Solution Manual
for
Adaptive Filter Theory 5e

Created by

Simon Haykin

and

Kelvin Hall

McMaster University

HAMILTON, ONTARIO

CANADA

2014

PEARSON

Acknowledgments

the co-authors of this Solutions Manual would like to express their gratitudes to:

1. Professor Tulai Adali, University of Maryland, Baltimore County, for introducing the first co-autor Simon Haykin, to the Wirtiger Calculus for dealing with partial differentials of complex data.

2. Ashique Rupam Mahomood, Department of Computing Science University of Alberta, Alberta, Canada, for his kind help to provide several computer experiments on application of the Autostep Method and their solutions, both within Chapter 13 of the textbook on Adaptive Filter Theory, as well related problems at the end of the chapter.

3. Erkan Baser for permitting us to reproduce his graduate student project in adaptive filter theory, 2013; the reproduction is verbatim, presented as an appendix at the end of the solution manual: The project entailed a revisit to the Adaptive Equalization Experiment in Chapter 6 on the LMS algorithm. This time, however, the projected involved using the IDBD algorithm and the Autostep Method, as well as the LMS and RLS algorithms as basis for comparative evaluation.

4. Ashkan Amir, Ph.D. student at McMaster University for helping the second co-author, Kelvin Hall during early stages of the work done on this Solutions Manual for the book on Adaptive Filter Theory.

Simon Haykin
Kelvin Douglas Hall
McMaster University
August, 2014.

Table of Contents

Corrections for Question-Descriptions in the Textbook

Through the creation of the solution manual, several minor errors were noticed in the question descriptions in the textbook. Below is a collected list of which questions are affected and what changes are needed to be made. The information of what corrections are to be made is also present in the solutions manual at the question affected.

1.10 The expression $x(n) = \nu(n) + 0.75\nu(n-1) + 0.75\nu(n-2)$ should read $x(n) = \nu(n) + 0.75\nu(n-1) + 0.25\nu(n-2)$

5.6 The cost function should be $J_{\mathbf{s}}(\mathbf{w}) = |e(n)|^4$ not $J_{\mathbf{s}}(w) = |e(n)|^4$

5.6 a) The update formula should be given as:
$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + 2\mu\mathbf{u}(n)e^*(n)\,|e(n)|^2$
not $\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu\mathbf{u}(n-i)e^*(n)\,|e(n)|^2 \quad i = 0, 1, \ldots, M-1$

6.16 The problem is overdefined by providing both a noise variance and an AR process variance. The noise variance as such can be ignored. However if the solution is found with the prescribed noise variance the resulting graphs will be nearly identical to those included in the solution manual.

11.1 a) The question was meant to ask to show that $\mu^{-\frac{1}{2}}\tilde{\mathbf{w}}(n+1)$ equals
$\mu^{-\frac{1}{2}}\tilde{\mathbf{w}}(n) - \mu^{\frac{1}{2}}\mathbf{u}(n)\left(d(n) - \hat{\mathbf{w}}^T(n)\mathbf{u}(n)\right)$
not $\mu^{-\frac{1}{2}}\tilde{\mathbf{w}}(n) - \mu^{-\frac{1}{2}}\mathbf{u}(n)\left(d(n) - \hat{\mathbf{w}}^T(n)\mathbf{u}(n)\right)$

11.1 c) The last bracketed expression on the right hand side of the question should read $\left(d(n) - \hat{\mathbf{w}}^T(n)\mathbf{u}(n)\right)^2$ not $\left(d(n) - \tilde{\mathbf{w}}^T(n)\mathbf{u}(n)\right)^2$

11.1 d) The last bracketed expression on the right hand side of the question should read $\left(d(n) - \hat{\mathbf{w}}^T(n)\mathbf{u}(n)\right)^2$ not $\left(d(n) - \tilde{\mathbf{w}}^T(n)\mathbf{u}(n)\right)^2$

11.2 The denominator of the inequality should read
$\mu^{-1}\mathbf{w}^T\mathbf{w} + \sum_{n=0}^{i-1}\left(d(n) - \mathbf{w}^T\mathbf{u}(n)\right)^2$ not $\mu^{-1}\mathbf{w}^T\mathbf{w} + \sum_{n=0}^{i-1}\nu^2(n)$.

11.4 The question should be asking to find the optimizing $\mathbf{w}$ as shown by

$$\mathbf{w} = \left[\mu^{-1}\mathbf{I} - \mathbf{u}(i)\mathbf{u}^T(i)\right]^{-1}\left(\sum_{n=0}^{i-1} e(n)\mathbf{u}(n) - \hat{d}(i)\mathbf{u}(i)\right),$$

not

$$\mathbf{w} = \left[\mu\mathbf{I} - \mathbf{u}(i)\mathbf{u}^T(i)\right]^{-1}\left(\sum_{n=0}^{i-1} e(n)\mathbf{u}(n) - \hat{d}(i)\mathbf{u}(i)\right).$$

11.5 b) The experiment is described in Section 6.7 not 6.8.

11.7 d) Tildes are missing from above the step size parameters of the Normalized LMS algorithm entries in table P 11.1

Notes on the Computer Simulations and Provided Programs

The computer experiments completed for this solutions manual were completed almost exclusively using Matlab®, for ease of readability. To improve the accessibility of the solutions, to the users of this manual, specialized signal processing toolkits were not used in programs included. Graphical solutions are provided along with the .m files in case the user of the textbook is interested in completing the exercises in a different programming language, in which case a graphical solution is available for comparison.

The solutions of the computer problems in Chapter 13 were completed by Ashique Rupam Mahmood, Computer Science, University of Alberta, the creator of the Autostep algorithm. The solutions being completed prior to the rest of the manual are only available in the programming language python, which is similar to Matlab® and therefore quite readable.

Chapter 1

Problem 1.1

Let

$$r_u(k) = \mathbb{E}[u(n)u^*(n-k)] \tag{1}$$

$$r_y(k) = \mathbb{E}[y(n)y^*(n-k)] \tag{2}$$

we are given that

$$y(n) = u(n+a) - u(n-a) \tag{3}$$

Hence, substituting Equation (3) into Equation (2), and then using Equation (1), we get

$$\begin{aligned} r_y(k) =& \mathbb{E}[(u(n+a) - u(n-a))(u^*(n+a-k) - u^*(n-a-k))] \\ =& 2r_u(k) - r_u(2a+k) - r_u(-2a+k) \end{aligned}$$

Problem 1.2

We know that the correlation matrix $\mathbf{R}$ is Hermitian; that is to say that

$$\mathbf{R}^H = \mathbf{R}$$

Given that the inverse matrix $\mathbf{R}^{-1}$ exists, we may write

$$\mathbf{R}^{-1}\mathbf{R}^H = \mathbf{I}$$

where $\mathbf{I}$ is the identity matrix. Taking the Hermitian transpose of both sides:

$$\mathbf{R}\mathbf{R}^{-H} = \mathbf{I}$$

Hence,

$$\mathbf{R}^{-H} = \mathbf{R}^{-1}$$

That is, the inverse matrix $\mathbf{R}^{-1}$ is Hermitian.

Problem 1.3

For the case of a two-by-two matrix, it may be stated as

$$\begin{aligned}\mathbf{R}_u = &\ \mathbf{R}_s + \mathbf{R}_\nu \\ = &\ \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix} + \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix} \\ = &\ \begin{bmatrix} r_{11}+\sigma^2 & r_{12} \\ r_{21} & r_{22}+\sigma^2 \end{bmatrix}\end{aligned}$$

For $\mathbf{R}_u$ to be nonsingular, we require

$$\det(\mathbf{R}_u) \neq 0$$

$$(r_{11}+\sigma^2)(r_2 2+\sigma^2) - r_{12}r_{22} \neq 0$$

With $r_{12} = r_{21}$ for real data, this condition reduces to

$$(r_{11}+\sigma^2)(r_{22}+\sigma^2) - r_{12}^2 \neq 0$$

Since this is a quadratic in σ^2, we may impose the following conditions on σ^2 for nonsingularity of $\mathbf{R}_u$:

$$\sigma^2 \neq \frac{1}{2}(r_{11}+r_{22})\left(\sqrt{1-\frac{4\Delta_r}{(r_{11}+r_{22})^2-1}}\right)$$

where $\Delta_r = r_{11}r_{22} - r_{12}^2$

Problem 1.4

We are given

$$\mathbf{R} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

This matrix is positive definite because it satisfies the condition:

$$\begin{aligned} a^T\mathbf{R}\, a &= \begin{bmatrix} a_1 & a_2 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \\ &= a_1^2 + 2a_1a_2 + a_2^2 \\ &= (a_1 + a_2)^2 > 0 \text{ for all nonzero values of } a_1 \text{ and } a_2 \end{aligned}$$

But the matrix $\mathbf{R}$ is singular because:

$$\det(\mathbf{R}) = (1)^2 - (1)^2 = 0$$

Hence, it is possible for a matrix to be both positive definite and singular at the same time.

Problem 1.5

a)

$$\mathbf{R}_{M+1} = \begin{bmatrix} r(0) & \mathbf{r}^H \\ \mathbf{r} & \mathbf{R}_M \end{bmatrix} \tag{1}$$

Let

$$\mathbf{R}_{M+1}^{-1} = \begin{bmatrix} a & \mathbf{b}^H \\ \mathbf{b} & \mathbf{C}_M \end{bmatrix} \tag{2}$$

where$a$, $\mathbf{b}$ and $\mathbf{C}$ are to be determined. Multiply Equation (1) by Equation (2):

$$\mathbf{I}_{M+1} = \begin{bmatrix} r(0) & \mathbf{r}^H \\ \mathbf{r} & \mathbf{R}_M \end{bmatrix} \begin{bmatrix} a & \mathbf{b}^H \\ \mathbf{b} & \mathbf{C} \end{bmatrix}$$

Where $\mathbf{I}_{M+1}$ is the identity matrix. Therefore,

$$r(0)a + \mathbf{r}^H\mathbf{b} = 1 \tag{3}$$

$$\mathbf{r}a + \mathbf{R}_M\mathbf{b} = \mathbf{0} \tag{4}$$

$$\mathbf{r}\mathbf{b}^H + \mathbf{R}_M\mathbf{C} = \mathbf{I}_M \tag{5}$$

$$r(0)\mathbf{b}^H + \mathbf{r}^H\mathbf{C} = \mathbf{0} \tag{6}$$

Equation (4) can be rearranged to solve for b as:

$$b = -\mathbf{R}_M^{-1}\mathbf{r}a \tag{7}$$

Hence, from equations (3) and (7):

$$a = \frac{1}{r(0) - \mathbf{r}^H \mathbf{R}_M^{-1} \mathbf{r}} \tag{8}$$

Correspondingly,

$$b = -\frac{\mathbf{R}_M^{-1} \mathbf{r}\mathbf{r}^H \mathbf{R}_M^{-1}}{r(0) - \mathbf{r}^H \mathbf{R}_M^{-1} \mathbf{r}} \tag{9}$$

From Equation (5):

$$\mathbf{C} = \mathbf{R}_M^{-1} - \mathbf{R}_M^{-1} \mathbf{r}\mathbf{b}^H$$

$$\mathbf{C} = \mathbf{R}_M^{-1} + \frac{\mathbf{R}_M^{-1} \mathbf{r}\mathbf{r}^H \mathbf{R}_M^{-1}}{r(0) - \mathbf{r}^H \mathbf{R}_M^{-1} \mathbf{r}} \tag{10}$$

As a check, the results of Equations (9) and (10) should satisfy Equation (6)

$$\begin{aligned} r(0)\mathbf{b}^H + \mathbf{r}^H \mathbf{C} = & -\frac{r(0)\mathbf{r}^H \mathbf{R}_M^{-1}}{r(0) - \mathbf{r}^H \mathbf{R}_M^{-1} \mathbf{r}} + \mathbf{r}^H \mathbf{R}_M^{-1} + \frac{\mathbf{r}^H \mathbf{R}_M^{-1} \mathbf{r}\mathbf{r}^H \mathbf{R}_M^{-1}}{r(0) - \mathbf{r}^H \mathbf{R}_M^{-1} \mathbf{R}} \\ = & \mathbf{0} \end{aligned}$$

We have thus shown that

$$\begin{aligned} \mathbf{R}_{M+1}^{-1} = & \begin{bmatrix} 0 & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_M^{-1} \end{bmatrix} + a \begin{bmatrix} 1 & -\mathbf{r}^H \mathbf{R}_M^{-1} \\ \mathbf{R}_M^{-1} \mathbf{r} & \mathbf{R}_M^{-1} \mathbf{r}\mathbf{r}^H \mathbf{R}_M^{-1} \end{bmatrix} \\ = & \begin{bmatrix} 0 & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_M^{-1} \end{bmatrix} + a \begin{bmatrix} 1 \\ -\mathbf{R}_M^{-1} \mathbf{r} \end{bmatrix} \begin{bmatrix} 1 & -\mathbf{r}^{\mathbf{H}} \mathbf{R}_M^{-1} \end{bmatrix} \end{aligned}$$

where the scalar a is defined by Equation (8)

b)

$$\mathbf{R}_{M+1} = \begin{bmatrix} \mathbf{R}_M & \mathbf{r}^{B*} \\ \mathbf{r}^{BT} & r(0) \end{bmatrix} \tag{11}$$

Let

$$\mathbf{R}_{M+1}^{-1} = \begin{bmatrix} \mathbf{D} & \mathbf{e} \\ \mathbf{e}^H & f \end{bmatrix} \tag{12}$$

where $\mathbf{D}$, $\mathbf{e}$ and f are to be determined. Multiplying Equation (11) by Equation (12) you get:

$$\mathbf{I}_{M+1} = \begin{bmatrix} \mathbf{R}_M & \mathbf{r}^{B*} \\ \mathbf{r}^{BT} & r(0) \end{bmatrix} \begin{bmatrix} \mathbf{D} & \mathbf{e} \\ \mathbf{e}^H & f \end{bmatrix}$$

Therefore:

$$\mathbf{R}_M\mathbf{D} + \mathbf{r}^{B*}\mathbf{e}^H = \mathbf{I} \tag{13}$$

$$\mathbf{R}_M\mathbf{e} + \mathbf{r}^{B*}f = \mathbf{0} \tag{14}$$

$$\mathbf{r}^{BT}\mathbf{e} + r(0)f = 1 \tag{15}$$

$$\mathbf{r}^{BT}\mathbf{D} + r(0)\mathbf{e}^H = \mathbf{0} \tag{16}$$

From Equation (14):

$$\mathbf{e} = -\mathbf{R}_M^{-1}\mathbf{r}^{B*} \tag{17}$$

Hence, from Equation (15) and Equation (17):

$$f = \frac{1}{r(0) - \mathbf{r}^{BT}\mathbf{R}_M^{-1}\mathbf{r}^{B*}} \tag{18}$$

