{ud}(n)$ in Eq. (4.25) and Eq. (4.26), respectively, we have to use

$$c(n,\beta) = \frac{1}{n}. \tag{4.28}$$

Sliding Window Weight Function

If we select the weight function as

$$\beta(n,i) = \begin{cases} 1 & n - L < i \leq n \\ 0 & \text{otherwise} \end{cases} \tag{4.29}$$

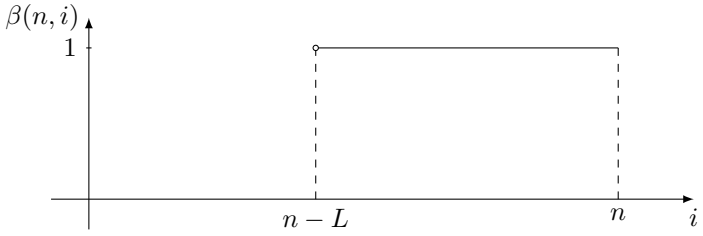

Figure 4.3: The sliding window weight function.

for $0 < L \leq n$, we obtain the sliding window weight function, and it is sketched in Fig. 4.3. If we select $L = n$, the sliding window weight function reduces to the growing window weight function. In order to obtain unbiased estimates of $\boldsymbol{R}_u(n)$ and $\boldsymbol{r}_{ud}(n)$ in Eq. (4.25) and Eq. (4.26), respectively, we have to use

$$c(n,\beta) = \frac{1}{L} \ . \tag{4.30}$$

Exponential Weight Function

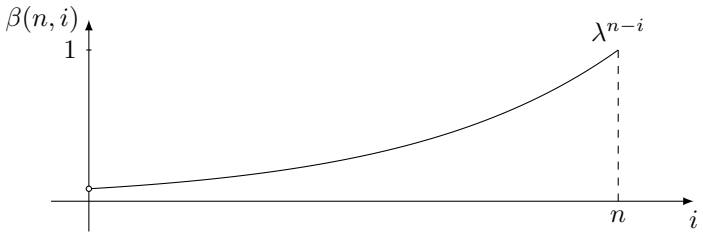

Figure 4.4: The exponential weight function.

If we select the weight function as

$$\beta(n,i) = \begin{cases} \lambda^{n-i} & 0 < i \leq n \\ 0 & \text{otherwise} \end{cases} \tag{4.31}$$

for $0 < \lambda \leq 1$, we obtain the exponential weight function, and it is sketched in Fig. 4.4. The parameter λ is called *the forgetting factor*. If we select $\lambda = 1$, the exponential weight function reduces to the growing window weight function. In order to obtain unbiased estimates of $\boldsymbol{R}_u(n)$ and $\boldsymbol{r}_{ud}(n)$ in Eq. (4.25) and Eq. (4.26), respectively, we have to use

$$c(n,\beta) = \begin{cases} \dfrac{1-\lambda}{1-\lambda^n} & 0 < \lambda < 1 \\ \dfrac{1}{n} & \lambda = 1 \end{cases} \ . \tag{4.32}$$

The Recursive Least-Squares Algorithm with an Exponential Weight Function

In an online algorithm, we have to solve the weighted normal equations

$$\boldsymbol{\Phi}(n)\boldsymbol{w}(n) = \boldsymbol{\varphi}(n) \tag{4.33}$$

for $\boldsymbol{w}(n)$ at every time index n. However, solving this equation directly as

$$\boldsymbol{w}(n) = \boldsymbol{\Phi}^{-1}(n)\boldsymbol{\varphi}(n) \tag{4.34}$$

yields a high computational complexity of the algorithm for the following two reasons.

1. The matrices $\boldsymbol{A}(n)$ and $\boldsymbol{B}(n)$, and the vector $\boldsymbol{d}(n)$ grows with n. Computing $\boldsymbol{\Phi}(n)$ and $\boldsymbol{\varphi}(n)$ directly would therefore be infeasible for an online algorithm, unless we use a weight function with a finite duration window.
2. For an M-tap FIR filter, it requires in the order of $\mathcal{O}(M^3)$ operations to solve the normal equations for the filter coefficient vector $\boldsymbol{w}(n)$.

The recursive least-squares (RLS) algorithm bypasses these two problems. Below, we consider how this is obtained for an exponential weight function, which is the most common weight function.

Recursive Computation of $\boldsymbol{\Phi}(n)$ and $\boldsymbol{\varphi}(n)$

We may compute the correlation matrix recursively by rewriting it as

$$\begin{aligned}\boldsymbol{\Phi}(n) &= \boldsymbol{A}^T(n)\boldsymbol{B}(n)\boldsymbol{A}(n) = \sum_{i=1}^{n} \lambda^{n-i}\boldsymbol{u}(i)\boldsymbol{u}^T(i) \qquad (4.35)\\ &= \lambda^0\boldsymbol{u}(n)\boldsymbol{u}^T(n) + \sum_{i=1}^{n-1} \lambda^{n-i}\boldsymbol{u}(i)\boldsymbol{u}^T(i) = \boldsymbol{u}(n)\boldsymbol{u}^T(n) + \lambda\sum_{i=1}^{n-1} \lambda^{n-1-i}\boldsymbol{u}(i)\boldsymbol{u}^T(i)\\ &= \boldsymbol{u}(n)\boldsymbol{u}^T(n) + \lambda\boldsymbol{\Phi}(n-1)\ . \qquad (4.36)\end{aligned}$$

Similarly, we may compute the cross-correlation vector recursively by rewriting it as

$$\begin{aligned}\boldsymbol{\varphi}(n) &= \boldsymbol{A}^T(n)\boldsymbol{B}(n)\boldsymbol{d}(n) = \sum_{i=1}^{n} \lambda^{n-i}\boldsymbol{u}(i)d(i) \qquad (4.37)\\ &= \lambda^0\boldsymbol{u}(n)d(n) + \sum_{i=1}^{n-1} \lambda^{n-i}\boldsymbol{u}(i)d(i) = \boldsymbol{u}(n)d(n) + \lambda\sum_{i=1}^{n-1} \lambda^{n-1-i}\boldsymbol{u}(i)d(i)\\ &= \boldsymbol{u}(n)d(n) + \lambda\boldsymbol{\varphi}(n-1)\ . \qquad (4.38)\end{aligned}$$

These recursive formulations of the correlation matrix and the cross-correlation vector clearly reduce the computational complexity and are suitable to use in an online algorithm.

Inversion of $\boldsymbol{\Phi}(n)$

For the inversion of the correlation matrix, we use the matrix inversion lemma stated in Sec. 4.1.1. Comparing the recursive formulation of the correlation matrix in Eq. (4.36) with the left side of Eq. (4.5), we obtain

$$\boldsymbol{X} = \lambda\boldsymbol{\Phi}(n-1) \tag{4.39}$$
$$\boldsymbol{U} = \boldsymbol{u}(n) \tag{4.40}$$
$$\boldsymbol{V} = \boldsymbol{u}^T(n) \tag{4.41}$$
$$\boldsymbol{Y} = 1\ . \tag{4.42}$$

Thus, invoking the matrix inversion lemma, we have that

$$\boldsymbol{\Phi}^{-1}(n) = \lambda^{-1}\boldsymbol{\Phi}^{-1}(n-1) - \lambda^{-2}\frac{\boldsymbol{\Phi}^{-1}(n-1)\boldsymbol{u}(n)\boldsymbol{u}^T(n)\boldsymbol{\Phi}^{-1}(n-1)}{1+\lambda^{-1}\boldsymbol{u}^T(n)\boldsymbol{\Phi}^{-1}(n-1)\boldsymbol{u}(n)}\ . \tag{4.43}$$

Note that the computational complexity of the right side of the equation is much lower than that of the left side of the equation when $\boldsymbol{\Phi}^{-1}(n-1)$ is known. In order to simplify the notation, we define

$$\boldsymbol{P}(n) = \boldsymbol{\Phi}^{-1}(n) \qquad \text{(inverse correlation matrix)} \tag{4.44}$$
$$\boldsymbol{k}(n) = \frac{\boldsymbol{P}(n-1)\boldsymbol{u}(n)}{\lambda+\boldsymbol{u}^T(n)\boldsymbol{P}(n-1)\boldsymbol{u}(n)} \qquad \text{(gain vector)} \tag{4.45}$$

which leads to that we may write Eq. (4.43) as

$$\boldsymbol{P}(n) = \lambda^{-1}\left[\boldsymbol{P}(n-1) - \boldsymbol{k}(n)\boldsymbol{u}^T(n)\boldsymbol{P}(n-1)\right]\ . \tag{4.46}$$

By rearranging the expression for the gain vector, we obtain

$$\begin{aligned}\boldsymbol{k}(n) &= \lambda^{-1}\left[\boldsymbol{P}(n-1) - \boldsymbol{k}(n)\boldsymbol{u}^T(n)\boldsymbol{P}(n-1)\right]\boldsymbol{u}(n)\\ &= \boldsymbol{P}(n)\boldsymbol{u}(n)\ .\end{aligned} \tag{4.47}$$