Correspondingly,

$$\mathbf{e} = -\frac{\mathbf{R}_M^{-1}\mathbf{r}^{B*}}{r(0) - \mathbf{r}^{BT}\mathbf{R}_M^{-1}\mathbf{r}^{B*}} \tag{19}$$

From Equation (13):

$$\begin{aligned} \mathbf{D} &= \mathbf{R}_M^{-1} - \mathbf{R}_M^{-1}\mathbf{r}^{B*}\mathbf{e}^H \\ &= \mathbf{R}_M^{-1} + \frac{\mathbf{R}_M^{-1}\mathbf{r}^{B*}\mathbf{r}^{BT}\mathbf{R}_M^{-1}}{r(0) - \mathbf{r}^{BT}\mathbf{R}_M^{-1}\mathbf{r}^{B*}} \end{aligned} \tag{20}$$

As a check, the results of Equation (19) and Equation (20) must satisfy Equation (16):

$$\mathbf{r}^{BT}\mathbf{D} + r(0)\mathbf{e}^H = \mathbf{0}$$

$$\mathbf{r}^{BT}\mathbf{R}_M^{-1} + \frac{\mathbf{r}^{BT}\mathbf{R}_M^{-1}\mathbf{r}^{B*}\mathbf{r}^{BT}\mathbf{R}_M^{-1}}{r(0) - \mathbf{r}^{BT}\mathbf{R}_M^{-1}\mathbf{r}^{B*}} - \frac{r(0)\mathbf{r}^{BT}\mathbf{R}_M^{-1}}{r(0) - \mathbf{r}^{BT}\mathbf{R}_M^{-1}\mathbf{r}^{B*}} = \mathbf{0}$$

We have thus shown that

$$\begin{aligned} \mathbf{R}_{M+1}^{-1} &= \begin{bmatrix} \mathbf{R}_M^{-1} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} + f \begin{bmatrix} \mathbf{R}_M^{-1}\mathbf{r}^{B*}\mathbf{r}^{BT}\mathbf{R}_M^{-1} & \mathbf{R}_M^{-1}\mathbf{r}^{B*} \\ -\mathbf{r}^{BT}\mathbf{R}_M^{-1} & 1 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{R}_M^{-1} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} + f \begin{bmatrix} -\mathbf{R}_M^{-1}\mathbf{r}^{B*} \\ 1 \end{bmatrix} \begin{bmatrix} -\mathbf{r}^{BT}\mathbf{R}_M^{-1} & 1 \end{bmatrix} \end{aligned}$$

where the scalar f is defined by Equation (18)

Problem 1.6

a)

We express the difference equation describing the first-order AR process $u(n)$ as

$$u(n) = \nu(n) + w_1 u(n-1)$$

where $w_1 = -a_1$. Solving the equation by repeated substitution, we get

$$\begin{aligned} u(n) =& \nu(n) + w_1\nu(n-1) + w_1 u(n-2) \\ =& \nu(n) + w_1\nu(n-1) + w_1^2\nu(n-2) + \ldots + w_1^{n-1}\nu(1) \end{aligned} \tag{1}$$

Here we used the initial condition

$$u(0) = 0$$

Taking the expected value of both sides of Equation (1) and using

$$\mathbb{E}[\nu(n)] = \mu$$

we get the geometric series

$$\begin{aligned} \mathbb{E}[u(n)] =& \quad \mu + w_1\mu + w_1^2\mu + \ldots + w_1^{n-1}\mu \\ =& \quad \left\{ \begin{array}{ll} \mu(\frac{1-w_1^n}{1-w_1}), & w_1 \neq 1 \\ \mu n, & w_1 = 1 \end{array} \right\} \end{aligned}$$

This result shows that if $\mu \neq 0$, then $\mathbb{E}[u(n)]$ is a function of time n. Accordingly, the AR process $u(n)$ is not stationary. If, however, the AR parameter satisfies the condition:

$$|a_1| < 1 \text{ or } |w_1| < 1$$

then

$$\mathbb{E}[u(n)] \to \frac{\mu}{1-w_1} \text{ as } n \to \infty$$

Under this condition, we say that the AR process is asymptotically stationary to order one.

b)

When the white noise process $\nu(n)$ has zero mean, the AR process $u(n)$ will likewise have zero mean. Then

$$\text{var}[\nu(n)] = \sigma_\nu^2$$

$$\text{var}[u(n)] = \mathbb{E}[u^2(n)] \tag{2}$$

Substituting Equation (1) into Equation (2), and recognizing that for the white noise process

$$\mathbb{E}[\nu(n)\nu(k)] = \begin{cases} \sigma_\nu^2 & n = k \\ 0, & n \neq k \end{cases} \tag{3}$$

we get the geometric series

$$\begin{aligned} \text{var}[u(n)] =& \sigma_\nu^2(1 + w_1^2 + w_1^4 + \ldots + w_1^{2n-2}) \\ =& \begin{cases} \sigma_\nu^2(\dfrac{1 - w_1^{2n}}{1 - w_1^2}), & w_1 \neq 1 \\ \sigma_\nu^2 n, & w_1 = 1 \end{cases} \end{aligned}$$

When $|a_1| < 1$ or $|w_1| < 1$, then

$$\text{var}[u(n)] \approx \frac{\sigma_\nu^2}{1 - w_1^2} = \frac{\sigma_\nu^2}{1 - a_1^2} \text{ for large } n$$

c)

The autocorrelation function of the AR process $u(n)$ equals $\mathbb{E}[u(n)u(n-k)]$. Substituting Equation (1) into this formula, and using Equation (3), we get

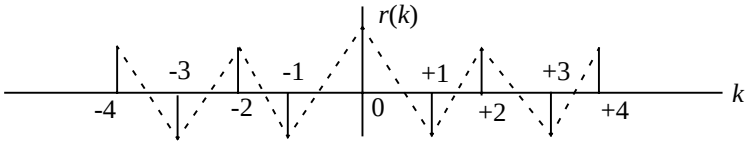
$$\begin{aligned} \mathbb{E}[u(n)u(n-k)] =& \sigma_\nu^2(w_1^k + w_1^{k+2} + \ldots + w_1^{k+2n-2}) \\ =& \begin{cases} \sigma_\nu^2 w_1^k(\frac{1-w_1^{2n}}{1-w_1^2}), & w_1 \neq 1 \\ \sigma_\nu^2 n, & w_1 = 1 \end{cases} \end{aligned}$$

For $|a_1| < 1$ or $|w_1| < 1$, we may therefore express this autocorrelation function as

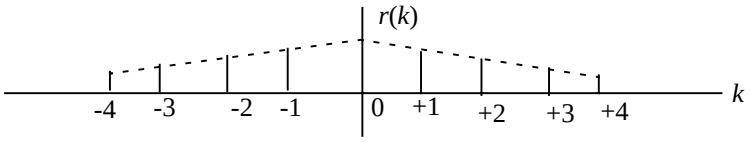
$$\begin{aligned} r(k) =& \mathbb{E}[u(n)u(n-k)] \\ \approx& \frac{\sigma_\nu^2 w_1^k}{1 - w_1^2} \text{ for large } n \end{aligned}$$

Case 1: $0 < a_1 < 1$

In this case, $w_1 = -a_1$ is negative, and $r(k)$ varies with k as follows:



Case 2: $-1 < a_1 < 0$
In this case, $w_1 = -a_1$ is positive, and $r(k)$ varies with k as follows:



Problem 1.7

a)

The second-order AR process $u(n)$ is described by the difference equation:

$$u(n) = u(n-1) - 0.5u(n-2) + \nu(n)$$

which, rewritten, states

$$w_1 = 1$$

$$w_2 = -0.5$$

as the AR parameters are equal to:

$$a_1 = -1$$

$$a_2 = 0.5$$

Accordingly, the Yule-Walker equation may be written as:

$$\begin{bmatrix} \mathbf{r}(0) & \mathbf{r}(1) \\ \mathbf{r}(1) & \mathbf{r}(0) \end{bmatrix} \begin{bmatrix} 1 \\ -0.5 \end{bmatrix} = \begin{bmatrix} \mathbf{r}(1) \\ \mathbf{r}(2) \end{bmatrix}$$

b)

Writing the Yule-Walker equations in expanded form:

$$\mathbf{r}(0) - 0.5\mathbf{r}(1) = \mathbf{r}(1)$$

$$\mathbf{r}(1) - 0.5\mathbf{r}(0) = \mathbf{r}(2)$$

Solving the first relation for $\mathbf{r}(1)$:

$$\mathbf{r}(1) = \frac{2}{3}\mathbf{r}(0) \tag{1}$$

Solving the second relation for $\mathbf{r}(2)$:

$$\mathbf{r}(2) = \frac{1}{6}\mathbf{r}(0) \tag{2}$$

c)

Since the noise $\nu(n)$ has zero mean, the associated AR process $u(n)$ will also have zero mean. Hence,

$$\begin{aligned} \text{var}[u(n)] = &\ \mathbb{E}[(u^2)] \\ = &\ \mathbf{r}(0) \end{aligned}$$

It is known that

$$\begin{aligned} \sigma_\nu^2 =& \sum_{k=0}^{2} a_k \mathbf{r}(k) \\ =& \mathbf{r}(0) + a_1\mathbf{r}(1) + a_2\mathbf{r}(2) \end{aligned} \tag{3}$$

Substituting Equation (1) and Equation (2) into Equation (3), and solving for $\mathbf{r}(0)$, we get:

$$\mathbf{r}(0) = \frac{\sigma_\nu^2}{1 + \frac{2}{3}a_1 + \frac{1}{6}a_2} = 1.2$$

Problem 1.8

By Definition,

$$\begin{aligned} P_0 =& \text{ Average power of the AR process } u(n) \\ =& \mathbb{E}[|u(n)|^2] \\ =& \mathbf{r}(0) \end{aligned} \tag{1}$$

where $\mathbf{r}(0)$ is the autocorrelation function of $u(n)$ with zero lag. We note that

$$\{a_1, a_2, \ldots, a_M\} \rightleftharpoons \left\{ \frac{\mathbf{r}(1)}{\mathbf{r}(0)}, \frac{\mathbf{r}(2)}{\mathbf{r}(0)}, \ldots, \frac{\mathbf{r}(M)}{\mathbf{r}(0)} \right\}$$

Equivalently, except for the scaling factor $\mathbf{r}(0)$,

$$\{a_1, a_2, \ldots, a_M\} \rightleftharpoons \{\mathbf{r}(1), \mathbf{r}(2), \ldots, \mathbf{r}(M)\} \tag{2}$$

Combining Equation (1) and Equation (2):

$$\{P_0, a_1, a_2, \ldots, a_M\} \rightleftharpoons \{\mathbf{r}(0), \mathbf{r}(1), \mathbf{r}(2), \ldots, \mathbf{r}(M)\} \tag{3}$$

Problem 1.9

a)

The transfer function of the MA model of Fig. 1.3 is

$$H(z) = 1 + b_1^* z^{-1} + b_2^* z^{-2} + \ldots + b_K^* z^{-K}$$

b)

The transfer function of the ARMA model of Fig. 1.4 is

$$H(z) = \frac{b_0 + b_1^* z^{-1} + b_2^* z^{-2} + \ldots + b_K^* z^{-K}}{1 + a_1^* z^{-1} + a_2^* z^{-2} + \ldots + a_M^* z^{-M}}$$

c)

The ARMA model reduces to an AR model when

$$b_0 = b_1 = \ldots = b_K = 0$$

The ARMA model reduces to MA model when

$$a_1 = a_2 = \ldots = a_M = 0$$

Problem 1.10

* Taking the z-transform of both sides of the correct equation:

$$X(z) = (1 + 0.75z^{-1} + 0.25z^{-2})V(z)$$

Hence, the transfer function of the MA model is:

$$\begin{aligned}\frac{X(z)}{V(z)} =& 1 + 0.75z^{-1} + 0.75z^{-1} \\ =& \frac{1}{(1 + 0.75z^{-1} + 0.75z^{-1})^{-1}} \end{aligned} \tag{1}$$

Using long division we may perform the following expansion of the denominator in Equation (1):

$$\begin{aligned}&(1 + 0.75z^{-1} + 0.75z^{-1})^{-1} \\ &\qquad = 1 - \frac{3}{4}z^{-1} + \frac{5}{16}z^{-2} - \frac{3}{64}z^{-3} - \frac{11}{256}z^{-4} - \frac{45}{1024}z^{-5} \\ &\qquad\quad - \frac{91}{4096}z^{-6} + \frac{93}{16283}z^{-7} - \frac{85}{65536}z^{-8} - \frac{627}{262144}z^{-9} + \frac{1541}{1048576}z^{-10} + \dots \\ &\qquad \approx 1 - 0.75z^{-1} + 0.3125z^{-2} - 0.0469z^{-3} - 0.043z^{-4} - 0.0439z^{-5} \\ &\qquad\quad - 0.0222z^{-6} + 0.0057z^{-7} - 0.0013z^{-8} - 0.0024z^{-9} + 0.0015z^{-10}\end{aligned} \tag{2}$$

a)

$$M = 2$$

Retaining terms in Equation (2) up to z^{-2}, we may approximate the MA model with an AR model of order two as follows:

$$\frac{X(z)}{V(z)} \approx \frac{1}{1 - 0.75z^{-1} + 0.3125z^{-2}}$$

*Correction: the question was meant to ask the reader to consider an MA process $x(n)$ of order two described by the difference equation

$$x(n) = \nu(n) + 0.75\nu(n-1) + 0.25\nu(n-2)$$

not the equation

$$x(n) = \nu(n) + 0.75\nu(n-1) + 0.75\nu(n-2)$$

b)

$$M = 5$$

Retaining terms in Equation (2) up to z^{-5}, we may approximate the MA model with an AR model of order two as follows:

$$\frac{X(z)}{V(z)} \approx \frac{1}{1 - 0.75z^{-1} + 0.3125z^{-2} - 0.0469z^{-3} - 0.043z^{-4} + 0.0439z^{-5}}$$

c)

$$M = 10$$

Retaining terms in Equation (2) up to z^{-10}, we may approximate the MA model with an AR model of order two as follows:

$$\frac{X(z)}{V(z)} \approx \frac{1}{D(z)}$$

where $D(z)$ is given by the polynomial on the right-hand side of Equation (2).

Problem 1.11

a)

The filter output is

$$x(n) = \mathbf{w}^H \mathbf{u}(n)$$

where $\mathbf{u}(n)$ is the tap-input vector. The average power of the filter output is therefore

$$\begin{aligned} \mathbb{E}[|x(n)|^2] &= \mathbb{E}[\mathbf{w}^H \mathbf{u}(n)\mathbf{u}^H(n)\mathbf{w}] \\ &= \mathbf{w}^H \mathbb{E}[\mathbf{u}(n)\mathbf{u}^H(n)]\mathbf{w} \\ &= \mathbf{w}^H \mathbf{R}\mathbf{w} \end{aligned}$$

b)

If $\mathbf{u}(n)$ is extracted from a zero-mean white noise with variance σ^2, then

$$\mathbf{R} = \sigma^2 \mathbf{I}$$

where $\mathbf{I}$ is the identity matrix. Hence,

$$\mathbb{E}[|x(n)|^2] = \sigma^2 \mathbf{w}^H \mathbf{w}$$

Problem 1.12

a)

The process $u(n)$ is a linear combination of Gaussian samples. Hence, $u(n)$ is Gaussian.

b)

From inverse filtering, we recognize that $\nu(n)$ may also be expressed as a linear combination of samples relating to $u(n)$. Hence, if $u(n)$ is Gaussian, then $\nu(n)$ is also Gaussian.

Problem 1.13

a)

From the Gaussian moment factoring theorem:

$$\begin{aligned}\mathbb{E}[(u_1^* u_2)^k] =& \mathbb{E}[u_1^* \dots u_1^* u_2 \dots u_2] \\ =& k! \mathbb{E}[u_1^* u_2] \dots \mathbb{E}[u_1^* u_2] \\ =& k! (\mathbb{E}[u_1^* u_2])^k \end{aligned} \tag{1}$$

b)

By allowing u_2 = u_1 = u, Equation (1) reduces to:

$$\mathbb{E}[|u|^{2k}] = k! (\mathbb{E}[|u|^2])^k$$

Problem 1.14

It is not permissible to interchange the order of expectation and limiting operation in Equation (1.113). The reason is that the expectation is a linear operation, whereas the limiting operation with respect to the number of samples N is nonlinear.

Problem 1.15

The filter output is

$$y(n) = \sum_i h(i) u(n-i)$$

Similarly, we may write

$$y(m) = \sum_k h(k)u(m-k)$$

Hence,

$$\begin{aligned} r_y(n,m) = &\ \mathbb{E}[y(n)y^*(m)] \\ = &\ \mathbb{E}\left[\sum_i h(i)u(n-i)\sum_k h^*(k)u^*(m-k)\right] \\ = &\ \sum_i\sum_k h(i)h^*(k)\mathbb{E}\left[u(n-i)u^*(m-k)\right] \\ = &\ \sum_i\sum_k h(i)h^*(k)r_u(n-i,m-k) \end{aligned}$$

Problem 1.16

The mean-square value of the filter output response to white noise input is

$$P_0 = \frac{2\sigma^2\Delta\omega}{\pi}$$

The value P_0 is linearly proportional to the filter bandwidth $\Delta\omega$. This relation holds irrespective of how small $\Delta\omega$ is compared to the mid-band frequency of the filter.

Problem 1.17

a)

The variance of the filter output is

$$\sigma_y^2 = \frac{2\sigma^2\Delta\omega}{\pi}$$

It has been stated that

$$\sigma^2 = 0.1 \text{ volts}^2$$

$$\Delta\omega = 2\pi \times 1 \text{ radians/sec}$$

Hence,

$$\sigma_y^2 = \frac{2 \times 0.1 \times 2\pi}{\pi} = 0.4 \text{ volts}^2$$

b)

The pdf of the filter output y is

$$\begin{aligned} f(y) = & \frac{1}{\sqrt{2\pi}\sigma_y} \exp(-y^2/0.8) \\ = & \frac{3.1623}{\sqrt{2\pi}} \exp(-y^2/0.8) \end{aligned}$$

Problem 1.18

a)

We are given

$$U_k = \sum_0^{N-1} u(n) \exp(-\mathrm{j}\, n\omega_k), \quad k = 0, 1, ..., N-1$$

where $u(n)$ is real valued and

$$\omega_k = \frac{2\pi}{N} k$$

Hence,

$$\begin{aligned} \mathbb{E}[U_k U_l^*] = & \mathbb{E}\left[\sum_{n=0}^{N-1}\sum_{m=0}^{N-1} u(n)u(m) \exp(-\mathrm{j}\, n\omega_k + \mathrm{j}\, m\omega_l)\right] \\ = & \sum_{n=0}^{N-1}\sum_{m=0}^{N-1} \exp(-\mathrm{j}\, n\omega_k + \mathrm{j}\, m\omega_l)\mathbb{E}[u(n)u(m)] \\ = & \sum_{n=0}^{N-1}\sum_{m=0}^{N-1} \exp(-\mathrm{j}\, n\omega_k + \mathrm{j}\, m\omega_l) r(n-m) \\ = & \sum_{n=0}^{N-1} \exp(\mathrm{j}\, m\omega_k) \sum_{m=0}^{N-1} r(n-m) \exp(-\mathrm{j}\, n\omega_k) \end{aligned} \tag{1}$$

By definition, we also have

$$\sum_{n=0}^{N-1} r(n)\exp(-\mathrm{j}\, n\omega_k) = S_k$$

Moreover, since $r(n)$ is periodic with period N, we may invoke the time-shifting property of the discrete Fourier transform to write

$$\sum_{n=0}^{N-1} r(n-m)\exp(-\mathrm{j}\, n\omega_k) = \exp(-\mathrm{j}\, m\omega_k)S_k$$

Recognizing that $\omega_k = (2\pi/N)k$, Equation (1) reduces to

$$\begin{aligned}\mathbb{E}[U_k U_l^*] = &\quad S_k \textstyle\sum_{m=0}^{N-1}\exp(\mathrm{j}\, m(\omega_l - \omega_k)) \\ &= \begin{cases} S_k & l = k \\ 0, & \text{otherwise} \end{cases}\end{aligned}$$

b)

Part **A**) shows that the complex spectral samples U_k are uncorrelated. If they are Gaussian, then they will also be statistically independent. Hence,

$$f_{\mathbf{U}}(U_0, U_1, \ldots, U_{N-1}) = \frac{1}{(2\pi)^N \det(\Lambda)}\exp\left(-\frac{1}{2}\mathbf{U}^H\Lambda\mathbf{U}\right)$$

where

$$\mathbf{U} = [U_0, U_1, \ldots, U_{N-1}]^T$$

$$\begin{aligned}\Lambda = &\quad \frac{1}{2}\mathbb{E}[\mathbf{U}\mathbf{U}^H] \\ = &\quad \frac{1}{2}\mathrm{diag}(S_0, S_1, ..., S_{N-1})\end{aligned}$$

$$\det(\Lambda) = \frac{1}{2^N}\prod_{k=0}^{N-1} S_k$$

Therefore,

$$f_{\mathbf{U}}(U_0, U_1, \ldots, U_{N-1}) = \frac{1}{(2\pi)^N 2^{-N} \prod_{k=0}^{N-1} S_k} \exp\left(-\frac{1}{2}\sum_{k=0}^{N-1} \frac{|U_k|^2}{\frac{1}{2}S_k}\right)$$

$$= \pi^{-N} \exp\left(\sum_{k=0}^{N-1}\left(-\frac{|U_k|^2}{S_k}\right) - \ln S_k\right)$$

Problem 1.19

The mean-square value of the increment process $\mathrm{d}\, z(\omega)$ is

$$\mathbb{E}[|\,\mathrm{d}\, z(\omega)|^2] = S(\omega)\,\mathrm{d}\,\omega$$

Hence, $\mathbb{E}[|\,\mathrm{d}\, z(\omega)|^2]$ is measured in watts.

Problem 1.20

The third-order cumulant of a process $u(n)$ is

$$\begin{aligned} c_3(\tau_1, \tau_2) &= \mathbb{E}[u(n)u(n+\tau_1)u(n+\tau_2)] \\ &= \text{third-order moment.} \end{aligned}$$

All odd-order moments of a Gaussian process are known to be zero; hence,

$$c_3(\tau_1, \tau_2) = 0$$

The fourth-order cumulant is

$$\begin{aligned} c_r(\tau_1, \tau_2, \tau_3) = \; & \mathbb{E}[u(n)u(n+\tau_1)u(n+\tau_2)u(n+\tau_3)] \\ & -\mathbb{E}[u(n)u(n+\tau_1)]\mathbb{E}[u(n+\tau_2)u(n+\tau_3)] \\ & -\mathbb{E}[u(n)u(n+\tau_2)]\mathbb{E}[u(n+\tau_1)u(n+\tau_3)] \\ & -\mathbb{E}[u(n)u(n+\tau_3)]\mathbb{E}[u(n+\tau_1)u(n+\tau_2)] \end{aligned}$$

For the special case of τ=τ_1=τ_2=τ_3, the fourth-order moment of a zero-mean Gaussian process of variance σ^2 is $3\sigma^4$, and its second-order moments of σ^2. Hence, the fourth-order cumulant is zero. Indeed, all cumulants higher than order two are zero

Problem 1.21

The trispectrum is

$$C_4(\omega_1,\omega_2,\omega_3)=\sum_{\tau_1=-\infty}^{\infty}\sum_{\tau_2=-\infty}^{\infty}\sum_{\tau_3=-\infty}^{\infty}c_4(\tau_1,\tau_2,\tau_3)\exp(-\mathrm{j}(\omega_1\tau_1+\omega_2\tau_2+\omega_3\tau_3))$$

Let the process be passed through a three-dimensional band-pass filter centered on ω_1, ω_2, and ω_3. We assume that the bandwidth (along each dimension) is small compared to the respective center frequency. The average power of the filter output is therefore proportional to the trispectrum, $C_4(\omega_1,\omega_2,\omega_3)$.

Problem 1.22

a)

Starting with the formula

$$c_k(\tau_1,\tau_2,\ldots,\tau_{k-1})=\gamma_k\sum_{i=-\infty}^{\infty}h_ih_{i+\tau_1}\ldots h_{i+\tau_{k-1}}$$

The third-order cumulant of the filter output is

$$c_3(\tau_1,\tau_2)=\gamma_3\sum_{i=-\infty}^{\infty}h_ih_{i+\tau_1}h_{i+\tau_2}$$

where γ_3 is the third-order cumulant of the filter input. The bispectrum is

$$\begin{aligned}c_3(\tau_1,\tau_2)=&\gamma_3\sum_{\tau_1=-\infty}^{\infty}\sum_{\tau_2=-\infty}^{\infty}c_3(\tau_1,\tau_2)\exp(-\mathrm{j}(\omega_1\tau_1+\omega_2\tau_2))\\=&\gamma_3\sum_{i=-\infty}^{\infty}\sum_{\tau_1=-\infty}^{\infty}\sum_{\tau_2=-\infty}^{\infty}h_ih_{i+\tau_1}h_{i+\tau_2}\exp(-\mathrm{j}(\omega_1\tau_1+\omega_2\tau_2))\end{aligned}$$

Hence,

$$C_3(\omega_1,\omega_2)=\gamma_3H\left(e^{\mathrm{j}\,\omega_1}\right)H\left(e^{\mathrm{j}\,\omega_2}\right)H^*\left(e^{\mathrm{j}(\omega_1+\omega_2)}\right)\tag{1}$$

b)

From the formula found in part **a)**, Equation (1), can be clearly deduced that

$$\arg[C_3(\omega_1,\omega_2)]=\arg\left[H\left(e^{\mathrm{j}\,\omega_1}\right)\right]+\arg\left[H\left(e^{\mathrm{j}\,\omega_2}\right)\right]-\arg\left[H\left(e^{\mathrm{j}(\omega_1+\omega_2)}\right)\right]$$

Problem 1.23

The output of a filter, which is defined by the impulse response h_i due to an input $u(i)$, is given by the convolution sum

$$y(n) = \sum_i h_i u(n-i)$$

The third-order cumulant of the filter output is, for example,

$$\begin{aligned} C_3(\tau_1, \tau_2) =& \mathbb{E}[y(n)y(n+\tau_1)y(n+\tau_2)] \\ =& \mathbb{E}\left[\sum_i h_i u(n-i) \sum_k h_k u(n+\tau_1-k) \sum_l h_l u(n+\tau_2-l)\right] \\ =& \mathbb{E}\left[\sum_i h_i u(n-i) \sum_k h_{k+\tau_1} u(n-k) \sum_l h_{l+\tau_2} u(n-l)\right] \\ =& \sum_i \sum_k \sum_l h_i h_{k+\tau_1} h_{l+\tau_2} \mathbb{E}[u(n-i)u(n-k)u(n-l)] \end{aligned}$$

For an input sequence of independent and identically distributed random variables, we note that

$$\mathbb{E}[u(n-i)u(n-k)u(n-l)] = \begin{cases} \gamma_3, & i=k=l \\ 0, & \text{otherwise} \end{cases}$$

Hence,

$$C_3(\tau_1, \tau_2) = \gamma_3 \sum_{i=-\infty}^{\infty} h_i h_{i+\tau_1} h_{i+\tau_2}$$

In general, we may thus write

$$C_k(\tau_1, \tau_2, \ldots, \tau_{k-1}) = \gamma_k \sum_{i=-\infty}^{\infty} h_i h_{i+\tau_1} \ldots h_{i+\tau_{k-1}}$$

Problem 1.24

By definition:

$$r^{(\alpha)}(k) = \frac{1}{N} \sum_{n=0}^{N-1} \mathbb{E}[u(n)u^*(n-k)e^{-\mathrm{j}\,2\pi\alpha n}]e^{\mathrm{j}\,\pi\alpha k}$$

Hence,

$$r^{(\alpha)}(-k) = \frac{1}{N}\sum_{n=0}^{N-1}\mathbb{E}[u(n)u^*(n+k)e^{-\mathrm{j}\,2\pi\alpha n}]e^{-\mathrm{j}\,\pi\alpha k}$$

$$r^{(\alpha)*}(k) = \frac{1}{N}\sum_{n=0}^{N-1}\mathbb{E}[u^*(n)u(n-k)e^{\mathrm{j}\,2\pi\alpha n}]e^{-\mathrm{j}\,\pi\alpha k}$$

We are told that the process $u(n)$ is cyclostationary, which means that

$$\mathbb{E}[u(n)u^*(n+k)e^{-\mathrm{j}\,2\pi\alpha n}] = \mathbb{E}[u^*(n)u(n-k)e^{\mathrm{j}\,2\pi\alpha n}]$$

It follows therefore that

$$r^{(\alpha)}(-k) = r^{(\alpha)*}(k)$$

Problem 1.25

For $\alpha = 0$, the input to the time-average cross-correlator reduces to the squared amplitude of a narrow-band filter with mid-band frequency ω. Correspondingly, the time-average cross-correlator reduces to an average power meter. Thus, for $\alpha = 0$, the instrumentation of Fig. 1.16 reduces to that of Fig. 1.13 in the book.