Recursive computation of $\boldsymbol{w}(n)$

We can now develop the recursive update equation of the filter coefficient vector. We have that

$$\boldsymbol{w}(n) = \boldsymbol{P}(n)\boldsymbol{\varphi}(n) \tag{4.48}$$
$$\begin{aligned}&= \boldsymbol{P}(n)\left[\boldsymbol{u}(n)d(n) + \lambda\boldsymbol{\varphi}(n-1)\right] = \lambda\boldsymbol{P}(n)\boldsymbol{P}^{-1}(n-1)\boldsymbol{w}(n-1) + \boldsymbol{k}(n)d(n)\\ &= \left[\boldsymbol{P}(n-1) - \boldsymbol{k}(n)\boldsymbol{u}^T(n)\boldsymbol{P}(n-1)\right]\boldsymbol{P}^{-1}(n-1)\boldsymbol{w}(n-1) + \boldsymbol{k}(n)d(n)\\ &= \boldsymbol{w}(n-1) - \boldsymbol{k}(n)\boldsymbol{u}^T(n)\boldsymbol{w}(n-1) + \boldsymbol{k}(n)d(n)\\ &= \boldsymbol{w}(n-1) + \boldsymbol{k}(n)\xi(n)\end{aligned} \tag{4.49}$$

where we have defined the a priori error as

$$\xi(n) = d(n) - \boldsymbol{u}^T(n)\boldsymbol{w}(n-1)\ . \tag{4.50}$$

The RLS Algorithm

The RLS algorithm may now be formulated as the following set of equations

$$\boldsymbol{\pi}(n) = \boldsymbol{P}(n-1)\boldsymbol{u}(n) \tag{4.51}$$

$$\boldsymbol{k}(n) = \frac{\boldsymbol{\pi}(n)}{\lambda + \boldsymbol{u}^T(n)\boldsymbol{\pi}(n)} \tag{4.52}$$

$$\xi(n) = d(n) - \boldsymbol{u}^T(n)\boldsymbol{w}(n-1) \tag{4.53}$$

$$\boldsymbol{w}(n) = \boldsymbol{w}(n-1) + \boldsymbol{k}(n)\xi(n) \tag{4.54}$$

$$\boldsymbol{P}(n) = \lambda^{-1}\left[\boldsymbol{P}(n-1) - \boldsymbol{k}(n)\boldsymbol{\pi}^T(n)\right] \ . \tag{4.55}$$

For an M-tap FIR filter, it requires in the order of $\mathcal{O}(M^2)$ operations to run one iteration of the RLS algorithm.

Initialisation

In order to start the RLS algorithm, we need to select values for the initial inverse correlation matrix $\boldsymbol{P}(0)$, the initial filter coefficient vector $\boldsymbol{w}(0)$, and the input samples $u(n)$ for $n = -M+1, -M+2, \ldots, 1$. Typically, we assume that

$$\boldsymbol{P}(0) = \delta^{-1}\boldsymbol{I} \tag{4.56}$$

$$\boldsymbol{w}(0) = \boldsymbol{0} \tag{4.57}$$

$$u(n) = 0 \ , \quad \text{for } -M+1 < n < 1 \ . \tag{4.58}$$

The first assumption implies that we assume that $\boldsymbol{u}(n)$ for $n < 1$ is a white random process with covariance matrix $\delta\boldsymbol{I}$. The value of δ should reflect the SNR of the input data with δ being small for a high SNR and δ being large for a low SNR [8, pp. 444–446]. This assumption introduces bias into the correlation matrix $\boldsymbol{\Phi}(n)$. However, this bias decreases to zero for an increasing n. An alternative initialisation, which does not introduce bias, is to estimate the correlation matrix and the cross-correlation vector as [10, pp. 545–546]

$$\boldsymbol{P}(0) = \left[\sum_{i=-M+1}^{0} \lambda^{-i}\boldsymbol{u}(i)\boldsymbol{u}^T(i)\right]^{-1} \tag{4.59}$$

$$\boldsymbol{\varphi}(0) = \sum_{i=-M+1}^{0} \lambda^{-i}\boldsymbol{u}(i)\boldsymbol{d}(i) \tag{4.60}$$

prior to starting the RLS algorithm at time $n = 1$. The initial value of the filter coefficient vector can be set to $\boldsymbol{w}(0) = \boldsymbol{P}(0)\boldsymbol{\varphi}(0)$. Note, that this approach requires that we know the input signal from time $n = -2M+2$ and the desired signal from time $n = -M+2$.

Selection of the Forgetting Factor

At time n, the memory of the sliding window RLS algorithm is the L newest samples indexed by $n-L+1, \ldots, n$. For the exponentially weighted RLS algorithm, the memory is controlled by the forgetting factor λ. Whereas the interpretation of L is simple, the corresponding interpretation of λ is not that intuitive when we have to investigate the memory of the exponentially weighted RLS algorithm. That is, we would like to interpret the forgetting factor as a sliding window

length. We call this window length for the effective window length and denote it by L_{eff}. A simple way of connecting L_{eff} and λ is by requiring that

$$\lim_{n\to\infty} \sum_{i=1}^{n} \lambda^{n-i} = \lim_{n\to\infty} \sum_{i=n-L_{\text{eff}}+1}^{n} 1 \ . \tag{4.61}$$

That is, when the RLS algorithm has reached steady-state, the area under the sliding window curve should equal the area under the exponential window curve. This leads to

$$\lim_{n\to\infty} \sum_{i=n-L_{\text{eff}}+1}^{n} 1 = L_{\text{eff}} \tag{4.62}$$

and

$$\lim_{n\to\infty} \sum_{i=1}^{n} \lambda^{n-i} = \lim_{n\to\infty} \sum_{k=0}^{n-1} \lambda^{k} = \lim_{n\to\infty} \sum_{k=0}^{n-1} \lambda^{k} = \lim_{n\to\infty} \frac{1-\lambda^{n}}{1-\lambda} = \frac{1}{1-\lambda} \tag{4.63}$$

where the second last equality follows for $\lambda \neq 1$ from the geometric series, and the last equality follows if $0 < \lambda < 1$. Thus, we have that

$$L_{\text{eff}} = \frac{1}{1-\lambda} \ . \tag{4.64}$$

Transient Analysis

The RLS algorithm is stable in the mean and the mean-square if $0 < \lambda \leq 1$. It may also be shown that the rate of the convergence of the RLS algorithm is typically an order of magnitude faster than the rate of the convergence of the LMS algorithm. Moreover, the rate of the convergence of the RLS algorithm is invariant to the condition number of the correlation matrix $\boldsymbol{R}_u$ of the input signal [8, p. 463, ch. 14].

Steady-State Analysis

It can be shown that [7, p. 510]

EMSE:
$$\begin{aligned} J_{\text{ex}} &= J_2(\boldsymbol{w}(\infty)) - J_{\text{min}} \\ &\approx J_{\text{min}} \frac{(1-\lambda)M}{1+\lambda-(1-\lambda)M} \end{aligned} \tag{4.65}$$

Misadjustment:
$$\mathcal{M} = \frac{J_{\text{ex}}}{J_{\text{min}}} \approx \frac{(1-\lambda)M}{1+\lambda-(1-\lambda)M} \tag{4.66}$$

MSD:
$$E[\|\boldsymbol{\Delta w}(\infty)\|^2] \approx J_{\text{ex}} \sum_{m=1}^{M} \frac{1}{\lambda_m} \tag{4.67}$$

where λ_m is the m'th eigenvalue of the correlation matrix $\boldsymbol{R}_u$ not to be confused with the forgetting factor λ. The approximations hold under certain conditions which may be found in [7, pp. 508–510].

Computational Cost

Table 4.1 shows the computational cost of the RLS algorithm in terms of the number of multiplications, additions or subtractions, and divisions. From the table, we see that the total number of flops is $5M^2+5M+1$. Thus, the RLS algorithm has a complexity of $\mathcal{O}(M^2)$. Note, that there exist faster ways of implementing the RLS algorithm [7, pp. 247]. Some of them even achieve linear complexity [11].

Term	$\times$	$+$ or $-$	$/$
$\boldsymbol{\pi}(n) = \boldsymbol{P}(n-1)\boldsymbol{u}(n)$	M^2	$M(M-1)$	
$\boldsymbol{k}(n) = \boldsymbol{\pi}(n)/(\lambda + \boldsymbol{u}^T(n)\boldsymbol{\pi}(n))$	M	M	1
$\xi(n) = d(n) - \boldsymbol{u}^T(n)\boldsymbol{w}(n-1)$	M	M	
$\boldsymbol{w}(n) = \boldsymbol{w}(n-1) + \boldsymbol{k}(n)\xi(n)$	M	M	
$\boldsymbol{P}(n) = (\boldsymbol{P}(n-1) - \boldsymbol{k}(n)\boldsymbol{\pi}^T(n))/\lambda$	M^2	M^2	M^2
Total	$2M^2+3M$	$2M^2+2M$	M^2+1

Table 4.1: Computational cost of the RLS algorithm.

Bibliography

[1] D. C. Lay, *Linear algebra and its applications*, 3rd ed. Pearson/Addison-Wesley, Sep. 2005.

[2] G. Strang, *Introduction to Linear Algebra*, 4th ed. Wellesley Cambridge Press, Feb. 2009.

[3] A. Antoniou and W.-S. Lu, *Practical Optimization: Algorithms and Engineering Applications*. Springer, Mar. 2007.

[4] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, Mar. 2004.

[5] S. M. Kay, *Intuitive Probability and Random Processes using MATLAB*. Springer Science, New York, Inc., 2006.

[6] A. Leon-Garcia, *Probability, Statistics, and Random Processes For Electrical Engineering*, 3rd ed. Prentice Hall, Jan. 2008.

[7] A. H. Sayed, *Fundamentals of Adaptive Filtering*. Wiley-IEEE Press, Jun. 2003.

[8] S. Haykin, *Adaptive Filter Theory*, 4th ed. Prentice Hall, Sep. 2001.

[9] H.-C. Shin and A. H. Sayed, "Mean-square performance of a family of affine projection algorithms," *IEEE Trans. Signal Process.*, vol. 52, no. 1, pp. 90–102, Jan. 2004.

[10] M. H. Hayes, *Statistical Digital Signal Processing and Modeling*. Wiley, 1996.

[11] D. T. M. Slock and T. Kailath, "Numerically stable fast transversal filters for recursive least squares adaptive filtering," *IEEE Trans. Signal Process.*, vol. 39, no. 1, pp. 92–114, Jan. 1991.

[12] K. B. Petersen and M. S. Pedersen, "The matrix cookbook," Online, oct 2008, version 20081110. [Online]. Available: http://www2.imm.dtu.dk/pubdb/p.php?3274

Appendix A

Summary

Fig. A.1 shows an overview over the various important concepts, equations and adaptive filters covered in these lecture notes. All relationships indicated by the arrows have been covered with the exception of the arrow from the steepest descent block to the NLMS block. The interested reader may establish this relationship by solving Problem 1 of Chapter 6 in [8].