Chapter 2

Problem 2.1

a)

Let

$$w_k = x + \mathrm{j}\, y$$

$$p(-k) = a + \mathrm{j}\, b$$

We may then write

$$\begin{aligned} f =& w_k p^*(-k) \\ =& (x + \mathrm{j}\, y)(a - \mathrm{j}\, b) \\ =& (ax + by) + \mathrm{j}(ay - bx) \end{aligned}$$

Letting

$$f = u + \mathrm{j}\, v$$

where

$$u = ax + by$$

$$v = ay - bx$$

Hence,

$$\frac{\partial u}{\partial x} = a \qquad \frac{\partial u}{\partial y} = b$$

$$\frac{\partial v}{\partial y} = a \qquad \frac{\partial v}{\partial x} = -b$$

From these results we can immediately see that

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}$$

$$\frac{\partial v}{\partial x} = -\frac{\partial u}{\partial y}$$

In other words, the product term $w_k p^*(-k)$ satisfies the Cauchy-Riemann equations, and so this term is analytic.

b)

Let

$$\begin{aligned} f =& w_k p^*(-k) \\ =& (x - \mathrm{j}\, y)(a + \mathrm{j}\, b) \\ =& (ax + by) + \mathrm{j}(bx - ay) \end{aligned}$$

Let

$$f = u + jv$$

with

$$u = ax + by$$

$$v = bx - ay$$

Hence,

$$\begin{aligned} \frac{\partial u}{\partial x} =& a \qquad \frac{\partial u}{\partial y} = b \\ \frac{\partial v}{\partial x} =& b \qquad \frac{\partial v}{\partial y} = -a \end{aligned}$$

From these results we immediately see that

$$\begin{aligned} \frac{\partial u}{\partial x} \neq& \frac{\partial v}{\partial y} \\ \frac{\partial v}{\partial x} =& -\frac{\partial u}{\partial y} \end{aligned}$$

In other words, the product term $w_k^* p(-k)$ does not satisfy the Cauchy-Riemann equations, and so this term is *not* analytic.

Problem 2.2

a)

From the Wiener-Hopf equation, we have

$$\mathbf{w}_0 = \mathbf{R}^{-1}\mathbf{p} \tag{1}$$

We are given that

$$\mathbf{R} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

$$\mathbf{p} = \begin{bmatrix} 0.5 \\ 0.25 \end{bmatrix}$$

Hence the inverse of $\mathbf{R}$ is

$$\begin{aligned} \mathbf{R}^{-1} =& \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}^{-1} \\ =& \frac{1}{0.75} \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}^{-1} \end{aligned}$$

Using Equation (1), we therefore get

$$\begin{aligned} \mathbf{w}_0 =& \frac{1}{0.75} \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.25 \end{bmatrix} \\ =& \frac{1}{0.75} \begin{bmatrix} 0.375 \\ 0 \end{bmatrix} \\ =& \begin{bmatrix} 0.5 \\ 0 \end{bmatrix} \end{aligned}$$

b)

The minimum mean-square error is

$$\begin{aligned} J_{\text{min}} =& \sigma_d^2 - \mathbf{p}^H \mathbf{w}_0 \\ =& \sigma_d^2 - \begin{bmatrix} 0.5 & 0.25 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0 \end{bmatrix} \\ =& \sigma_d^2 - 0.25 \end{aligned}$$

c)

The eigenvalues of the matrix $\mathbf{R}$ are roots of the characteristic equation:

$$(1-\lambda)^2 - (0.5)^2 = 0$$

That is, the two roots are

$$\lambda_1 = 0.5 \quad \text{and } \lambda_2 = 1.5$$

The associated eigenvectors are defined by

$$\mathbf{Rq} = \lambda\mathbf{q}$$

For $\lambda_1 = 0.5$, we have

$$\begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix} \begin{bmatrix} q_{11} \\ q_{12} \end{bmatrix} = 0.5 \begin{bmatrix} q_{11} \\ q_{12} \end{bmatrix}$$

Expanded this becomes

$$q_{11} + 0.5q_{12} = 0.5q_{11}$$

$$0.5q_{11} + q_{12} = 0.5q_{12}$$

Therefore,

$$q_{11} = -q_{12}$$

Normalizing the eigenvector $\mathbf{q}_1$ to unit length, we therefore have

$$\mathbf{q}_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Similarly, for the eigenvalue $\lambda_2 = 1.5$, we may show that

$$\mathbf{q}_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Accordingly, we may express the Wiener filter in terms of its eigenvalues and eigenvectors as follows:

$$\begin{aligned}
\mathbf{w}_0 &= \left(\sum_{i=1}^{2} \frac{1}{\lambda_i}\mathbf{q}_i\mathbf{q}_i^H\right)\mathbf{p} \\
&= \left(\frac{1}{\lambda_1}\mathbf{q}_1\mathbf{q}_1^H + \frac{1}{\lambda_2}\mathbf{q}_2\mathbf{q}_2^H\right)\mathbf{p} \\
&= \left(\begin{bmatrix}1\\-1\end{bmatrix}\begin{bmatrix}1 & -1\end{bmatrix} + \frac{1}{3}\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}1 & 1\end{bmatrix}\right)\begin{bmatrix}0.5\\0.25\end{bmatrix} \\
&= \left(\begin{bmatrix}1 & -1\\-1 & 1\end{bmatrix} + \frac{1}{3}\begin{bmatrix}1 & 1\\1 & 1\end{bmatrix}\right)\begin{bmatrix}0.5\\0.25\end{bmatrix} \\
&= \begin{bmatrix}\frac{4}{3} & -\frac{2}{3}\\-\frac{2}{3} & \frac{4}{3}\end{bmatrix}\begin{bmatrix}0.5\\0.25\end{bmatrix} \\
&= \begin{bmatrix}\frac{4}{6} - \frac{1}{6}\\-\frac{1}{3} + \frac{1}{3}\end{bmatrix} \\
&= \begin{bmatrix}0.5\\0\end{bmatrix}
\end{aligned}$$

Problem 2.3

a)

From the Wiener-Hopf equation we have

$$\mathbf{w}_0 = \mathbf{R}^{-1}\mathbf{p} \tag{1}$$

We are given

$$\mathbf{R} = \begin{bmatrix}1 & 0.5 & 0.25\\0.5 & 1 & 0.5\\0.25 & 0.5 & 1\end{bmatrix}$$

and

$$\mathbf{p} = \begin{bmatrix}0.5 & 0.25 & 0.125\end{bmatrix}^T$$

Hence, the use of these values in Equation (1) yields

$$\begin{aligned}\mathbf{w}_0 =& \mathbf{R}^{-1}\mathbf{p} \\ =& \begin{bmatrix} 1 & 0.5 & 0.25 \\ 0.5 & 1 & 0.5 \\ 0.25 & 0.5 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0.5 \\ 0.25 \\ 0.125 \end{bmatrix} \\ =& \begin{bmatrix} 1.33 & -0.67 & 0 \\ -0.67 & 1.67 & -0.67 \\ 0 & -0.67 & 1.33 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.25 \\ 0.125 \end{bmatrix}\end{aligned}$$

$$\mathbf{w}_0 = \begin{bmatrix} 0.5 & 0 & 0 \end{bmatrix}^T$$

b)

The Minimum mean-square error is

$$\begin{aligned}J_{\text{min}} =& \sigma_d^2 - \mathbf{p}^H \mathbf{w}_0 \\ =& \sigma_d^2 - \begin{bmatrix} 0.5 & 0.25 & 0.125 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0 \\ 0 \end{bmatrix} \\ =& \sigma_d^2 - 0.25\end{aligned}$$

c)

The eigenvalues of the matrix $\mathbf{R}$ are

$$\begin{bmatrix} \lambda_1 & \lambda_2 & \lambda_3 \end{bmatrix} = \begin{bmatrix} 0.4069 & 0.75 & 1.8431 \end{bmatrix}$$

The corresponding eigenvectors constitute the orthogonal matrix:

$$\mathbf{Q} = \begin{bmatrix} -0.4544 & -0.7071 & 0.5418 \\ 0.7662 & 0 & 0.6426 \\ -0.4544 & 0.7071 & 0.5418 \end{bmatrix}$$

Accordingly, we may express the Wiener filter in terms of its eigenvalues and eigenvectors as follows:

$$\mathbf{w}_0 = \left(\sum_{i=1}^{3} \frac{1}{\lambda_i} \mathbf{q}_i \mathbf{q}_i^H \right) \mathbf{p}$$

$$\mathbf{w}_0 = \left(\frac{1}{0.4069} \begin{bmatrix} -0.4544 \\ 0.7662 \\ -0.4544 \end{bmatrix} \begin{bmatrix} -0.4544 & 0.7662 & -0.4544 \end{bmatrix} + \frac{1}{0.75} \begin{bmatrix} -0.7071 \\ 0 \\ 0.7071 \end{bmatrix} \begin{bmatrix} -0.7071 & 0 & -0.7071 \end{bmatrix} + \frac{1}{1.8431} \begin{bmatrix} 0.5418 \\ 0.6426 \\ 0.5418 \end{bmatrix} \begin{bmatrix} 0.5418 & 0.6426 & 0.5418 \end{bmatrix} \right) \times \begin{bmatrix} 0.5 \\ 0.25 \\ 0.125 \end{bmatrix}$$

$$\mathbf{w}_0 = \left(\frac{1}{0.4069} \begin{bmatrix} 0.2065 & -0.3482 & 0.2065 \\ -0.3482 & 0.5871 & -0.3482 \\ 0.2065 & -0.3482 & 0.2065 \end{bmatrix} + \frac{1}{0.75} \begin{bmatrix} 0.5 & 0 & -0.5 \\ 0 & 0 & 0 \\ -0.5 & 0 & 0.5 \end{bmatrix} + \frac{1}{1.8431} \begin{bmatrix} 0.2935 & 0.3482 & 0.2935 \\ 0.3482 & 0.4129 & 0.3482 \\ 0.2935 & 0.3482 & 0.2935 \end{bmatrix} \right) \times \begin{bmatrix} 0.5 \\ 0.25 \\ 0.125 \end{bmatrix}$$

$$= \begin{bmatrix} 0.5 \\ 0 \\ 0 \end{bmatrix}$$

Problem 2.4

By definition, the correlation matrix

$$\mathbf{R} = \mathbb{E}[\mathbf{u}(n)\mathbf{u}^H(n)]$$

Where

$$\mathbf{u}(n) = \begin{bmatrix} u(n) \\ u(n-1) \\ \vdots \\ u(0) \end{bmatrix}$$

Invoking the ergodicity theorem,

$$\mathbf{R}(N) = \frac{1}{N+1} \sum_{n=0}^{N} \mathbf{u}(n)\mathbf{u}^H(n)$$

Likewise, we may compute the cross-correlation vector

$$\mathbf{p} = \mathbb{E}[\mathbf{u}(n)d^*(n)]$$

as the time average

$$\mathbf{p}(N) = \frac{1}{N+1}\sum_{n=0}^{N}\mathbf{u}(n)d^*(n)$$

The tap-weight vector of the wiener filter is thus defined by the matrix product

$$\mathbf{w}_0(N) = \left(\sum_{n=0}^{N}\mathbf{u}(n)\mathbf{u}^H(n)\right)^{-1}\left(\sum_{n=0}^{N}\mathbf{u}(n)d^*(n)\right)$$

Problem 2.5

a)

$$\begin{aligned}\mathbf{R} =& \mathbb{E}[\mathbf{u}(n)\mathbf{u}^H(n)]\\ =& \mathbb{E}[(\alpha(n)\mathbf{s}(n)+\mathbf{v}(n))(\alpha^*(n)\mathbf{s}^H(n)+\mathbf{v}^H(n))]\end{aligned}$$

With $\alpha(n)$ uncorrelated with $v(n)$, we have

$$\begin{aligned}\mathbf{R} =& \mathbb{E}[|\alpha(n)|^2]\mathbf{s}(n)\mathbf{s}^H(n) + \mathbb{E}[\mathbf{v}(n)\mathbf{v}^H(n)]\\ =& \sigma_\alpha^2\mathbf{s}(n)\mathbf{s}^H(n) + \mathbf{R}_v\end{aligned} \tag{1}$$

where $\mathbf{R}_v$ is the correlation matrix of $\mathbf{v}$

b)

The cross-correlation vector between the input vector $\mathbf{u}(n)$ and the desired response $d(n)$ is

$$\mathbf{p} = \mathbb{E}[\mathbf{u}(n)d^*(n)] \tag{2}$$

If $d(n)$ is uncorrelated with $\mathbf{u}(n)$, we have

$$\mathbf{p} = \mathbf{0}$$

Hence, the tap-weight of the wiener filter is

$$\begin{aligned}\mathbf{w}_0 =& \mathbf{R}^{-1}\mathbf{p}\\ =& \mathbf{0}\end{aligned}$$

c)

With $\sigma_\alpha^2 = 0$, Equation (1) reduces to

$$\mathbf{R} = \mathbf{R}_v$$

with the desired response

$$d(n) = v(n-k)$$

Equation (2) yields

$$\begin{aligned}
\mathbf{p} =& \mathbb{E}[(\alpha(n)\mathbf{s}(n) + \mathbf{v}(n)v^*(n-k))] \\
=& \mathbb{E}[(\mathbf{v}(n)v^*(n-k))] \\
=& \mathbb{E}\left[\begin{bmatrix} v(n) \\ v(n-1) \\ \vdots \\ v(n-M+1) \end{bmatrix} (v^*(n-k))\right] \\
=& \mathbb{E}\begin{bmatrix} r_v(n) \\ r_v(n-1) \\ \vdots \\ r_v(k-M+1) \end{bmatrix}, \quad 0 \leq k \leq M-1
\end{aligned} \tag{3}$$

where $r_v(k)$ is the autocorrelation of $v(n)$ for lag k. Accordingly, the tap-weight vector of the (optimum) wiener filter is

$$\begin{aligned}
\mathbf{w}_0 =& \mathbf{R}^{-1}\mathbf{p} \\
=& \mathbf{R}_v^{-1}\mathbf{p}
\end{aligned}$$

where $\mathbf{p}$ is defined in Equation (3).

d)

For a desired response

$$d(n) = \alpha(n)\exp(-\mathrm{j}\,\omega\tau)$$

The cross-correlation vector $\mathbf{p}$ is

$$\begin{aligned}
\mathbf{p} =& \mathbb{E}[\mathbf{u}(n)(d^*n)] \\
=& \mathbb{E}[(\alpha(n)\mathbf{s}(n) + \mathbf{v}(n))\,\alpha^*(n)\exp(-\mathrm{j}\,\omega\tau)] \\
=& \mathbf{s}(n)\exp(\mathrm{j}\,\omega\tau)\mathbb{E}[|\alpha(n)|^2] \\
=& \sigma_\alpha^2\mathbf{s}(n)\exp(\mathrm{j}\,\omega\tau) \\
=& \sigma_\alpha^2 \begin{bmatrix} 1 \\ \exp(-\mathrm{j}\,\omega) \\ \vdots \\ \exp((-\mathrm{j}\,\omega)(M-1)) \end{bmatrix} \exp(\mathrm{j}\,\omega\tau) \\
=& \sigma_\alpha^2 \begin{bmatrix} \exp(\mathrm{j}\,\omega\tau) \\ \exp(\mathrm{j}\,\omega(\tau-1)) \\ \vdots \\ \exp((\mathrm{j}\,\omega)(\tau-M+1)) \end{bmatrix}
\end{aligned}$$

The corresponding value of the tap-weight vector of the Wiener filter is

$$\begin{aligned}
\mathbf{w}_0 =& \sigma_\alpha^2(\sigma_\alpha^2\mathbf{s}(n)\mathbf{s}^H(n) + \mathbf{R}_v)^{-1} \begin{bmatrix} \exp(\mathrm{j}\,\omega\tau) \\ \exp(\mathrm{j}\,\omega(\tau-1)) \\ \vdots \\ \exp((\mathrm{j}\,\omega)(\tau-M+1)) \end{bmatrix} \\
=& \left(\mathbf{s}(n)\mathbf{s}^H(n) + \frac{1}{\sigma_\alpha^2}\mathbf{R}_v\right)^{-1} \begin{bmatrix} \exp(\mathrm{j}\,\omega\tau) \\ \exp(\mathrm{j}\,\omega(\tau-1)) \\ \vdots \\ \exp((\mathrm{j}\,\omega)(\tau-M+1)) \end{bmatrix}
\end{aligned}$$

Problem 2.6

The optimum filtering solution is defined by the Wiener-Hopf equation

$$\mathbf{R}\mathbf{w}_0 = \mathbf{p} \tag{1}$$

for which the minimum mean-square error is

$$J_{\text{min}} = \sigma_d^2 - \mathbf{p}^H\mathbf{w}_0 \tag{2}$$

Combine Equations (1) and Equation(2) into a single relation:

$$\begin{bmatrix} \sigma_d^2 & \mathbf{p}^H \\ \mathbf{p} & \mathbf{R} \end{bmatrix} \begin{bmatrix} 1 \\ \mathbf{w}_0 \end{bmatrix} = \begin{bmatrix} J_{\text{min}} \\ \mathbf{0} \end{bmatrix}$$

Define

$$\mathbf{A} = \begin{bmatrix} \sigma_d^2 & \mathbf{p}^H \\ \mathbf{p} & \mathbf{R} \end{bmatrix} \tag{3}$$

Since

$$\sigma_d^2 = \mathbb{E}[d(n)d^*(n)]$$

$$\mathbf{p} = \mathbb{E}[\mathbf{u}(n)d^*(n)]$$

$$\mathbf{R} = \mathbb{E}[\mathbf{u}(n)\mathbf{u}^H(n)]$$

we may rewrite Equation (3) as

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} \mathbb{E}[d(n)d^*(n)] & \mathbb{E}[d(n)\mathbf{u}^H(n)] \\ \mathbb{E}[\mathbf{u}(n)d^*(n)] & \mathbb{E}[\mathbf{u}(n)\mathbf{u}^H(n)] \end{bmatrix} \\ &= \mathbb{E}\left\{ \begin{bmatrix} d(n) \\ \mathbf{u}(n) \end{bmatrix} \begin{bmatrix} d^*(n) & \mathbf{u}^H(n) \end{bmatrix} \right\} \end{aligned}$$

The minimum mean-square error equals

$$J_{\text{min}} = \sigma_d^2 - \mathbf{p}^H \mathbf{w}_0 \tag{4}$$

Eliminating σ_d^2 between Equation (1) and Equation (4):

$$J(\mathbf{w}) = J_{\text{min}} + \mathbf{p}^H \mathbf{w}_0 - \mathbf{p}^H \mathbf{R}\mathbf{w} - \mathbf{w}^H \mathbf{R}\mathbf{w}_0 + \mathbf{w}^H \mathbf{R}\mathbf{w} \tag{5}$$

Eliminating $\mathbf{p}$ between Equation (2) and Equation (5)

$$J(\mathbf{w}) = J_{\text{min}} + \mathbf{w}_0^H \mathbf{R}\mathbf{w}_0 - \mathbf{w}_0^H \mathbf{R}\mathbf{w} - \mathbf{w}^H \mathbf{R}\mathbf{w}_0 + \mathbf{w}^H \mathbf{R}\mathbf{w} \tag{6}$$

where we have used the property $\mathbf{R}^H = \mathbf{R}$. We may rewrite Equation (6) as

$$J(\mathbf{w}) = J_{\mathbf{min}} + (\mathbf{w} - \mathbf{w}_0)^H \mathbf{R}(\mathbf{w} - \mathbf{w}_0)$$

which clearly shows that $J(\mathbf{w}_0) = J_{\mathbf{min}}$

Problem 2.7

The minimum mean-square error is

$$J_{\text{min}} = \sigma_d^2 - \mathbf{p}^H \mathbf{R}^{-1} \mathbf{p} \tag{1}$$

Using the spectral theorem, we may express the correlation matrix $\mathbf{R}$ as

$$\mathbf{R} = \mathbf{Q}\Lambda\mathbf{Q}^H$$

$$\mathbf{R} = \sum_{k=1}^{M} \lambda_k \mathbf{q}_k \mathbf{q}_k^H \tag{2}$$

Substituting Equation (2) into Equation (1)

$$\begin{aligned} J_{\text{min}} =& \sigma_d^2 - \sum_{k=1}^{M} \frac{1}{\lambda_k} \mathbf{p}^H \mathbf{q}_k \mathbf{p}^H \mathbf{q}_k \\ =& \sigma_d^2 - \sum_{k=1}^{M} \frac{1}{\lambda_k} |\mathbf{p}^H \mathbf{q}_k|^2 \end{aligned}$$

Problem 2.8

When the length of the Wiener filter is greater than the model order m, the tail end of the tap-weight vector of the Wiener filter is zero; thus,

$$\mathbf{w}_0 = \begin{bmatrix} \mathbf{a}_m \\ \mathbf{0} \end{bmatrix}$$

Therefore, the only possible solution for the case of an over-fitted model is

$$\mathbf{w}_0 = \begin{bmatrix} \mathbf{a}_m \\ \mathbf{0} \end{bmatrix}$$

Problem 2.9

a)

The Wiener solution is defined by

$$\mathbf{R}_M \mathbf{a}_M = \mathbf{p}_M$$

$$\begin{bmatrix} \mathbf{R}_M & \mathbf{r}_{M-m} \\ \mathbf{r}_{M-m}^H & \mathbf{R}_{M-m,M-m} \end{bmatrix} \begin{bmatrix} \mathbf{a}_m \\ \mathbf{0}_{M-m} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_m \\ \mathbf{p}_{M-m} \end{bmatrix}$$

$$\mathbf{R}_M \mathbf{a}_m = \mathbf{p}_m$$

$$\mathbf{r}_{M-m}^H \mathbf{a}_m = \mathbf{p}_{M-m}$$

$$\mathbf{p}_{M-m} = \mathbf{r}_{M-m}^H \mathbf{a}_m = \mathbf{r}_{M-m}^H \mathbf{R}_M^{-1} \mathbf{p}_m \tag{1}$$

b)

Applying the conditions of Equation (1) to the example in Section 2.7 in the textbook

$$\mathbf{r}_{M-m}^H = \begin{bmatrix} -0.05 & 0.1 & 0.15 \end{bmatrix}$$

$$\mathbf{a}_m = \begin{bmatrix} 0.8719 \\ -0.9129 \\ 0.2444 \end{bmatrix}$$

The last entry in the 4-by-1 vector $\mathbf{p}$ is therefore

$$\begin{aligned} \mathbf{r}_{M-m}^H \mathbf{a}_m &= -0.0436 - 0.0912 + 0.1222 \\ &= -0.0126 \end{aligned}$$

Problem 2.10

$$\begin{aligned} J_{\text{min}} &= \sigma_d^2 - \mathbf{p}^H \mathbf{w}_0 \\ &= \sigma_d^2 - \mathbf{p}^H \mathbf{R}^{-1} \mathbf{p} \end{aligned}$$

when $m = 0$,

$$\begin{aligned} J_{\text{min}} &= \sigma_d^2 \\ &= 1.0 \end{aligned}$$

When $m = 1$,

$$\begin{aligned} J_{\text{min}} &= 1 - 0.5 \times \frac{1}{1.1} \times 0.5 \\ &= 0.9773 \end{aligned}$$

when $m = 2$

$$\begin{aligned} J_{\text{min}} &= 1 - \begin{bmatrix} 0.5 & -0.4 \end{bmatrix} \begin{bmatrix} 1.1 & 0.5 \\ 0.5 & 1.1 \end{bmatrix}^{-1} \begin{bmatrix} 0.5 \\ -0.4 \end{bmatrix} \\ &= 1 - 0.6781 \\ &= 0.3219 \end{aligned}$$

when $m = 3$,

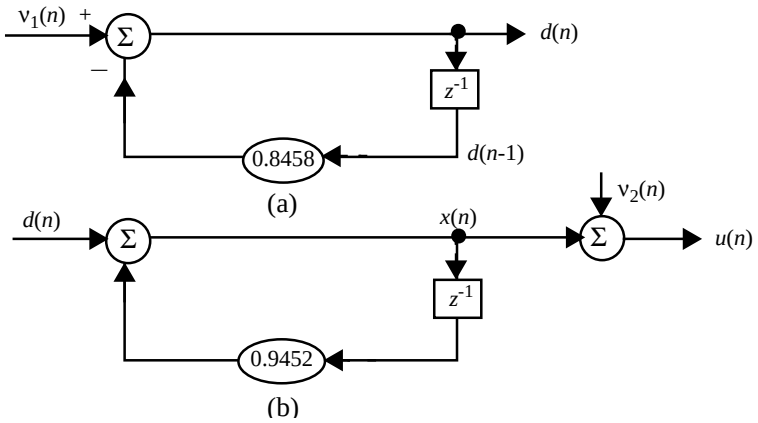
$$\begin{aligned} J_{\text{min}} &= 1 - \begin{bmatrix} 0.5 & -0.4 & -0.2 \end{bmatrix} \begin{bmatrix} 1.1 & 0.5 & 0.1 \\ 0.5 & 1.1 & 0.5 \\ 0.1 & 0.5 & 1.1 \end{bmatrix}^{-1} \begin{bmatrix} 0.5 \\ -0.4 \\ -0.2 \end{bmatrix} \\ &= 1 - 0.6859 \\ &= 0.3141 \end{aligned}$$

when $m = 4$,

$$\begin{aligned} J_{\text{min}} &= 1 - 0.6859 \\ &= 0.3141 \end{aligned}$$

Thus any further increase in the filter order beyond $m = 3$ does not produce any meaningful reduction in the minimum mean-square error.

Problem 2.11



a)

$$u(n) = x(n) + v_2(n) \tag{1}$$

$$d(n) = -d(n-1) \times 0.8458 + v_1(n) \tag{2}$$

$$x(n) = d(n) + 0.9458x(n-1) \tag{3}$$

Equation (3) rearranged to solve for $d(n)$ is

$$d(n) = x(n) - 0.9458x(n-1)$$

Using Equation (2) and Equation (3):

$$x(n) - 0.9458x(n-1) = 0.8458[-x(n-1) + 0.9458x(n-2)] + v_1(n)$$

Rearranging the terms this produces:

$$\begin{aligned} x(n) =& (0.9458 - 8.8458)x(n-1) + 0.8x(n-2) + v_1(n) \\ =& (0.1)x(n-1) + 0.8x(n-2) + v_1(n) \end{aligned}$$

b)

$$u(n) = x(n) + v_2(n)$$

where $x(n)$ and $v_2(n)$ are uncorrelated, therefore

$$\mathbf{R} = \mathbf{R}_x + \mathbf{R}_{v_2}$$

$$\mathbf{R}_x = \begin{bmatrix} r_x(0) & r_x(1) \\ r_x(1) & r_x(0) \end{bmatrix}$$

$$\begin{aligned} r_x(0) =& \sigma_x^2 \\ =& \frac{1+a_2}{1-a_2} \frac{\sigma_1^2}{(1+a_2)^2 - a_1^2} = 1 \end{aligned}$$

$$r_x(1) = \frac{-a_1}{1+a_2}$$

$$r_x(1) = 0.5$$

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

$$\mathbf{R}_{v_2} = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$$

$$\mathbf{R} = \mathbf{R}_x + \mathbf{R}_{v_2} = \begin{bmatrix} 1.1 & 0.5 \\ 0.5 & 1.1 \end{bmatrix}$$

$$\mathbf{p} = \begin{bmatrix} p(0) \\ p(1) \end{bmatrix}$$

$$p(k) = \mathbb{E}[u(n-k)d(n)], \quad k = 0, 1$$

$$\begin{aligned} p(0) =& r_x(0) + b_1 r_x(-1) \\ =& 1 - 0.9458 \times 0.5 \\ =& 0.5272 \end{aligned}$$

$$\begin{aligned} p(1) =& r_x(1) + b_1 r_x(0 \\ =& 0.5 - 0.9458 \\ =& -0.4458 \end{aligned}$$

Therefore,

$$\mathbf{p} = \begin{bmatrix} 0.5272 \\ -0.4458 \end{bmatrix}$$

c)

The optimal weight vector is given by the equation $\mathbf{w}_0 = \mathbf{R}^{-1}\mathbf{p}$; hence,

$$\begin{aligned} \mathbf{w}_0 &= \begin{bmatrix} 1.1 & 0.5 \\ 0.5 & 1.1 \end{bmatrix}^{-1} \begin{bmatrix} 0.5272 \\ -0.4458 \end{bmatrix} \\ &= \begin{bmatrix} 0.8363 \\ -0.7853 \end{bmatrix} \end{aligned}$$