Table A.1 shows a summary of the adaptive filters covered in these lecture notes. Most the expressions for the mean-square stability, the excess mean-square error (EMSE), the misadjustment, and the mean-square deviation (MSD) are approximations, and they can therefore only be used as a rule of thumb. For more accurate expressions see the description of the adaptive filters and the references therein.

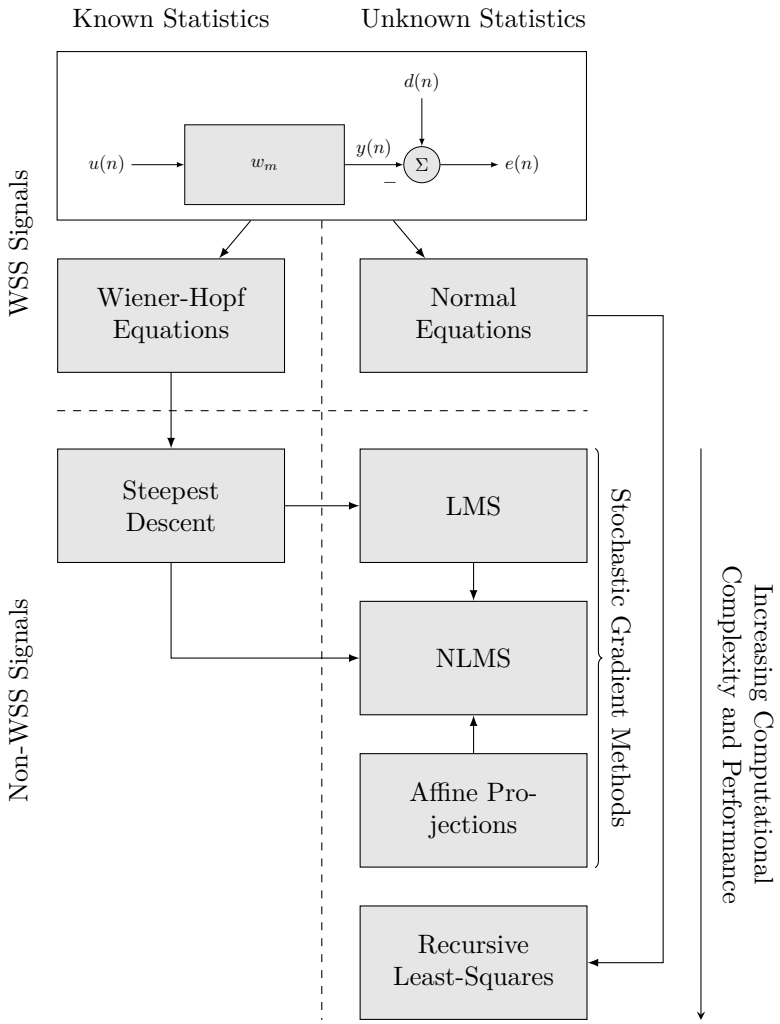

Figure A.1: Overview over the various important concepts, equations and adaptive filters covered in these lecture notes.

Name	Algorithm	Cost	Mean-Square Stability	EMSE	Misadjustment	MSD
SD	$\boldsymbol{g}(\boldsymbol{w}(n)) = 2\boldsymbol{R}_u\boldsymbol{w}(n) - 2\boldsymbol{r}_{ud}(n)$ $\boldsymbol{w}(n+1) = \boldsymbol{w}(n) - \frac{\mu}{2}\boldsymbol{g}(\boldsymbol{w}(n))$	$\mathcal{O}(M)$	$0 < \mu < \frac{2}{\lambda_{\max}}$	0	0	0
LMS	$e(n) = d(n) - \boldsymbol{u}^T(n)\boldsymbol{w}(n)$ $\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu\boldsymbol{u}(n)e(n)$	$\mathcal{O}(M)$	$0 < \mu < \frac{2}{\mathrm{tr}(\boldsymbol{R}_u)}$	$\frac{\mu}{2}J_{\min}\mathrm{tr}(\boldsymbol{R}_u)$	$\frac{\mu}{2}\mathrm{tr}(\boldsymbol{R}_u)$	$\frac{\mu}{2}J_{\min}M$
NLMS	$e(n) = d(n) - \boldsymbol{u}^T(n)\boldsymbol{w}(n)$ $\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \frac{\beta}{\epsilon + \|\boldsymbol{u}(n)\|^2}\boldsymbol{u}(n)e(n)$	$\mathcal{O}(M)$	$0 < \beta < 2$	$\frac{\beta}{2}J_{\min}$	$\frac{\beta}{2}$	$\frac{\beta J_{\min}}{2\mathrm{tr}(\boldsymbol{R}_u)}$
APA	$\boldsymbol{e}(n) = \boldsymbol{d}(n) - \boldsymbol{U}^T(n)\boldsymbol{w}(n)$ $\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \beta\boldsymbol{U}(n)[\epsilon\boldsymbol{I} + \boldsymbol{U}^T(n)\boldsymbol{U}(n)]^{-1}\boldsymbol{e}(n)$	$\mathcal{O}(MK^2)$	$0 < \beta < 2$	$\frac{\beta}{2}J_{\min}K$	$\frac{\beta}{2}K$	no simple expression
RLS	$\boldsymbol{\pi}(n) = \boldsymbol{P}(n-1)\boldsymbol{u}(n)$ $\boldsymbol{k}(n) = \frac{\boldsymbol{\pi}(n)}{\lambda + \boldsymbol{u}^T(n)\boldsymbol{\pi}(n)}$ $\xi(n) = d(n) - \boldsymbol{u}^T(n)\boldsymbol{w}(n-1)$ $\boldsymbol{w}(n) = \boldsymbol{w}(n-1) + \boldsymbol{k}(n)\xi(n)$ $\boldsymbol{P}(n) = \lambda^{-1}\left[\boldsymbol{P}(n-1) - \boldsymbol{k}(n)\boldsymbol{\pi}^T(n)\right]$	$\mathcal{O}(M^2)$	$0 < \lambda \leq 1$	$\frac{J_{\min}\frac{1-\lambda}{1+\lambda}M}{1-\frac{1-\lambda}{1+\lambda}M}$	$\frac{\frac{1-\lambda}{1+\lambda}M}{1-\frac{1-\lambda}{1+\lambda}M}$	$\frac{J_{\min}\frac{1-\lambda}{1+\lambda}M}{1-\frac{1-\lambda}{1+\lambda}M}\sum_{m=1}^{M}\frac{1}{\lambda_m}$

Table A.1: Overview over the basic adaptive filters and their properties. Most of the expressions for the mean-square stability, the excess mean-square error (EMSE), the misadjustment, and the mean-square deviation (MSD) are approximations.

Appendix B

Transient and Steady-State Analysis of the LMS Adaptive Filter

In this appendix, we give a more detailed transient and steady-state analysis of the LMS adaptive filter than found in Lecture 2. Specifically, we derive the condition for mean-square convergence and the expressions for the learning curve, the excess mean-square error (EMSE), the mean-square deviation (MSD), and the misadjustment. We do this for two reasons. Firstly, it is instructive to see how the analysis is carried out, and, secondly, it highlights how difficult the analysis is - even of the simple LMS adaptive filter.

In the derivation below, we use the analysis model previously discussed in Sec. 3.2.1. We also need the following important result.

A Special Fourth Order Moment of a Gaussian Random Vector

Let $\boldsymbol{u}$ be a real-valued $M \times 1$ dimensional Gaussian random vector with zero-mean and correlation matrix $\boldsymbol{R}_u$. Moreover, let $\boldsymbol{W}$ be an $M \times M$ symmetric matrix. We then have that [7, p. 44]

$$E[\boldsymbol{u}\boldsymbol{u}^T\boldsymbol{W}\boldsymbol{u}\boldsymbol{u}^T] = \boldsymbol{R}_u \mathrm{tr}(\boldsymbol{W}\boldsymbol{R}_u) + 2\boldsymbol{R}_u\boldsymbol{W}\boldsymbol{R}_u \ . \tag{B.1}$$

If $\boldsymbol{R}_u = \boldsymbol{X}\boldsymbol{\Lambda}\boldsymbol{X}^T$ is the eigenvalue decomposition of the correlation matrix and if we define $\boldsymbol{F} = \boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X}$, we may write

$$E[\boldsymbol{u}\boldsymbol{u}^T\boldsymbol{W}\boldsymbol{u}\boldsymbol{u}^T] = \boldsymbol{X}[\boldsymbol{\Lambda}\mathrm{tr}(\boldsymbol{F}\boldsymbol{\Lambda}) + 2\boldsymbol{\Lambda}\boldsymbol{F}\boldsymbol{\Lambda}]\boldsymbol{X}^T \ . \tag{B.2}$$

With this result in mind, we can now evaluate the following expectation

$$E[(\boldsymbol{I} - \mu\boldsymbol{u}\boldsymbol{u}^T)\boldsymbol{W}(\boldsymbol{I} - \mu\boldsymbol{u}\boldsymbol{u}^T)] = \boldsymbol{W} + \mu^2 E[\boldsymbol{u}\boldsymbol{u}^T\boldsymbol{W}\boldsymbol{u}\boldsymbol{u}^T] - \mu\boldsymbol{R}_u\boldsymbol{W} - \mu\boldsymbol{W}\boldsymbol{R}_u \tag{B.3}$$

$$= \boldsymbol{W} + \mu^2\boldsymbol{R}_u\mathrm{tr}(\boldsymbol{W}\boldsymbol{R}_u) + 2\mu^2\boldsymbol{R}_u\boldsymbol{W}\boldsymbol{R}_u - \mu\boldsymbol{R}_u\boldsymbol{W} - \mu\boldsymbol{W}\boldsymbol{R}_u \tag{B.4}$$

$$= \boldsymbol{X}[\boldsymbol{F} - \mu(\boldsymbol{\Lambda}\boldsymbol{F} + \boldsymbol{F}\boldsymbol{\Lambda}) + \mu^2(\boldsymbol{\Lambda}\mathrm{tr}(\boldsymbol{F}\boldsymbol{\Lambda}) + 2\boldsymbol{\Lambda}\boldsymbol{F}\boldsymbol{\Lambda})]\boldsymbol{X}^T \ . \tag{B.5}$$

If $\boldsymbol{F}$ is diagonal, then $\boldsymbol{F}\boldsymbol{\Lambda} = \boldsymbol{\Lambda}\boldsymbol{F}$ and we have that

$$E[(\boldsymbol{I} - \mu\boldsymbol{u}\boldsymbol{u}^T)\boldsymbol{W}(\boldsymbol{I} - \mu\boldsymbol{u}\boldsymbol{u}^T)] = \boldsymbol{X}[\boldsymbol{F} - 2\mu\boldsymbol{\Lambda}\boldsymbol{F} + \mu^2(\boldsymbol{\Lambda}\mathrm{tr}(\boldsymbol{F}\boldsymbol{\Lambda}) + 2\boldsymbol{\Lambda}^2\boldsymbol{F})]\boldsymbol{X}^T \ . \tag{B.6}$$

This result is very important for the derivation below.