Problem 2.12

a)

For $M = 3$ taps, the correlation matrix of the tap inputs is

$$\mathbf{R} = \begin{bmatrix} 1.1 & 0.5 & 0.85 \\ 0.5 & 1.1 & 0.5 \\ 0.85 & 0.5 & 1.1 \end{bmatrix}$$

The cross-correlation vector between the tap inputs and the desired response is

$$\mathbf{p} = \begin{bmatrix} 0.527 \\ -0.446 \\ 0.377 \end{bmatrix}$$

b)

The inverse of the correlation matrix is

$$\mathbf{R}^{-1} = \begin{bmatrix} 2.234 & -0.304 & -1.666 \\ -0.304 & 1.186 & -0.304 \\ -1.66 & -0.304 & 2.234 \end{bmatrix}$$

Hence, the optimum weight vector is

$$\mathbf{w}_0 = \mathbf{R}^{-1}\mathbf{p} = \begin{bmatrix} 0.738 \\ -0.803 \\ 0.138 \end{bmatrix}$$

The minimum mean-square error is

$$J_{\text{min}} = 0.15$$

Problem 2.13

a)

The correlation matrix $\mathbf{R}$ is

$$\begin{aligned}
\mathbf{R} =& \mathbb{E}[\mathbf{u}(n)\mathbf{u}^H(n)] \\
=& \mathbb{E}[|A_1|^2] \begin{bmatrix} e^{-\mathrm{j}\omega_1 n} \\ e^{-\mathrm{j}\omega_1(n-1)} \\ \vdots \\ e^{-\mathrm{j}\omega_1(n-M+1)} \end{bmatrix} \begin{bmatrix} e^{+\mathrm{j}\omega_1 n} & e^{+\mathrm{j}\omega_1(n-1)} & \dots & e^{+\mathrm{j}\omega_1(n-M+1)} \end{bmatrix} \\
=& \mathbb{E}[|A_1|^2]\mathbf{s}(\omega_1)\mathbf{s}^H(\omega_1) + \mathbf{I}\mathbb{E}[|v(n)|^2] \\
=& \sigma_1^2\mathbf{s}(\omega_1)\mathbf{s}^H(\omega_1) + \sigma_v^2\mathbf{I}
\end{aligned}$$

where $\mathbf{I}$ is the identity matrix.

b)

The tap-weights vector of the Wiener filter is

$$\mathbf{w}_0 = \mathbf{R}^{-1}\mathbf{p}$$

From part **a)**,

$$\mathbf{R} = \sigma_1^2\mathbf{s}(\omega_1)\mathbf{s}^H(\omega_1) + \sigma_v^2\mathbf{I}$$

We are given

$$\mathbf{p} = \sigma_0^2\mathbf{s}(\omega_0)$$

To invert the matrix $\mathbf{R}$, we use the matrix inversion lemma (see Chapter 10), as described here:

If:

$$\mathbf{A} = \mathbf{B}^{-1} + \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^H$$

then:

$$\mathbf{A}^{-1} = \mathbf{B} - \mathbf{B}\mathbf{C}(\mathbf{D} + \mathbf{C}^H\mathbf{B}\mathbf{C})^{-1}\mathbf{C}^H\mathbf{B}$$

In our case:

$$\mathbf{A} = \sigma_v^2\mathbf{I}$$

$$\mathbf{B}^{-1} = \sigma_v^2 \mathbf{I}$$
$$\mathbf{D}^{-1} = \sigma_1^2$$
$$\mathbf{C} = \mathbf{s}(\omega_1)$$

Hence,

$$\mathbf{R}^{-1} = \frac{1}{\sigma_v^2}\mathbf{I} - \frac{\frac{1}{\sigma_v^2}\mathbf{s}(\omega_1)\mathbf{s}^H(\omega_1)}{\frac{\sigma_v^2}{\sigma_1^2} + \mathbf{s}^H(\omega_1)\mathbf{s}(\omega_1)}$$

The corresponding value of the Wiener tap-weight vector is

$$\mathbf{w}_0 = \mathbf{R}^{-1}\mathbf{p}$$

$$\mathbf{w}_0 = \frac{\sigma_0^2}{\sigma_v^2}\mathbf{s}(\omega_0) - \frac{\frac{\sigma_0^2}{\sigma_v^2}\mathbf{s}(\omega_1)\mathbf{s}^H(\omega_1)}{\frac{\sigma_v^2}{\sigma_1^2} + \mathbf{s}^H(\omega_1)\mathbf{s}(\omega_1)}\mathbf{s}(\omega_0)$$

we note that

$$\mathbf{s}^H(\omega_1)\mathbf{s}(\omega_1) = M$$

which is a scalar hence,

$$\mathbf{w}_0 = \frac{\sigma_0^2}{\sigma_v^2}\mathbf{s}(\omega_0) - \left(\frac{\sigma_0^2}{\sigma_v^2}\frac{\mathbf{s}^H(\omega_1)\mathbf{s}(\omega_1)}{\frac{\sigma_v^2}{\sigma_0^2} + M}\mathbf{s}(\omega_0)\right)$$

Problem 2.14

The output of the array processor equals

$$e(n) = u(1,n) - wu(2,n)$$

The mean-square error equals

$$\begin{aligned} J(w) =& \mathbb{E}[|e(n)|^2] \\ =& \mathbb{E}[(u(1,n) - wu(2,n))(u^*(1,n) - w^*u^*(2,n))] \\ =& \mathbb{E}[|u(1,n)|^2] + |w|^2\mathbb{E}[|u(2,n)|^2] - w\mathbb{E}[u(2,n)u^*(1,n)] - w\mathbb{E}[u(1,n)u^*(2,n)] \end{aligned}$$

Differentiating $J(w)$ with respect to w:

$$\frac{\partial J}{\partial w} = -2\mathbb{E}[u(1,n)u^*(2,n)] + 2w\mathbb{E}[|u(2,n)|^2]$$

Putting $\frac{\partial J}{\partial w} = 0$ and solving for the optimum value of w:

$$w_0 = \frac{\mathbb{E}[u(1,n)u^*(2,n)]}{\mathbb{E}[|u(2,n)|^2]}$$

Problem 2.15

Define the index of the performance (i.e., cost function)

$$J(\mathbf{w}) = \mathbb{E}[|e(n)|^2] + \mathbf{c}^H\mathbf{s}^H\mathbf{w} + \mathbf{w}^H\mathbf{s}\mathbf{c} - 2\mathbf{c}^H\mathbf{D}^{1/2}\mathbf{1}$$

$$J(\mathbf{w}) = \mathbf{w}^H\mathbf{R}\mathbf{w} + \mathbf{c}^H\mathbf{s}^H\mathbf{w} + \mathbf{w}^H\mathbf{s}\mathbf{c} - 2\mathbf{c}^H\mathbf{D}^{1/2}\mathbf{1}$$

Differentiate $J(\mathbf{w})$ with respect to $\mathbf{w}$ and set the result equal to zero:

$$\frac{\partial J}{\partial \mathbf{w}} = 2\mathbf{R}\mathbf{w} + 2\mathbf{s}\mathbf{c} = \mathbf{0}$$

Hence,

$$\mathbf{w}_0 = -\mathbf{R}^{-1}\mathbf{s}\mathbf{c}$$

But, we must constrain $\mathbf{w}_0$ as

$$\mathbf{s}^H\mathbf{w}_0 = \mathbf{D}^{1/2}\mathbf{1}$$

therefore, the vector $\mathbf{c}$ equals

$$\mathbf{c} = -(\mathbf{s}^H\mathbf{R}^{-1}\mathbf{s})^{-1}\mathbf{D}^{1/2}\mathbf{1}$$

Correspondingly, the optimum weight vector equals

$$\mathbf{w}_0 = \mathbf{R}^{-1}\mathbf{s}(\mathbf{s}^H\mathbf{R}^{-1}\mathbf{s})^{-1}\mathbf{D}^{1/2}\mathbf{1}$$

Problem 2.16

The weight vector $\mathbf{w}$ of the beamformer that maximizes the output signal-to-noise ratio:

$$(\text{SNR})_0 = \frac{\mathbf{w}^H \mathbf{R}_S \mathbf{w}}{\mathbf{w}^H \mathbf{R}_v \mathbf{w}}$$

is derived in part **b)** of the problem 2.18; there it is shown that the optimum weight vector $\mathbf{w}_{SN}$ so defined is given by

$$\mathbf{w}_{SN} = \mathbf{R}_v^{-1} \mathbf{s} \tag{1}$$

where $\mathbf{s}$ is the signal component and $\mathbf{R}_v$ is the correlation matrix of the noise $\mathbf{v}(n)$. On the other hand, the optimum weight vector of the LCMV beamformer is defined by

$$\mathbf{w}_0 = g^* \frac{\mathbf{R}^{-1} s(\phi)}{s^H(\phi) \mathbf{R}^{-1} s(\phi)} \tag{2}$$

where $s(\phi)$ is the steering vector. In general, the formulas (1) and (2) yield different values for the weight vector of the beamformer.

Problem 2.17

Let τ_i be the propagation delay, measured from the zero-time reference to the ith element of a nonuniformly spaced array, for a plane wave arriving from a direction defined by angle θ with respect to the perpendicular to the array. For a signal of angular frequency ω, this delay amounts to a phase shift equal to $-\omega\tau_i$. Let the phase shifts for all elements of the array be collected together in a column vector denoted by $\mathbf{d}(\omega, \theta)$. The response of a beamformer with weight vector $\mathbf{w}$ to a signal (with angular frequency ω) originates from angle $\theta = \mathbf{w}^H \mathbf{d}(\omega, \theta)$. Hence, constraining the response of the array at ω and θ to some value g involves the linear constraint

$$\mathbf{w}^H \mathbf{d}(\omega, \theta) = g$$

Thus, the constraint vector $\mathbf{d}(\omega, \theta)$ serves the purpose of generalizing the idea of an LCMV beamformer beyond simply the case of a uniformly spaced array. Everything else is the same as before, except for the fact that the correlation matrix of the received signal is no longer Toeplitz for the case of a nonuniformly spaced array

Problem 2.18

a)

Under hypothesis H_1, we have

$$\mathbf{u} = \mathbf{s} + \mathbf{v}$$

The correlation matrix of $\mathbf{u}$ equals

$$\mathbf{R} = \mathbb{E}[\mathbf{u}\mathbf{u}^T]$$

$$\mathbf{R} = \mathbf{s}\mathbf{s}^T + \mathbf{R}_N, \quad \text{where } \mathbf{R}_N = \mathbb{E}[\mathbf{v}\mathbf{v}^T]$$

The tap-weight vector $\mathbf{w}_k$ is chosen so that $\mathbf{w}_k^T\mathbf{u}$ yields an optimum estimate of the kth element of $\mathbf{s}$. Thus, with $s(k)$ treated as the desired response, the cross-correlation vector between $\mathbf{u}$ and $s(k)$ equals

$$\begin{aligned}\mathbf{p}_k =& \mathbb{E}[\mathbf{u}s(k)] \\ =& \mathbf{s}\mathbf{s}(k), \quad k = 1, 2, \ldots, m\end{aligned}$$

Hence, the Wiener-Hopf equation yields the optimum value of $\mathbf{w}_k$ as

$$\mathbf{w}_{k0} = \mathbf{R}^{-1}\mathbf{p}_k$$

$$\mathbf{w}_{k0} = (\mathbf{s}\mathbf{s}^T + \mathbf{R}_N)^{-1}\mathbf{s}s(k), \quad k = 1, 2, \ldots, M \tag{1}$$

To apply the matrix inversion lemma (introduced in Problem 2.13), we let

$$\mathbf{A} = \mathbf{R}$$

$$\mathbf{B}^{-1} = \mathbf{R}_N$$

$$\mathbf{C} = \mathbf{s}$$

$$\mathbf{D} = 1$$

Hence,

$$\mathbf{R}^{-1} = \mathbf{R}_N^{-1} - \frac{\mathbf{R}_N^{-1}\mathbf{s}\mathbf{s}^T\mathbf{R}_N^{-1}}{1 + \mathbf{s}^T\mathbf{R}_N^{-1}\mathbf{s}} \tag{2}$$

Substituting Equation (2) into Equation (1) yields:

$$\mathbf{w}_{k0} = \left(\mathbf{R}_N^{-1} - \frac{\mathbf{R}_N^{-1}\mathbf{s}\mathbf{s}^T\mathbf{R}_N^{-1}}{1 + \mathbf{s}^T\mathbf{R}_N^{-1}\mathbf{s}}\right)\mathbf{s}s(k)$$

$$\mathbf{w}_{k0} = \frac{\mathbf{R}_N^{-1}\mathbf{s}(1 + \mathbf{s}^T\mathbf{R}_N^{-1}\mathbf{s}) - \mathbf{R}_N^{-1}\mathbf{s}\mathbf{s}^T\mathbf{R}_N^{-1}\mathbf{s}}{1 + \mathbf{s}^T\mathbf{R}_N^{-1}\mathbf{s}} s(k)$$

$$\mathbf{w}_{k0} = \frac{s(k)}{1 + \mathbf{s}^T\mathbf{R}_N^{-1}\mathbf{s}}\mathbf{R}_N^{-1}\mathbf{s}$$

b)

The output signal-to-noise ratio is

$$\begin{aligned}\text{SNR} =& \frac{\mathbb{E}[(\mathbf{w}^T\mathbf{s})^2]}{\mathbb{E}[(\mathbf{w}^T\mathbf{v})^2]} \\ =& \frac{\mathbf{w}^T\mathbf{s}\mathbf{s}^T\mathbf{w}}{\mathbf{w}^T\mathbb{E}[\mathbf{v}\mathbf{v}^T]\mathbf{w}} \\ =& \frac{\mathbf{w}^T\mathbf{s}\mathbf{s}^T\mathbf{w}}{\mathbf{w}^T\mathbf{R}_N\mathbf{w}}\end{aligned} \tag{3}$$

Since $\mathbf{R}_N$ is positive definite, we may write,

$$\mathbf{R}_N = \mathbf{R}_N^{1/2}\mathbf{R}_N^{1/2}$$

Define the vector

$$\mathbf{a} = \mathbf{R}_N^{1/2}\mathbf{w}$$

or equivalently,

$$\mathbf{w} = \mathbf{R}_N^{-1/2}\mathbf{a} \tag{4}$$

Accordingly, we may rewrite Equation (3) as follows

$$\text{SNR} = \frac{\mathbf{a}^T\mathbf{R}_N^{1/2}\mathbf{s}\mathbf{s}^T\mathbf{R}_N^{1/2}\mathbf{a}}{\mathbf{a}^T\mathbf{a}} \tag{5}$$

where we have used the symmetric property of $\mathbf{R}_N$. Define the normalized vector

$$\bar{\mathbf{a}} = \frac{\mathbf{a}}{||\mathbf{a}||}$$

where $||\mathbf{a}||$ is the norm of $\mathbf{a}$. Equation (5) may be rewritten as:

$$\text{SNR} = \bar{\mathbf{a}}^T\mathbf{R}_N^{1/2}\mathbf{s}\mathbf{s}^T\mathbf{R}_N^{1/2}\bar{\mathbf{a}}$$

$$\text{SNR} = \left|\bar{\mathbf{a}}^T \mathbf{R}_N^{1/2} \mathbf{s}\right|^2$$

Thus the output signal-to-noise ratio SNR equals the squared magnitude of the inner product of the two vectors $\bar{\mathbf{a}}$ and $\mathbf{R}_N^{1/2}\mathbf{s}$. This inner product is maximized when $\mathbf{a}$ equals $\mathbf{R}_N^{-1/2}$. That is,

$$\mathbf{a}_{SN} = \mathbf{R}_N^{-1/2}\mathbf{s} \tag{6}$$

Let $\mathbf{w}_{SN}$ denote the value of the tap-weight vector that corresponds to Equation (6). Hence, the use of Equation (4) in Equation (6) yields

$$\mathbf{w}_{SN} = \mathbf{R}_N^{-1/2}(\mathbf{R}_N^{-1/2}\mathbf{s})$$

$$\mathbf{w}_{SN} = \mathbf{R}_N^{-1}\mathbf{s}$$

c)

Since the noise vector $\mathbf{v}(n)$ is Gaussian, its joint probability density function equals

$$f_{\mathbf{v}}(\mathbf{v}) = \frac{1}{(2\pi)^{M/2}(\det(\mathbf{R}_N))^{1/2}} \exp\left(-\frac{1}{2}\mathbf{v}^T\mathbf{R}_N^{-1}\mathbf{v}\right)$$

Under the hypothesis H_0 we have

$$\mathbf{u} = \mathbf{v}$$

and

$$f_{\mathbf{u}}(\mathbf{u}|H_0) = \frac{1}{(2\pi)^{M/2}(\det\mathbf{R}_N)^{1/2}} \exp\left(-\frac{1}{2}\mathbf{u}^T\mathbf{R}_N^{-1}\mathbf{u}\right)$$

Under hypothesis H_1 we have

$$\mathbf{u} = \mathbf{s} + \mathbf{v}$$

and

$$f_{\mathbf{u}}(\mathbf{u}|H_1) = \frac{1}{(2\pi)^{M/2}(\det\mathbf{R}_N)^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{u}-\mathbf{s})^T\mathbf{R}_N^{-1}(\mathbf{u}-\mathbf{s})\right)$$

Hence, the likelihood ratio is defined by

$$\begin{aligned}\Lambda =& \frac{f_{\mathbf{u}}(\mathbf{u}|H_1)}{f_{\mathbf{u}}(\mathbf{u}|H_0)} \\ =& \exp\left(-\frac{1}{2}\mathbf{s}^T\mathbf{R}_N^{-1}\mathbf{s} + \mathbf{s}^T\mathbf{R}_N^{-1}\mathbf{u}\right)\end{aligned}$$

The natural logarithm of the likelihood ratio equals

$$\ln \Lambda = -\frac{1}{2}\mathbf{s}^T\mathbf{R}_N^{-1}\mathbf{s} + \mathbf{s}^T\mathbf{R}_N^{-1}\mathbf{u} \tag{7}$$

The first term in (7) represents a constant. Hence, testing $\ln \Lambda$ against a threshold is equivalent to the test

$$\mathbf{s}^T\mathbf{R}_N^{-1}\mathbf{u} \underset{H_0}{\overset{H_1}{\gtrless}} \lambda$$

where λ is some threshold. Equivalently, we may write

$$\mathbf{w}_{ML} = \mathbf{R}_N^{-1}\mathbf{s}$$

where $\mathbf{w}_{ML}$ is the maximum likelihood weight vector.

The results of parts **a)**, **b)**, and **c)** show that the three criteria discussed here yield the same optimum value for the weight vector, except for a scaling factor.

Problem 2.19

a)

Assuming the use of a noncausal Wiener filter, we write

$$\sum_{i=-\infty}^{\infty} w_{0i}r(i-k) = p(-k), \quad k = 0, \pm 1, \pm 2, \ldots, \pm\infty \tag{1}$$

where the sum now extends from $i = -\infty$ to $i = \infty$. Define the z-transforms:

$$S(z) = \sum_{k=-\infty}^{\infty} r(k)z^{-k}, \qquad H_u(z) = \sum_{k=-\infty}^{\infty} w_{0,k}z^{-k}$$

$$P(z) = \sum_{k=-\infty}^{\infty} p(-k)z^{-k} = P(z^{-1})$$

Hence, applying the z-transform to Equation (1):

$$H_u(z)S(z) = P(z^{-1})$$

$$H_u(z) = \frac{P(1/z)}{S(z)} \tag{2}$$

b)

$$P(z) = \frac{0.36}{\left(1 - \dfrac{0.2}{z}\right)(1 - 0.2z)}$$

$$P(1/z) = \frac{0.36}{(1 - 0.2z)\left(1 - \dfrac{0.2}{z}\right)}$$

$$S(z) = 1.37\frac{(1 - 0.146z^{-1})(1 - 0.146z)}{(1 - 0.2z^{-1})(1 - 0.2z)}$$

Thus, applying Equation (2) yields

$$\begin{aligned} H_u(z) &= \frac{0.36}{1.37(1 - 0.146z^{-1})(1 - 0.146z)} \\ &= \frac{0.36z^{-1}}{1.37(1 - 0.146z^{-1})(z^{-1} - 0.146)} \\ &= \frac{0.2685}{1 - 0.146z^{-1}} + \frac{0.0392}{z^{-1} - 0.146} \end{aligned}$$

Clearly, this system is noncausal. Its impulse response is $h(n) =$ inverse z-transform of $H_u(z)$ is given by

$$h(n) = 0.2685(0.146)^n u_{\text{step}}(n) - \frac{0.0392}{0.146}\left(\frac{1}{0.146}\right)^n u_{\text{step}}(-n)$$

where $u_{\text{step}}(n)$ is the unit-step function:

$$u_{\text{step}}(n) = \begin{cases} 1 \text{ for } n = 0, 1, 2, \ldots \\ 0 \text{ for } n = -1, -2, \ldots \end{cases}$$

and $u_{\text{step}}(-n)$ is its mirror image:

$$u_{\text{step}}(-n) = \begin{cases} 1 \text{ for } n = 0, -1, -2, \ldots \\ 0 \text{ for } n = 1, 2, \ldots \end{cases}$$

Simplifying,

$$h_u(n) = 0.2685 \times (0.146)^n u_{\text{step}}(n) - 0.2685 \times (6.849)^{-n} u_{\text{step}}(-n)$$

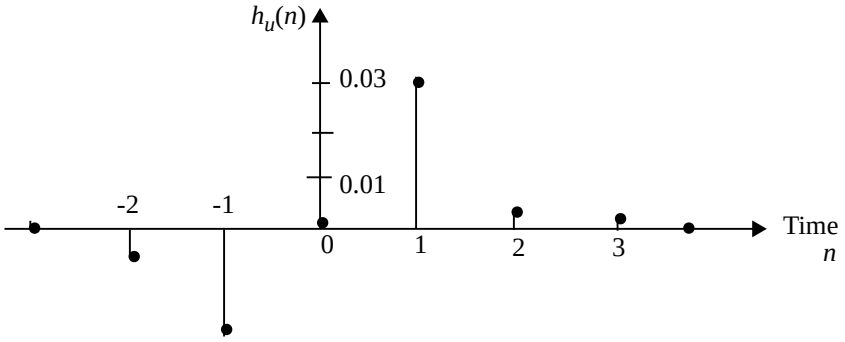
Evaluating $h_u(n)$ for varying n:

$$h_u(0) = 0$$

$$h_u(1) = 0.03, \qquad h_u(2) = 0.005, \qquad h_u(3) = 0.0008$$

$$h_u(-1) = -0.03, \quad h_u(-2) = -0.005, \quad h_u(-3) = -0.0008$$

The preceding values for $h_u(n)$ are plotted in the following figure:



c)

A delay of 3 time units applied to the impulse response will make the system causal and therefore realizable.

Chapter 3

Problem 3.1

a)

Let $\mathbf{a}_M$ denote the tap-weight vector of the forward prediction-error filter. With a tap-input vector $\mathbf{u}_{M+1}(n)$, the forward prediction error at the filter output is

$$f_M(n) = \mathbf{a}_M^H \mathbf{u}_{M+1}(n)$$

The mean-squared value of $f_M(n)$ equals

$$\begin{aligned}\mathbb{E}[|f_M(n)|^2] =& \mathbb{E}[\mathbf{a}_M^H \mathbf{u}_{M+1}(n)\mathbf{u}_{M+1}^H(n)\mathbf{a}_M] \\ =& \mathbf{a}_M^H \mathbb{E}[\mathbf{u}_{M+1}(n)\mathbf{u}_{M+1}^H(n)]\mathbf{a}_M \\ =& \mathbf{a}_M^H \mathbf{R}_{M+1}\mathbf{a}_M\end{aligned}$$

Where $\mathbf{R}_{M+1} = \mathbb{E}\left[\mathbf{u}_{M+1}(n)\mathbf{u}_{M+1}^H(n)\right]$ is the correlation matrix of the tap-input vector.

b)

The leading element of the vector $\mathbf{a}$ equals 1. Hence, the constrained cost function to be minimized is

$$J(\mathbf{a}_M) = \mathbf{a}_M^H \mathbf{R}_M \mathbf{a}_M + \lambda \mathbf{a}_M^H \mathbf{1} + \lambda^* \mathbf{1}^T \mathbf{a}_M$$

where λ is the Lagrange multiplier and $\mathbf{1}$ is the first unit vector defined by $\mathbf{1} = \begin{bmatrix}1 & 0 & \dots & 0\end{bmatrix}$. Differentiating $J(\mathbf{a}_M)$ with respect to $\mathbf{a}_M$, and setting the result equal to zero yields

$$2\mathbf{R}_{M+1}\mathbf{a}_M + 2\lambda\mathbf{1} = \mathbf{0}$$

Solving for $\mathbf{a}_M$:

$$\mathbf{R}_{M+1}\mathbf{a}_M = \lambda\mathbf{1} \tag{1}$$

However, we may partition $\mathbf{R}_{M+1}$ as

$$\mathbf{R}_{M+1} = \begin{bmatrix} r(0) & \mathbf{r}^H \\ \mathbf{r} & \mathbf{R}_M \end{bmatrix}$$

Hence,

$$\begin{aligned} -\lambda =& [r((0), \mathbf{r}^H)]\mathbf{a}_M \\ =& P_M \end{aligned}$$

where P_M is the minimum prediction-error power. Accordingly, we may rewrite Equation (1) as

$$\mathbf{R}_{M+1}\mathbf{a}_M = P_M\mathbf{1} = \begin{bmatrix} P_M \\ \mathbf{0} \end{bmatrix}$$

Problem 3.2

a)

Let $\mathbf{a}_M^{B*}$ denote the tap-weight vector of the backward prediction-error filter. With a tap-input vector $\mathbf{u}_{M+1}(n)$ the backward prediction error is

$$b_M(n) = \mathbf{a}_M^{BT}\mathbf{u}_{M+1}(n)$$

The mean-square value of $b_M(n)$ is

$$\begin{aligned} \mathbb{E}[|b_M(n)|^2] =& \mathbb{E}[\mathbf{a}_M^{BT}\mathbf{u}_{M+1}(n)\mathbf{u}_{M+1}^H(n)\mathbf{a}_M^{B*}] \\ =& \mathbf{a}_M^{BT}\mathbb{E}[\mathbf{u}_{M+1}(n)\mathbf{u}_{M+1}^H(n)]\mathbf{a}_M^{B*} \\ =& \mathbf{a}_M^{BT}\mathbf{R}_{M+1}\mathbf{a}_M^{B*} \end{aligned}$$

b)

The last element of $\mathbf{a}_M^{B*}$ equals 1. Hence, the constrained objective function to be minimized is

$$J(\mathbf{a}_M) = \mathbf{a}_M^{BT}\mathbf{R}_{M+1}\mathbf{a}_M^{B*} + \lambda\mathbf{a}_M^{BT}\mathbf{1}^B + \lambda^*\mathbf{1}^{BT}\mathbf{a}_M^{B*}$$

where λ is the Lagrange multiplier, and thinking backwards:

$$\mathbf{1}^{BT} = \begin{bmatrix} 0 & 0 & \dots & 1 \end{bmatrix}$$

Differentiating $J(\mathbf{a}_M)$ with respect to $\mathbf{a}_M$,

$$2\mathbf{R}_{M+1}\mathbf{a}_M^{B*} + 2\lambda\mathbf{1}^B = \mathbf{0}$$

Solving for $\mathbf{a}_M^{B*}$, we get

$$\mathbf{R}_{M+1}\mathbf{a}_M^{B*} = -\lambda\mathbf{1}^B$$

However, we may express $\mathbf{R}_{M+1}$ in the partitioned form:

$$\mathbf{R}_{M+1} = \begin{bmatrix} \mathbf{R}_M & \mathbf{r}^{B*} \\ \mathbf{R}^{BT} & r(0) \end{bmatrix} \tag{1}$$

Therefore,

$$\begin{aligned} -\lambda &= \begin{bmatrix} \mathbf{r}^{BT} & \mathbf{r}(0) \end{bmatrix} \mathbf{a}_M^{B*} \\ &= P_M \end{aligned}$$

where P_M is the minimum backward prediction-error power. We may thus rewrite Equation (1) as

$$\mathbf{R}_{M+1}\mathbf{a}_M^{B*} = P_M\mathbf{1}^B = \begin{bmatrix} \mathbf{0} \\ P_M \end{bmatrix}$$

Problem 3.3

a)

Writing the Wiener-Hopf equation

$$\mathbf{R}\mathbf{g} = \mathbf{r}^{B*}$$

in expanded form, we have

$$\begin{bmatrix} r(0) & r(1) & \dots & r(M-1) \\ r(-1) & r(0) & \dots & r(M-2) \\ \vdots & \vdots & \ddots & \vdots \\ r(-M+1) & r(-M+2) & \dots & r(0) \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_M \end{bmatrix} = \begin{bmatrix} r(M) \\ r(M-1) \\ \vdots \\ r(1) \end{bmatrix}$$