The Analysis Model

The LMS algorithm is given by

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu\boldsymbol{u}(n)e(n) \tag{B.7}$$

where $\boldsymbol{w}(n)$, $\boldsymbol{u}(n)$, and $e(n)$ are the filter vector, the input vector and the error, respectively, at time index n. The positive scalar μ is the step-size, and the error is given by

$$e(n) = d(n) - \boldsymbol{u}^T(n)\boldsymbol{w}(n) \tag{B.8}$$

where the desired signal $d(n)$ in the analysis model (see also Sec. 3.2.1) is given by

$$d(n) = v(n) + \boldsymbol{u}^T(n)\boldsymbol{w}_o \ . \tag{B.9}$$

We assume that $v(n)$ is white Gaussian noise with variance σ_v^2 and uncorrelated with the input signal $u(n)$. Finally, we assume that $\boldsymbol{u}(n)$ is a white Gaussian process with correlation matrix $\boldsymbol{R}_u$. This implies that $\boldsymbol{u}(n)$ and $\boldsymbol{w}(n)$ are uncorrelated.

The difference between the optimal filter vector $\boldsymbol{w}_o$ and the filter vector $\boldsymbol{w}(n)$ is called the weight error, and it is given by

$$\begin{aligned}
\boldsymbol{\Delta w}(n) &= \boldsymbol{w}_o - \boldsymbol{w}(n) = \boldsymbol{w}_o - \boldsymbol{w}(n-1) - \mu\boldsymbol{u}(n-1)e(n-1) && \text{(B.10)}\\
&= \boldsymbol{\Delta w}(n-1) - \mu\boldsymbol{u}(n-1)[v(n) + \boldsymbol{u}^T(n-1)\boldsymbol{w}_o - \boldsymbol{u}^T(n-1)\boldsymbol{w}(n-1)] && \text{(B.11)}\\
&= [\boldsymbol{I} - \mu\boldsymbol{u}(n-1)\boldsymbol{u}^T(n-1)]\boldsymbol{\Delta w}(n-1) - \mu\boldsymbol{u}(n-1)v(n) \ . && \text{(B.12)}
\end{aligned}$$

In terms of the weight error, the error is given by

$$e(n) = d(n) - \boldsymbol{u}^T(n)\boldsymbol{w}(n) = v(n) + \boldsymbol{u}^T(n)\boldsymbol{\Delta w}(n) \ , \tag{B.13}$$

and the minimum mean-squared error (MMSE) is achieved at the optimum where $\boldsymbol{\Delta w}(n) = \boldsymbol{0}$ and thus given by

$$J_{\min} = J_1(\boldsymbol{w}_o) = E[v^2(n)] = \sigma_v^2 \ . \tag{B.14}$$

Transient Analysis

In the transient analysis, we first consider how we should select the step-size μ in order to ensure the mean-square stability of the LMS algorithm. Second, we derive an expression for the learning curve.

Mean-Square Convergence

As already discussed in Lecture 2, we say that the LMS algorithm converges in the mean-square if

$$\lim_{n\to\infty} E[\|\boldsymbol{\Delta w}(n)\|^2] = c < \infty \ . \tag{B.15}$$

In this section, we show that the LMS algorithm converges in the mean square if the step-size μ is selected such that it satisfies

$$f(\mu) = \frac{\mu}{2} \sum_{m=1}^{M} \frac{\lambda_m}{1 - \mu\lambda_m} < 1 \tag{B.16}$$

where λ_m is the m'th eigenvalue of the correlation matrix $\boldsymbol{R}_u$.

In order to do this, we investigate $E[\|\boldsymbol{\Delta w}(n)\|^2]$ as n grows. For convenience of notation, it turns out to be easier to work with $E[\|\boldsymbol{c}(n)\|^2]$ where $\boldsymbol{c}(n) = \boldsymbol{X}^T\boldsymbol{\Delta w}(n)$ and $\boldsymbol{X}$ contains the M eigenvectors of $\boldsymbol{R}_u$. Since $\boldsymbol{X}$ is orthogonal, it follows that

$$E[\|\boldsymbol{c}(n)\|^2] = E[\|\boldsymbol{X}^T\boldsymbol{\Delta w}(n)\|^2] = E[\|\boldsymbol{\Delta w}(n)\|^2] \tag{B.17}$$

Thus, $E[\|\boldsymbol{c}(n)\|^2]$ converges in the same way as $E[\|\boldsymbol{\Delta w}(n)\|^2]$ does. In the derivation below, we express $E[\|\boldsymbol{c}(n)\|^2]$ as a function of $\boldsymbol{c}(0)$, μ, $\boldsymbol{\Lambda}$, and $J_{\min}$. We do this in a recursive manner starting by expressing $E[\|\boldsymbol{c}(n)\|^2]$ as a function of $\boldsymbol{c}(n-1)$, μ, $\boldsymbol{\Lambda}$, and $J_{\min}$. Once we have obtained an expression for this recursion, we express $E[\|\boldsymbol{c}(n)\|^2]$ as a function of $\boldsymbol{c}(n-2)$, μ, $\boldsymbol{\Lambda}$, and $J_{\min}$ in the next recursion and so on.

The First Recursion

Multiplying Eq. (B.12) from the left with $\boldsymbol{X}^T$ and inserting it into $E[\|\boldsymbol{c}(n)\|^2]$ yields

$$\begin{aligned} E[\|\boldsymbol{c}(n)\|^2] &= E[\|\boldsymbol{X}^T[\boldsymbol{I} - \mu\boldsymbol{u}(n-1)\boldsymbol{u}^T(n-1)]\boldsymbol{X}\boldsymbol{c}(n-1) - \mu\boldsymbol{X}^T\boldsymbol{u}(n-1)v(n)\|^2] && \text{(B.18)} \\ &= E[\boldsymbol{c}^T(n-1)\boldsymbol{X}^T[\boldsymbol{I} - \mu\boldsymbol{u}(n-1)\boldsymbol{u}^T(n-1)]^2\boldsymbol{X}\boldsymbol{c}(n-1)] \\ &\quad + \mu^2 E[v^2(n)\boldsymbol{u}^T(n-1)\boldsymbol{u}(n-1)] && \text{(B.19)} \end{aligned}$$

where the last equality follows from the assumption that $v(n)$ and $\boldsymbol{u}(n)$ are uncorrelated. The expectation of the last term is given by

$$\begin{aligned} E[v^2(n)\boldsymbol{u}^T(n-1)\boldsymbol{u}(n-1)] &= E[v^2(n)]E[\boldsymbol{u}^T(n-1)\boldsymbol{u}(n-1)] && \text{(B.20)} \\ &= J_{\min}E[\mathrm{tr}(\boldsymbol{u}^T(n-1)\boldsymbol{u}(n-1))] && \text{(B.21)} \\ &= J_{\min}E[\mathrm{tr}(\boldsymbol{u}(n-1)\boldsymbol{u}^T(n-1))] && \text{(B.22)} \\ &= J_{\min}\mathrm{tr}(E[\boldsymbol{u}(n-1)\boldsymbol{u}^T(n-1)]) && \text{(B.23)} \\ &= J_{\min}\mathrm{tr}(\boldsymbol{R}_u) && \text{(B.24)} \\ &= J_{\min}\mathrm{tr}(\boldsymbol{\Lambda})\ . && \text{(B.25)} \end{aligned}$$

The second equality follows since the trace of a scalar equals the value of the scaler. The third equality follows from a property of the trace operator given by

$$\mathrm{tr}(\boldsymbol{AB}) = \mathrm{tr}(\boldsymbol{BA}) \tag{B.26}$$

for any pair of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ with $\boldsymbol{A}$ and $\boldsymbol{B}^T$ being of the same dimension. The fourth equality follows since both the expectation and the trace operators are linear operators whose order can therefore be interchanged. Finally, the last equality follows since

$$\mathrm{tr}(\boldsymbol{R}_u) = \mathrm{tr}(\boldsymbol{X}\boldsymbol{\Lambda}\boldsymbol{X}^T) = \mathrm{tr}(\boldsymbol{\Lambda}\boldsymbol{X}^T\boldsymbol{X}) = \mathrm{tr}(\boldsymbol{\Lambda})\ . \tag{B.27}$$

This was previously stated in the beginning of Lecture 2.

In order to evaluate the expectation of the first term in Eq. (B.19), we use the law of iterated expectations given by[1]

$$E[X] = E[E[X|Y]] \ , \tag{B.28}$$

and the assumption that $\boldsymbol{u}(n)$ and $\boldsymbol{w}(n)$ are uncorrelated. From this, we obtain that

$$\begin{aligned} E[\boldsymbol{c}^T(n-1)\boldsymbol{X}^T[\boldsymbol{I}-\mu\boldsymbol{u}(n-1)\boldsymbol{u}^T(n-1)]^2\boldsymbol{X}\boldsymbol{c}(n-1)] = \\ E[\boldsymbol{c}^T(n-1)\boldsymbol{X}^T E[(\boldsymbol{I}-\mu\boldsymbol{u}(n-1)\boldsymbol{u}^T(n-1))^2]\boldsymbol{X}\boldsymbol{c}(n-1)] \ . \end{aligned} \tag{B.29}$$

The inner expectation is on the same form as Eq. (B.6) with $\boldsymbol{W} = \boldsymbol{F} = \boldsymbol{I}$. Thus, we have that

$$\boldsymbol{S}(1) = \boldsymbol{X}^T E[(\boldsymbol{I}-\mu\boldsymbol{u}(n-1)\boldsymbol{u}^T(n-1))^2]\boldsymbol{X} \tag{B.30}$$

$$= \boldsymbol{I} - 2\mu\boldsymbol{\Lambda} + \mu^2[\boldsymbol{\Lambda}\mathrm{tr}(\boldsymbol{\Lambda}) + 2\boldsymbol{\Lambda}^2] \ . \tag{B.31}$$

which is a diagonal matrix. Thus, we can write Eq. (B.19) as

$$E[\|\boldsymbol{c}(n)\|^2] = E[\boldsymbol{c}^T(n-1)\boldsymbol{S}(1)\boldsymbol{c}(n-1)] + \mu^2 J_{\min}\mathrm{tr}(\boldsymbol{\Lambda}) \ . \tag{B.32}$$

The Second Recursion

Since the second term of Eq. (B.32) does not depend on $\boldsymbol{c}(n)$, we consider the first term of Eq. (B.32). Multiplying Eq. (B.12) for time index $n-1$ from the left with $\boldsymbol{X}^T$ and inserting it into the first term yields

$$\begin{aligned} &E[\boldsymbol{c}^T(n-1)\boldsymbol{S}(1)\boldsymbol{c}(n-1)] = \\ &\quad E[\boldsymbol{c}^T(n-2)\boldsymbol{X}^T[\boldsymbol{I}-\mu\boldsymbol{u}(n-2)\boldsymbol{u}^T(n-2)]\boldsymbol{X}\boldsymbol{S}(1)\boldsymbol{X}^T[\boldsymbol{I}-\mu\boldsymbol{u}(n-2)\boldsymbol{u}^T(n-2)]\boldsymbol{X}\boldsymbol{c}(n-2)] \\ &\quad + \mu^2 E[v^2(n)\boldsymbol{u}^T(n-2)\boldsymbol{X}\boldsymbol{S}(1)\boldsymbol{X}^T\boldsymbol{u}(n-2)] \ . \end{aligned} \tag{B.33}$$

The expectation of the last term is given by

$$\begin{aligned} E[v^2(n)\boldsymbol{u}^T(n-2)\boldsymbol{X}\boldsymbol{S}(1)\boldsymbol{X}^T\boldsymbol{u}(n-2)] &= E[v^2(n)]E[\boldsymbol{u}^T(n-2)\boldsymbol{X}\boldsymbol{S}(1)\boldsymbol{X}^T\boldsymbol{u}(n-2)] \\ &= J_{\min}E[\mathrm{tr}(\boldsymbol{u}^T(n-2)\boldsymbol{X}\boldsymbol{S}(1)\boldsymbol{X}^T\boldsymbol{u}(n-2))] \\ &= J_{\min}\mathrm{tr}(\boldsymbol{X}\boldsymbol{S}(1)\boldsymbol{X}^T E[\boldsymbol{u}(n-2)\boldsymbol{u}^T(n-2)]) \\ &= J_{\min}\mathrm{tr}(\boldsymbol{S}(1)\boldsymbol{\Lambda}) \ . \end{aligned} \tag{B.34}$$

The equalities in the derivation above follows from the same arguments as in the derivation for the equivalent expression in the first recursion. Using the same arguments as in the first recursion, the first term of Eq. (B.33) may be written as an outer and an inner expectation. The inner expectation can be evaluated using Eq. (B.6) with $\boldsymbol{W} = \boldsymbol{X}\boldsymbol{S}(1)\boldsymbol{X}^T$ or $\boldsymbol{F} = \boldsymbol{S}(1)$. Thus, we have that $\boldsymbol{S}(2)$ is given by

$$\begin{aligned} \boldsymbol{S}(2) &= \boldsymbol{X}^T E\left[[\boldsymbol{I}-\mu\boldsymbol{u}(n-2)\boldsymbol{u}^T(n-2)]\boldsymbol{X}\boldsymbol{S}(1)\boldsymbol{X}^T[\boldsymbol{I}-\mu\boldsymbol{u}(n-2)\boldsymbol{u}^T(n-2)]\right]\boldsymbol{X} \\ &= \boldsymbol{S}(1) - 2\mu\boldsymbol{\Lambda}\boldsymbol{S}(1) + \mu^2[\boldsymbol{\Lambda}\mathrm{tr}(\boldsymbol{S}(1)\boldsymbol{\Lambda}) + 2\boldsymbol{\Lambda}\boldsymbol{S}(1)\boldsymbol{\Lambda}] \end{aligned} \tag{B.35}$$

[1] It follows since

$$\begin{aligned} E[E[X|Y]] &= \int E[X|Y]f_Y(y)dy = \int\left[\int x f_{X|Y}(x|y)dx\right]f_Y(y)dy = \int\int x f_{X,Y}(x,y)dxdy \\ &= \int x\left[\int f_{X,Y}(x,y)dy\right]dx = \int x f_X(x)dx = E[X] \ . \end{aligned}$$

which is diagonal since $\boldsymbol{S}(1)$ is diagonal. Thus, in total we have that

$$E[\|\boldsymbol{c}(n)\|^2] = E[\boldsymbol{c}^T(n-1)\boldsymbol{S}(1)\boldsymbol{c}(n-1)] + \mu^2 J_{\min}\mathrm{tr}(\boldsymbol{\Lambda}) \tag{B.36}$$

$$= E[\boldsymbol{c}^T(n-2)\boldsymbol{S}(2)\boldsymbol{c}(n-2)] + \mu^2 J_{\min}\mathrm{tr}(\boldsymbol{S}(1)\boldsymbol{\Lambda}) + \mu^2 J_{\min}\mathrm{tr}(\boldsymbol{\Lambda}) \tag{B.37}$$

$$= E[\boldsymbol{c}^T(n-2)\boldsymbol{S}(2)\boldsymbol{c}(n-2)] + \mu^2 J_{\min}\mathrm{tr}([\boldsymbol{S}(1)+\boldsymbol{I}]\boldsymbol{\Lambda}) \ . \tag{B.38}$$

The Third Recursion

Since the second term of Eq. (B.38) does not depend on $\boldsymbol{c}(n)$, we consider the first term of Eq. (B.38). This has the same form as the first term of Eq. (B.32) so the derivation of the third recursion is the same as for the second recursion, except that all the time indices should be decreased by one. We therefore obtain that

$$E[\|\boldsymbol{c}(n)\|^2] = E[\boldsymbol{c}^T(n-1)\boldsymbol{S}(1)\boldsymbol{c}(n-1)] + \mu^2 J_{\min}\mathrm{tr}(\boldsymbol{\Lambda}) \tag{B.39}$$

$$= E[\boldsymbol{c}^T(n-2)\boldsymbol{S}(2)\boldsymbol{c}(n-2)] + \mu^2 J_{\min}\mathrm{tr}([\boldsymbol{S}(1)+\boldsymbol{I}]\boldsymbol{\Lambda}) \tag{B.40}$$

$$= E[\boldsymbol{c}^T(n-3)\boldsymbol{S}(3)\boldsymbol{c}(n-3)] + \mu^2 J_{\min}\mathrm{tr}([\boldsymbol{S}(2)+\boldsymbol{S}(1)+\boldsymbol{I}]\boldsymbol{\Lambda}) \ . \tag{B.41}$$

The n'th Recursion

From the first, second, and third recursion, it is not hard to see a pattern for the recursions. Therefore, for the n'th recursion, we have that

$$E[\|\boldsymbol{c}(n)\|^2] = \boldsymbol{c}^T(0)\boldsymbol{S}(n)\boldsymbol{c}(0) + \mu^2 J_{\min}\sum_{i=0}^{n-1}\mathrm{tr}(\boldsymbol{S}(i)\boldsymbol{\Lambda}) \tag{B.42}$$

where

$$\boldsymbol{S}(n) = \boldsymbol{S}(n-1) - 2\mu\boldsymbol{\Lambda}\boldsymbol{S}(n-1) + \mu^2[\boldsymbol{\Lambda}\mathrm{tr}(\boldsymbol{S}(n-1)\boldsymbol{\Lambda}) + 2\boldsymbol{\Lambda}\boldsymbol{S}(n-1)\boldsymbol{\Lambda}] \tag{B.43}$$

with $\boldsymbol{S}(0) = \boldsymbol{I}$. From the recursion in Eq. (B.42), we see that we must require that $\boldsymbol{c}^T(0)\boldsymbol{S}(n)\boldsymbol{c}(0)$ remains bounded for $n \to \infty$, regardless of the initial conditions $\boldsymbol{c}(0)$. Since $\boldsymbol{c}^T(0)\boldsymbol{S}(n)\boldsymbol{c}(0)$ remains bounded if and only if all the eigenvalues of $\boldsymbol{S}(n)$ are in the interval $[-1, 1]$, we therefore perform an eigenvalue analysis of $\boldsymbol{S}(n)$.

Eigenvalue Analysis of $\boldsymbol{S}(n)$

Define the vector of ones

$$\mathbf{1} = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^T \ . \tag{B.44}$$

Using this vector, we may write the trace of a diagonal matrix $\boldsymbol{M}$ as

$$\mathrm{tr}(\boldsymbol{M}) = \mathbf{1}^T\boldsymbol{M}\mathbf{1} \ . \tag{B.45}$$

Since $\boldsymbol{S}(n)$ and $\boldsymbol{\Lambda}$ are diagonal matrices, the product $\boldsymbol{S}(n)\boldsymbol{\Lambda}$ is also diagonal, and we have that

$$\boldsymbol{S}(n-1)\boldsymbol{\Lambda} = \boldsymbol{\Lambda}\boldsymbol{S}(n-1) \tag{B.46}$$

$$\mathrm{tr}(\boldsymbol{S}(n)\boldsymbol{\Lambda}) = \mathbf{1}^T\boldsymbol{S}(n)\boldsymbol{\Lambda}\mathbf{1} = \mathbf{1}^T\boldsymbol{\Lambda}\boldsymbol{S}(n)\mathbf{1} \ . \tag{B.47}$$

Multiplying both sides of Eq. (B.43) from the right with $\mathbf{1}$, we obtain

$$\boldsymbol{S}(n)\mathbf{1} = \boldsymbol{S}(n-1)\mathbf{1} - 2\mu\boldsymbol{\Lambda}\boldsymbol{S}(n-1)\mathbf{1} + \mu^2[\boldsymbol{\Lambda}\mathbf{1}\mathbf{1}^T\boldsymbol{\Lambda}\boldsymbol{S}(n-1)\mathbf{1} + 2\boldsymbol{\Lambda}^2\boldsymbol{S}(n-1)\mathbf{1}] \tag{B.48}$$

$$= [\boldsymbol{I} - 2\mu\boldsymbol{\Lambda} + \mu^2(\boldsymbol{\Lambda}\mathbf{1}\mathbf{1}^T\boldsymbol{\Lambda} + 2\boldsymbol{\Lambda}^2)]\boldsymbol{S}(n-1)\mathbf{1} \ . \tag{B.49}$$

Now, we define

$$\boldsymbol{D} = \boldsymbol{I} - 2\mu\boldsymbol{\Lambda} + \mu^2(\boldsymbol{\Lambda}\mathbf{1}\mathbf{1}^T\boldsymbol{\Lambda} + 2\boldsymbol{\Lambda}^2) \tag{B.50}$$

so that we have

$$\begin{aligned} \boldsymbol{S}(n)\mathbf{1} &= \boldsymbol{D}\boldsymbol{S}(n-1)\mathbf{1} && \text{(B.51)}\\ &= \boldsymbol{D}^2\boldsymbol{S}(n-2)\mathbf{1} && \text{(B.52)}\\ &\vdots \\ &= \boldsymbol{D}^n\boldsymbol{S}(0)\mathbf{1} && \text{(B.53)} \end{aligned}$$

This means that the elements of the diagonal matrix $\boldsymbol{S}(n)$ are bounded if and only if all the eigenvalues of $\boldsymbol{D}$ are in the interval $[-1, 1]$.

Eigenvalue Analysis of $\boldsymbol{D}$

We can write $\boldsymbol{D}$ as

$$\boldsymbol{D} = 2(\boldsymbol{I} - \mu\boldsymbol{\Lambda})^2 + 2\mu\boldsymbol{\Lambda} + \mu^2\boldsymbol{\Lambda}\mathbf{1}\mathbf{1}^T\boldsymbol{\Lambda} \tag{B.54}$$

which is clearly positive semidefinite since $\mu > 0$ and $\lambda_m > 0$. Thus, all eigenvalues of $\boldsymbol{D}$ are non-negative, and the only requirement to the eigenvalues is therefore that they are smaller than one. This is equivalent to requiring that $\boldsymbol{I} - \boldsymbol{D}$ is positive definite. From Sylvester's criterion, we know that an $M \times M$ matrix $\boldsymbol{A}$ is positive definite if and only if the determinant of all the upper $m \times m$ matrices $\boldsymbol{A}_m$ of $\boldsymbol{A}$ are positive for $m = 1, \ldots, M$. If $\mathbf{1}_m$ is a vector consisting of m ones, $\boldsymbol{I}_m$ is the $m \times m$ identity matrix, and $\boldsymbol{\Lambda}_m$ and $\boldsymbol{D}_m$ are the upper $m \times m$ matrices of $\boldsymbol{\Lambda}$ and $\boldsymbol{D}$, respectively, we have that

$$\boldsymbol{D}_m = \boldsymbol{I}_m - 2\mu\boldsymbol{\Lambda}_m + \mu^2\boldsymbol{\Lambda}_m(\mathbf{1}_m\mathbf{1}_m^T + 2\boldsymbol{I}_m)\boldsymbol{\Lambda}_m \ , \qquad \text{for } m = 1, \ldots, M \ , \tag{B.55}$$

and we require that

$$|\boldsymbol{I}_m - \boldsymbol{D}_m| > 0 \ , \qquad \text{for } m = 1, \ldots, M \tag{B.56}$$

for $\boldsymbol{I} - \boldsymbol{D}$ to be positive definite. The determinant of $\boldsymbol{I}_m - \boldsymbol{D}_m$ is

$$\begin{aligned} |\boldsymbol{I}_m - \boldsymbol{D}_m| &= |2\mu\boldsymbol{\Lambda}_m - \mu^2\boldsymbol{\Lambda}_m(\mathbf{1}_m\mathbf{1}_m^T + 2\boldsymbol{I}_m)\boldsymbol{\Lambda}_m| && \text{(B.57)}\\ &= |\mu^2\boldsymbol{\Lambda}_m||2\mu^{-1}\boldsymbol{I}_m - 2\boldsymbol{\Lambda}_m - \mathbf{1}_m\mathbf{1}_m^T\boldsymbol{\Lambda}_m| && \text{(B.58)} \end{aligned}$$

Since $\mu > 0$ and $\lambda_m > 0$, the determinant of $\mu^2\boldsymbol{\Lambda}_m$ is positive for all $m = 1, \ldots, M$. In order to evaluate the second determinant, we use the matrix determinant lemma which states that

$$|\boldsymbol{A} + \boldsymbol{U}\boldsymbol{V}^T| = |\boldsymbol{A}||\boldsymbol{I} + \boldsymbol{V}^T\boldsymbol{A}^{-1}\boldsymbol{U}| \tag{B.59}$$

where $\boldsymbol{A}$ is an $N \times N$ matrix, and $\boldsymbol{U}$ and $\boldsymbol{V}$ are $N \times M$ matrices. From the matrix determinant lemma, we have that

$$|2\mu^{-1}\boldsymbol{I}_m - 2\boldsymbol{\Lambda}_m - \mathbf{1}_m\mathbf{1}_m^T\boldsymbol{\Lambda}_m| = |2\mu^{-1}\boldsymbol{I}_m - 2\boldsymbol{\Lambda}_m||1 - \frac{1}{2}\mathbf{1}_m^T\boldsymbol{\Lambda}_m(\mu^{-1}\boldsymbol{I}_m - \boldsymbol{\Lambda}_m)^{-1}\mathbf{1}_m| \tag{B.60}$$

The argument of the first determinant is a diagonal matrix. Thus, it leads to

$$\begin{aligned} & 0 < 2\mu^{-1} - 2\lambda_m \ , \qquad \text{for } m = 1, \ldots, M && \text{(B.61)}\\ \Updownarrow & \\ & \mu < \frac{1}{\lambda_{\max}} \ . && \text{(B.62)} \end{aligned}$$

The argument of the second determinant is a scalar. It leads to

$$0 < 1 - \frac{1}{2}\mathbf{1}_m^T \mathbf{\Lambda}_m (\mu^{-1}\boldsymbol{I}_m - \mathbf{\Lambda}_m)^{-1}\mathbf{1}_m \ , \qquad \text{for } m = 1, \ldots, M \tag{B.63}$$

$$\Updownarrow$$

$$1 > \frac{1}{2}\mathbf{1}_m^T \mathbf{\Lambda}_m (\mu^{-1}\boldsymbol{I}_m - \mathbf{\Lambda}_m)^{-1}\mathbf{1}_m \ , \qquad \text{for } m = 1, \ldots, M \tag{B.64}$$

$$= \frac{1}{2}\mathrm{tr}(\mathbf{\Lambda}_m (\mu^{-1}\boldsymbol{I}_m - \mathbf{\Lambda}_m)^{-1}) \ , \qquad \text{for } m = 1, \ldots, M \tag{B.65}$$

$$= \frac{1}{2}\sum_{i=1}^{m} \frac{\lambda_i}{\mu^{-1} - \lambda_i} \ , \qquad \text{for } m = 1, \ldots, M \tag{B.66}$$

$$= \frac{\mu}{2}\sum_{i=1}^{m} \frac{\lambda_i}{1 - \mu\lambda_i} = f_m(\mu) \ , \qquad \text{for } m = 1, \ldots, M \ . \tag{B.67}$$

The first bound on the step-size in Eq. (B.62) ensures that $\lambda_i/(1-\mu\lambda_i)$ is always positive. Thus,

$$f_1(\mu) < f_2(\mu) < \cdots < f_M(\mu) \ , \qquad \text{for } \mu \in [0, \lambda_{\max}^{-1}] \ . \tag{B.68}$$

Therefore, if the step-size satisfies

$$f(\mu) = f_M(\mu) < 1 \ , \tag{B.69}$$

then all of the functions $f_m(\mu)$ are also smaller than one. Moreover, $f(0) = 0$, and $f(\mu)$ is an increasing function as long as the first bound on the step-size in Eq. (B.62) is satisfied. The latter follows since the derivative of $f(\mu)$ satisfies

$$\frac{df}{d\mu} = \frac{1}{2}\sum_{m=1}^{M} \frac{\lambda_m}{(1-\mu\lambda_m)^2} > 0 \ , \qquad \text{for } \mu \in [0, \lambda_{\max}^{-1}] \ . \tag{B.70}$$

These observations lead to the conclusion that $\boldsymbol{I} - \boldsymbol{D}$ is positive definite, provided that the step-size satisfies the bound

$$\boxed{f(\mu) = \frac{\mu}{2}\sum_{m=1}^{M} \frac{\lambda_m}{1-\mu\lambda_m} < 1} \ . \tag{B.71}$$

In matrix notation, we can write this bound as

$$\boxed{f(\mu) = \frac{1}{2}\mathbf{1}^T \mathbf{\Lambda}(\mu^{-1}\boldsymbol{I} - \mathbf{\Lambda})^{-1}\mathbf{1} = \frac{1}{2}\mathrm{tr}(\mathbf{\Lambda}(\mu^{-1}\boldsymbol{I} - \mathbf{\Lambda})^{-1}) < 1} \tag{B.72}$$

where the last equality follows since $\mathbf{\Lambda}(\mu^{-1}\boldsymbol{I} - \mathbf{\Lambda})^{-1}$ is diagonal. Moreover, since

$$\lim_{\mu \to \lambda_{\max}^{-1}} f(\mu) = \infty > 1 \ , \tag{B.73}$$

the first bound in Eq. (B.62) is always satisfied if the second bound in Eq. (B.71) is satisfied. Thus, we have shown that the LMS algorithm converges in the mean-square if and only if the step-size satisfies Eq. (B.71) or, equivalently, Eq. (B.72).

Learning Curve

The value of the cost function at time n is given by

$$J_1(\boldsymbol{w}(n)) = E[e^2(n)] = E[(v(n) + \boldsymbol{u}^T(n)\boldsymbol{\Delta w}(n))^2] \tag{B.74}$$

$$= E[\boldsymbol{\Delta w}^T(n)\boldsymbol{u}(n)\boldsymbol{u}^T(n)\boldsymbol{\Delta w}(n)] + J_{\min} \tag{B.75}$$

where the last equality follows from the fact that $v(n)$ and $\boldsymbol{u}(n)$ are uncorrelated. If we also use the law of iterated expectations and that $\boldsymbol{u}(n)$ and $\boldsymbol{w}(n)$ are uncorrelated, we obtain that

$$J_1(\boldsymbol{w}(n)) = E[\boldsymbol{\Delta w}^T(n)E[\boldsymbol{u}(n)\boldsymbol{u}^T(n)|\boldsymbol{w}(n)]\boldsymbol{\Delta w}(n)] + J_{\min} \tag{B.76}$$

$$= E[\boldsymbol{\Delta w}^T(n)E[\boldsymbol{u}(n)\boldsymbol{u}^T(n)]\boldsymbol{\Delta w}(n)] + J_{\min} \tag{B.77}$$

$$= E[\boldsymbol{\Delta w}^T(n)\boldsymbol{R}_u\boldsymbol{\Delta w}(n)] + J_{\min}\ . \tag{B.78}$$

Finally, if we replace the correlation matrix $\boldsymbol{R}_u$ of $\boldsymbol{u}(n)$ with its eigenvalue decomposition $\boldsymbol{R}_u = \boldsymbol{X}\boldsymbol{\Lambda}\boldsymbol{X}^T$, we have that

$$J_1(\boldsymbol{w}(n)) = E[\boldsymbol{c}^T(n)\boldsymbol{\Lambda}\boldsymbol{c}(n)] + J_{\min} \tag{B.79}$$

where we again have defined that $\boldsymbol{c}(n) = \boldsymbol{X}^T\boldsymbol{\Delta w}(n)$. Now, for $\boldsymbol{S}(0) = \boldsymbol{\Lambda}$, we obtain from Eq. (B.42) that

$$\boxed{J_1(\boldsymbol{w}(n)) = \boldsymbol{c}^T(0)\boldsymbol{S}(n)\boldsymbol{c}(0) + \mu^2 J_{\min}\sum_{i=0}^{n-1}\mathrm{tr}(\boldsymbol{S}(i)\boldsymbol{\Lambda}) + J_{\min}} \tag{B.80}$$

where

$$\boldsymbol{S}(n) = \boldsymbol{S}(n-1) - 2\mu\boldsymbol{\Lambda}\boldsymbol{S}(n-1) + \mu^2[\boldsymbol{\Lambda}\mathrm{tr}(\boldsymbol{S}(n-1)\boldsymbol{\Lambda}) + 2\boldsymbol{\Lambda}\boldsymbol{S}(n-1)\boldsymbol{\Lambda}] \tag{B.81}$$

$$= \mathrm{diag}(\boldsymbol{S}(n)\mathbf{1}) = \mathrm{diag}(\boldsymbol{D}^n\boldsymbol{\Lambda}\mathbf{1})\ . \tag{B.82}$$

Here, $\mathrm{diag}(\cdot)$ creates a diagonal matrix from a vector.

Steady-State Analysis

In the steady-state analysis, we derive expressions for the mean-square deviation (MSD), the excess mean-square error (EMSE), and the misadjustment of the LMS algorithm. These expressions can be derived in two different ways. One way is presented in [7, pp. 462–465]. Here, we give another derivation which we believe is more intuitive.

Mean-Square Deviation

The MSD is given by the limit

$$\text{MSD:} \qquad \lim_{n\to\infty} E[\|\boldsymbol{\Delta w}(n)\|^2]\ , \tag{B.83}$$

and we denote this limit by $E[\|\boldsymbol{\Delta w}(\infty)\|^2]$. We make direct use of the definition in the derivation of the MSD. Again, we define $\boldsymbol{c}(n) = \boldsymbol{X}^T\boldsymbol{\Delta w}(n)$. Thus, from Eq. (B.42), we have that

$$E[\|\boldsymbol{\Delta w}(\infty)\|^2] = \lim_{n\to\infty} E[\|\boldsymbol{\Delta w}(n)\|^2] = \lim_{n\to\infty} E[\|\boldsymbol{c}(n)\|^2] \tag{B.84}$$

$$= \lim_{n\to\infty} \left\{ \boldsymbol{c}^T(0)\boldsymbol{S}(n)\boldsymbol{c}(0) + \mu^2 J_{\min} \sum_{i=0}^{n-1} \operatorname{tr}(\boldsymbol{S}(i)\boldsymbol{\Lambda}) \right\} \tag{B.85}$$

$$= \lim_{n\to\infty} \boldsymbol{c}^T(0)\boldsymbol{S}(n)\boldsymbol{c}(0) + \lim_{n\to\infty} \mu^2 J_{\min} \sum_{i=0}^{n-1} \operatorname{tr}(\boldsymbol{S}(i)\boldsymbol{\Lambda}) \ . \tag{B.86}$$

with $\boldsymbol{S}(0) = \boldsymbol{I}$. If we select the step-size μ such that the LMS algorithm converges in the mean-square, the first term equals zero. Thus, we have that

$$E[\|\boldsymbol{\Delta w}(\infty)\|^2] = \lim_{n\to\infty} \mu^2 J_{\min} \sum_{i=0}^{n-1} \operatorname{tr}(\boldsymbol{S}(i)\boldsymbol{\Lambda}) = \mu^2 J_{\min} \sum_{i=0}^{\infty} \operatorname{tr}(\boldsymbol{S}(i)\boldsymbol{\Lambda}) \ . \tag{B.87}$$

The matrices $\boldsymbol{S}(i)$ and $\boldsymbol{\Lambda}$ are both diagonal, and we therefore use Eq. (B.47) to obtain

$$E[\|\boldsymbol{\Delta w}(\infty)\|^2] = \mu^2 J_{\min} \sum_{i=0}^{\infty} \mathbf{1}^T\boldsymbol{\Lambda}\boldsymbol{S}(i)\mathbf{1} \tag{B.88}$$

$$= \mu^2 J_{\min} \sum_{i=0}^{\infty} \mathbf{1}^T\boldsymbol{\Lambda}\boldsymbol{D}^i\mathbf{1} \tag{B.89}$$

$$= \mu^2 J_{\min} \mathbf{1}^T\boldsymbol{\Lambda} \sum_{i=0}^{\infty} \left[\boldsymbol{D}^i\right] \mathbf{1} \tag{B.90}$$

where the second equality follows from Eq. (B.53) with $\boldsymbol{S}(0) = \boldsymbol{I}$ and $\boldsymbol{D}$ defined in Eq. (B.50). Since all the eigenvalues of $\boldsymbol{D}$ have a magnitude smaller than 1, we have from the geometric series of matrices that [12, p. 58]

$$\sum_{i=0}^{\infty} \boldsymbol{D}^i = (\boldsymbol{I} - \boldsymbol{D})^{-1} \ . \tag{B.91}$$

Thus, we obtain that

$$E[\|\boldsymbol{\Delta w}(\infty)\|^2] = \mu^2 J_{\min} \mathbf{1}^T\boldsymbol{\Lambda}(\boldsymbol{I} - \boldsymbol{D})^{-1}\mathbf{1} \tag{B.92}$$

$$= \mu^2 J_{\min} \mathbf{1}^T\boldsymbol{\Lambda}(2\mu\boldsymbol{\Lambda} - \mu^2(\boldsymbol{\Lambda}\mathbf{1}\mathbf{1}^T\boldsymbol{\Lambda} + 2\boldsymbol{\Lambda}^2))^{-1}\mathbf{1} \tag{B.93}$$

$$= \mu^2 J_{\min} \mathbf{1}^T\boldsymbol{\Lambda}[(2\boldsymbol{\mu}^{-1}\boldsymbol{I} - 2\boldsymbol{\Lambda} - \boldsymbol{\Lambda}\mathbf{1}\mathbf{1}^T)\mu^2\boldsymbol{\Lambda}]^{-1}\mathbf{1} \tag{B.94}$$

$$= J_{\min} \mathbf{1}^T(2\boldsymbol{\mu}^{-1}\boldsymbol{I} - 2\boldsymbol{\Lambda} - \boldsymbol{\Lambda}\mathbf{1}\mathbf{1}^T)^{-1}\mathbf{1} \ . \tag{B.95}$$

Now, by defining

$$\boldsymbol{G} = \boldsymbol{\mu}^{-1}\boldsymbol{I} - \boldsymbol{\Lambda} \tag{B.96}$$

and using the matrix inversion lemma from Eq. (4.5), we obtain

$$E[\|\boldsymbol{\Delta w}(\infty)\|^2] = J_{\min}\mathbf{1}^T(2\boldsymbol{G} - \boldsymbol{\Lambda}\mathbf{1}\mathbf{1}^T)^{-1}\mathbf{1} \tag{B.97}$$

$$= J_{\min}\mathbf{1}^T\left[\frac{1}{2}\boldsymbol{G}^{-1} + \frac{1}{2}\boldsymbol{G}^{-1}\boldsymbol{\Lambda}\mathbf{1}\left(1 - \frac{1}{2}\mathbf{1}^T\boldsymbol{G}^{-1}\boldsymbol{\Lambda}\mathbf{1}\right)^{-1}\mathbf{1}^T\boldsymbol{G}^{-1}\frac{1}{2}\right]\mathbf{1} \tag{B.98}$$

$$= J_{\min}\frac{1}{2}\mathbf{1}^T\boldsymbol{G}^{-1}\mathbf{1}\left[1 + \left(1 - \frac{1}{2}\mathbf{1}^T\boldsymbol{G}^{-1}\boldsymbol{\Lambda}\mathbf{1}\right)^{-1}\frac{1}{2}\mathbf{1}^T\boldsymbol{G}^{-1}\boldsymbol{\Lambda}\mathbf{1}\right] \tag{B.99}$$

$$= J_{\min}\frac{1}{2}\mathbf{1}^T\boldsymbol{G}^{-1}\mathbf{1}\left(1 - \frac{1}{2}\mathbf{1}^T\boldsymbol{G}^{-1}\boldsymbol{\Lambda}\mathbf{1}\right)^{-1} . \tag{B.100}$$

Finally, from Eq. (B.72), we have that

$$f(\mu) = \frac{1}{2}\mathbf{1}^T\boldsymbol{\Lambda}\boldsymbol{G}^{-1}\mathbf{1} = \frac{1}{2}\mathbf{1}^T\boldsymbol{G}^{-1}\boldsymbol{\Lambda}\mathbf{1} \tag{B.101}$$

which leads to

$$\boxed{E[\|\boldsymbol{\Delta w}(\infty)\|^2] = \frac{J_{\min}}{1 - f(\mu)}\frac{\mu}{2}\sum_{m=1}^{M}\frac{1}{1 - \mu\lambda_m}} . \tag{B.102}$$

Excess Mean-Square Error

The derivation of the EMSE is done in the same way as the MSE was derived. The EMSE is given by the limit

$$\text{EMSE:} \qquad \lim_{n\to\infty} J_1(\boldsymbol{w}(n)) - J_{\min} , \tag{B.103}$$

and we denote it by J_{ex}. Inserting the expression for the learning curve from Eq. (B.80) in this limit yields

$$J_{\text{ex}} = \lim_{n\to\infty}\left\{\boldsymbol{c}^T(0)\boldsymbol{S}(n)\boldsymbol{c}(0) + \mu^2 J_{\min}\sum_{i=0}^{n-1}\text{tr}(\boldsymbol{S}(i)\boldsymbol{\Lambda})\right\} \tag{B.104}$$

$$= \lim_{n\to\infty}\boldsymbol{c}^T(0)\boldsymbol{S}(n)\boldsymbol{c}(0) + \lim_{n\to\infty}\mu^2 J_{\min}\sum_{i=0}^{n-1}\text{tr}(\boldsymbol{S}(i)\boldsymbol{\Lambda}) \tag{B.105}$$

where $\boldsymbol{S}(0) = \boldsymbol{\Lambda}$ and $\boldsymbol{c}(n) = \boldsymbol{X}^T\boldsymbol{\Delta w}(n)$. If we select the step-size μ such that the LMS algorithm converges in the mean-square, the first term equals zero. Thus, we have that

$$J_{\text{ex}} = \lim_{n\to\infty}\mu^2 J_{\min}\sum_{i=0}^{n-1}\text{tr}(\boldsymbol{S}(i)\boldsymbol{\Lambda}) = \mu^2 J_{\min}\sum_{i=0}^{\infty}\text{tr}(\boldsymbol{S}(i)\boldsymbol{\Lambda}) . \tag{B.106}$$

Note that the expression for the EMSE is the same as for the MSD, except for the value of $\boldsymbol{S}(0)$. The matrices $\boldsymbol{S}(i)$ and $\boldsymbol{\Lambda}$ are both diagonal, and we therefore use Eq. (B.47) to obtain

$$J_{\text{ex}} = \mu^2 J_{\min}\sum_{i=0}^{\infty}\mathbf{1}^T\boldsymbol{\Lambda}\boldsymbol{S}(i)\mathbf{1} \tag{B.107}$$

$$= \mu^2 J_{\min}\sum_{i=0}^{\infty}\mathbf{1}^T\boldsymbol{\Lambda}\boldsymbol{D}^i\boldsymbol{\Lambda}\mathbf{1} \tag{B.108}$$

$$= \mu^2 J_{\min}\mathbf{1}^T\boldsymbol{\Lambda}\sum_{i=0}^{\infty}\left[\boldsymbol{D}^i\right]\boldsymbol{\Lambda}\mathbf{1} \tag{B.109}$$

where the second equality follows from Eq. (B.53) with $\boldsymbol{S}(0) = \boldsymbol{\Lambda}$ and $\boldsymbol{D}$ defined in Eq. (B.50). Since all the eigenvalues of $\boldsymbol{D}$ have a magnitude smaller than 1, we have from the geometric series of matrices that [12, p. 58]

$$\sum_{i=0}^{\infty} \boldsymbol{D}^i = (\boldsymbol{I} - \boldsymbol{D})^{-1} \ . \tag{B.110}$$

Thus, we obtain that

$$\begin{aligned} J_{\text{ex}} &= \mu^2 J_{\min} \mathbf{1}^T \boldsymbol{\Lambda} (\boldsymbol{I} - \boldsymbol{D})^{-1} \boldsymbol{\Lambda} \mathbf{1} && \text{(B.111)} \\ &= \mu^2 J_{\min} \mathbf{1}^T \boldsymbol{\Lambda} (2\mu\boldsymbol{\Lambda} - \mu^2(\boldsymbol{\Lambda}\mathbf{1}\mathbf{1}^T\boldsymbol{\Lambda} + 2\boldsymbol{\Lambda}^2))^{-1} \boldsymbol{\Lambda} \mathbf{1} && \text{(B.112)} \\ &= \mu^2 J_{\min} \mathbf{1}^T \boldsymbol{\Lambda} [(2\boldsymbol{\mu}^{-1}\boldsymbol{I} - 2\boldsymbol{\Lambda} - \boldsymbol{\Lambda}\mathbf{1}\mathbf{1}^T)\mu^2\boldsymbol{\Lambda}]^{-1} \boldsymbol{\Lambda} \mathbf{1} && \text{(B.113)} \\ &= J_{\min} \mathbf{1}^T (2\boldsymbol{\mu}^{-1}\boldsymbol{I} - 2\boldsymbol{\Lambda} - \boldsymbol{\Lambda}\mathbf{1}\mathbf{1}^T)^{-1} \boldsymbol{\Lambda} \mathbf{1} \ . && \text{(B.114)} \end{aligned}$$

Now, by defining

$$\boldsymbol{G} = \boldsymbol{\mu}^{-1}\boldsymbol{I} - \boldsymbol{\Lambda} \tag{B.115}$$

and using the matrix inversion lemma from Eq. (4.5), we obtain

$$\begin{aligned} J_{\text{ex}} &= J_{\min} \mathbf{1}^T (2\boldsymbol{G} - \boldsymbol{\Lambda}\mathbf{1}\mathbf{1}^T)^{-1} \boldsymbol{\Lambda} \mathbf{1} && \text{(B.116)} \\ &= J_{\min} \mathbf{1}^T \left[\frac{1}{2}\boldsymbol{G}^{-1} + \frac{1}{2}\boldsymbol{G}^{-1}\boldsymbol{\Lambda}\mathbf{1} \left(1 - \frac{1}{2}\mathbf{1}^T\boldsymbol{G}^{-1}\boldsymbol{\Lambda}\mathbf{1}\right)^{-1} \mathbf{1}^T\boldsymbol{G}^{-1}\frac{1}{2} \right] \boldsymbol{\Lambda}\mathbf{1} && \text{(B.117)} \\ &= J_{\min} \frac{1}{2}\mathbf{1}^T\boldsymbol{G}^{-1}\boldsymbol{\Lambda}\mathbf{1} \left[1 + \left(1 - \frac{1}{2}\mathbf{1}^T\boldsymbol{G}^{-1}\boldsymbol{\Lambda}\mathbf{1}\right)^{-1} \frac{1}{2}\mathbf{1}^T\boldsymbol{G}^{-1}\boldsymbol{\Lambda}\mathbf{1} \right] && \text{(B.118)} \\ &= J_{\min} \frac{1}{2}\mathbf{1}^T\boldsymbol{G}^{-1}\boldsymbol{\Lambda}\mathbf{1} \left(1 - \frac{1}{2}\mathbf{1}^T\boldsymbol{G}^{-1}\boldsymbol{\Lambda}\mathbf{1}\right)^{-1} \ . && \text{(B.119)} \end{aligned}$$

Finally, from Eq. (B.72), we have that

$$f(\mu) = \frac{1}{2}\mathbf{1}^T\boldsymbol{\Lambda}\boldsymbol{G}^{-1}\mathbf{1} = \frac{1}{2}\mathbf{1}^T\boldsymbol{G}^{-1}\boldsymbol{\Lambda}\mathbf{1} \tag{B.120}$$

which leads to

$$\boxed{J_{\text{ex}} = J_{\min} \frac{f(\mu)}{1 - f(\mu)}} \ . \tag{B.121}$$

Misadjustment

The expression for the misadjustment is

$$\boxed{\mathcal{M} = \frac{J_{\text{ex}}}{J_{\min}} = \frac{f(\mu)}{1 - f(\mu)}} \ . \tag{B.122}$$