Equivalently, we may write

$$\sum_{k=1}^{M} g_k\, r(k-i) = r(M+1-i), \quad i = 1, 2, \dots, M$$

Let $k = M - l + 1$, or $l = M - k + 1$. Then

$$\sum_{l=1}^{M} g_{M-l+1} r(M - l + 1 - i) = r(M + 1 - i), \quad i = 1, 2, \ldots, M$$

Next, put $M + 1 - i = j$, or $i = M + 1 - j$. Then

$$\sum_{l=1}^{M} g_{M-l+1} r(j - l) = r(j), \quad j = 1, 2, \ldots, M$$

Putting this relation into matrix form, we write

$$\begin{bmatrix} r(0) & r(1) & \ldots & r(-M+1) \\ r(-1) & r(0) & \ldots & r(-M+2) \\ \vdots & \vdots & \ddots & \vdots \\ r(M-1) & r(M-2) & \ldots & r(0) \end{bmatrix} \begin{bmatrix} g_M \\ g_{M-1} \\ \vdots \\ g_1 \end{bmatrix} = \begin{bmatrix} r(1) \\ r(2) \\ \vdots \\ r(M) \end{bmatrix}$$

This, in turn, may be put in the compact form

$$\mathbf{R}^T \mathbf{g}^B = \mathbf{r}^*$$

b)

The product $\mathbf{R}^{BT}\mathbf{g}$ is given by

$$\mathbf{R}^{BT}\mathbf{g} = \begin{bmatrix} r(-M) & r(-M+1) & \ldots & r(-1) \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_M \end{bmatrix}$$

$$= \sum_{k=1}^{M} g_k r(k - 1 - M) \tag{1}$$

The product $\mathbf{r}^T\mathbf{g}^B$ is given by

$$\mathbf{r}^T\mathbf{g}^B = \begin{bmatrix} r(-1) & r(-2) & \ldots & r(-M) \end{bmatrix} \begin{bmatrix} g_M \\ g_{M-1} \\ \vdots \\ g_1 \end{bmatrix}$$

$$\mathbf{r}^T\mathbf{g}^B = \sum_{k=1}^{M} g_{M+1-k} r(-k)$$

Put $M + 1 - k = l$, or $k = M + 1 - l$. Hence,

$$\mathbf{r}^T\mathbf{g}^B = \sum_{l=1}^{M} g_l r(l - 1 - M) \tag{2}$$

From Equations (1) and (2):

$$\mathbf{R}^{BT}\mathbf{g} = \mathbf{r}^T\mathbf{g}^B$$

Problem 3.4

Starting with the formula

$$r(m) = -\kappa_m^* P_{m-1} - \sum_{k=1}^{m-1} a_{m-1,k}^* r(m-k)$$

and solving for κ_m, we get

$$\kappa_m = \frac{r_m^*}{P_{m-1}} - \frac{1}{P_{m-1}} \sum_{k=1}^{m-1} a_{m-1,k} r^*(m-k) \tag{1}$$

We also note:

$$a_{m,k} = a_{m-1,k} + \kappa_m a_{m-1,m-k}^*, \quad k = 0, 1, \ldots, m \tag{2}$$

$$P_m = P_{m-1}(1 - |\kappa_m|^2) \tag{3}$$

a)

We are given

$r(0) = 1$
$r(1) = 0.8$
$r(2) = 0.6$
$r(3) = 0.4$

We also note that

$$P_0 = r(0) = 1$$

Hence, the use of Equation (1) for m=1 yields

$$\kappa_1 = -\frac{r^*(1)}{P_0} = 0.8$$

The use of Equation (3) for m=1 yields

$$P_1 = P_0(1 - |\kappa|^2) = 1 - 0.64 = 0.36$$

We next reapply Equation (1) for m=2

$$\kappa_2 = -\frac{r^*(2)}{P_1} - \frac{r^*(1)}{P_1}\kappa_1$$

where we have noted that

$$\kappa_1 = a_{1,1}$$

Hence,

$$\kappa_2 = -\frac{0.6}{0.36} + \frac{0.8 \times 0.8}{0.36} = \frac{0.04}{0.36} = \frac{1}{9} \approx 0.1111$$

Next, the use of Equation (3) for m=2 yields

$$P_2 = P_1(1 - |\kappa_2|^2)$$

$$P_2 = 0.36\left(1 - \frac{1}{81}\right) \approx 0.04$$

Next, we reapply Equation (1) for m=3:

$$\kappa_3 = -\frac{r^*(3)}{P_2}(a_{2,1}r^*(2) + a_{2,2}r^*(1))$$

From Equation (2) we have

$$a_{2,2} = \kappa_2 = \frac{1}{9} = 0.1111$$

$$\begin{aligned} a_{2,1} =& a_{1,1} + \kappa_2 a^*_{1,1} = \kappa_1 + \kappa_2\kappa_1 \\ =& -\frac{4}{5} - \frac{4}{90} \times \frac{4}{5} = -\frac{188}{225} = -0.8356 \end{aligned}$$

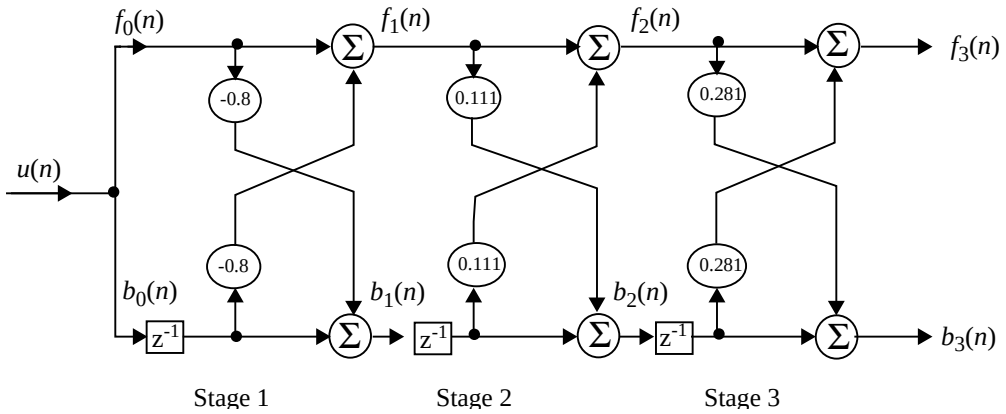
Hence,

$$\begin{aligned} \kappa_3 =& -\frac{0.4}{0.04} - \frac{1}{0.04}(-0.8356 \times 0.6 + 0.111 \times 0.8) \\ =& -1 + \frac{0.41248}{0.04} = \frac{0.01248}{0.04} = 0.312 \end{aligned}$$

Note that all the reflection coefficients have a magnitude less than unity; hence, the lattice-predictor (prediction-error filter) representation of the process is minimum phase.

b)

This lattice-predictor representation is as shown in the following figure:



c)

From the calculations presented in part **a)**, we have

$$P_0 = 1$$

$$P_1 = 0.36$$

$$P_2 = 0.04$$

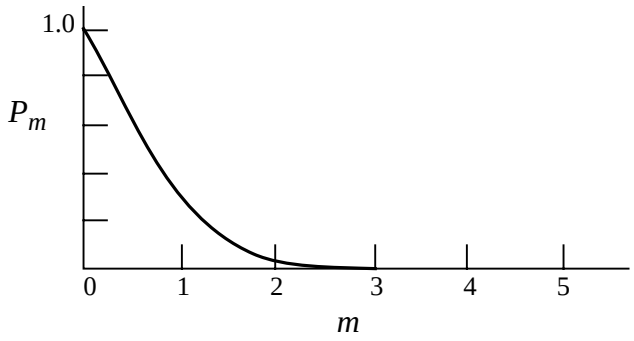
To Complete the calculations required, we note that

$$P_3 = P_2(1 - |\kappa_3|^2)$$

$$P_3 = 0.04(1 - 0.281^2) = 0.04(1 - 0.079)$$

$$P_3 = 0.03684$$

From the power plot



We note that the average power P_m decreases exponentially with the prediction order m.

Problem 3.5

The estimation error $e(n)$ is

$$e(n) = u(n) - \mathbf{w}^H \mathbf{x}(n)$$

where

$$\mathbf{x}(n) = \mathbf{u}(n - \Delta)$$

$$\mathbf{x}(n) = \begin{bmatrix} u(n-\Delta) & u(n-1-\Delta) & \dots & u(n-M-\Delta) \end{bmatrix}^T$$

The mean-square value of the estimation error is

$$\begin{aligned} J =& \mathbb{E}[|e(n)|^2] \\ =& \mathbb{E}[(u(n) - \mathbf{w}^H \mathbf{x}(n))(u^*(n) - \mathbf{x}^H(n)\mathbf{w})] \\ =& \mathbb{E}[|u(n)|^2] - \mathbf{w}^H \mathbb{E}[\mathbf{x}(n)u^*(n)] - \mathbb{E}[u(n)\mathbf{x}^H(n)]\mathbf{w} + \mathbf{w}^H \mathbb{E}[\mathbf{x}(n)\mathbf{x}^H(n)]\mathbf{w} \\ =& P_0 - \mathbf{w}^H \mathbb{E}[\mathbf{u}(n-\Delta)u^*(n)] - \mathbb{E}[u(n)\mathbf{u}^H(n-\Delta)]\mathbf{w} + \mathbf{w}^H \mathbb{E}[\mathbf{u}(n-\Delta)\mathbf{u}^H(n-\Delta)]\mathbf{w} \end{aligned} \tag{1}$$

We next note the following:

$$\begin{aligned} \mathbb{E}[\mathbf{u}(n-\Delta)u^*(n)] =& \mathbb{E}\left\{ \begin{bmatrix} u(n-\Delta) \\ u(n-1-\Delta) \\ \vdots \\ u(n-M-\Delta) \end{bmatrix} \right\} u^*(n) \\ =& \begin{bmatrix} r(-\Delta) \\ r(-\Delta-1) \\ \vdots \\ r(-\Delta-M) \end{bmatrix} = \mathbf{r}_\Delta \end{aligned}$$

and

$$\mathbb{E}[u(n)\mathbf{u}^H(n-\Delta)] = \begin{bmatrix} r(-\Delta) \\ r(-\Delta-1) \\ \vdots \\ r(-\Delta-M) \end{bmatrix}^H = \mathbf{r}_\Delta^H$$

$$\mathbb{E}[\mathbf{u}(n-\Delta)\mathbf{u}^H(n-\Delta)] = \mathbf{R}$$

We may thus rewrite Equation (1) as

$$J = P_0 - \mathbf{w}^H\mathbf{r}_\Delta - \mathbf{r}_\Delta^H\mathbf{w} + \mathbf{R}$$

The optimum value for the weight vector is therefore

$$\mathbf{w}_0 = \mathbf{R}^{-1}\mathbf{r}_\Delta$$

where $\mathbf{R}^{-1}$ is the inverse of the correlation matrix $\mathbf{R}$

Problem 3.6

We are given the difference equation

$$u(n) = 0.9u(n-1) + v(n)$$

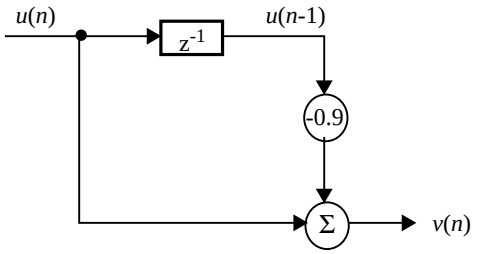
a)

For a prediction-error filter of order two, we have

$a_{2,1} = -0.9$

$a_{2,2} = 0$

The prediction-error filter representation of the process is therefore



b)

The corresponding reflection coefficients of the lattice predictor are

$\kappa_1 = a_{2,1} = -0.9$

$\kappa_2 = a_{2,2} = 0$

We are given a first-order difference equation as the description of the AR process $u(n)$. It is therefore natural that we use a first-order predictor for the representation of this process.

Problem 3.7

a.i)

The tap-weight vector of the prediction-error filter of order M is

$$\mathbf{a}_M = \begin{bmatrix} 1 \\ -\mathbf{w}_0 \end{bmatrix} \tag{1}$$

where

$$\mathbf{w}_0 = \mathbf{R}_M^{-1}\mathbf{r}_M \tag{2}$$

$$\begin{aligned} \mathbf{r}_M =& \mathbb{E}[\mathbf{u}_M(n-1)u^*(n)] \\ =& \sigma_\alpha^2 \begin{bmatrix} e^{-\mathrm{j}\,\omega} \\ e^{-\mathrm{j}\,2\omega} \\ \vdots \\ e^{-\mathrm{j}\,M\omega} \end{bmatrix} = \sigma_\alpha^2 e^{-\mathrm{j}\,\omega}\mathbf{s}_M(\omega) \end{aligned} \tag{3}$$

$$\mathbf{R}_M = \sigma_\alpha^2 \mathbf{s}_M(\omega)\mathbf{s}_M^H(\omega) + \sigma_v^2\mathbf{I}_M$$

$$\mathbf{s}_M(\omega) = \begin{bmatrix} 1 \\ e^{-\mathrm{j}\,\omega} \\ \vdots \\ e^{-\mathrm{j}\,\omega(M-1)} \end{bmatrix} \tag{4}$$

From the matrix inversion lemma (see Chapter 10 in the textbook), we have:
If

$$\mathbf{A} = \mathbf{B}^{-1} + \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^H$$

Then

$$\mathbf{A}^{-1} = \mathbf{B} - \mathbf{C}\mathbf{D}(\mathbf{D} + \mathbf{C}^H\mathbf{B}\mathbf{C})^{-1}\mathbf{C}^H\mathbf{B}$$

For our application:

$$\mathbf{A} = \mathbf{R}_M$$

$$\mathbf{B}^{-1} = \sigma_v^2\mathbf{I}_M$$

$$\mathbf{D}^{-1} = \sigma_\alpha^2$$

$$\mathbf{C} = \mathbf{s}_M(\omega)$$

Hence,

$$\begin{aligned}\mathbf{R}_M^{-1} =& \frac{1}{\sigma_v^2}\mathbf{I}_M - \frac{\dfrac{1}{\sigma_v^4}}{\dfrac{1}{\sigma_\alpha^2} + \dfrac{1}{\sigma_v^2}\mathbf{s}_M^H(\omega)\mathbf{s}_M(\omega)}\mathbf{s}_M(\omega)\mathbf{s}_M^H(\omega) \\ =& \frac{1}{\sigma_v^2}\mathbf{I}_M - \frac{\dfrac{1}{\sigma_v^2}}{\dfrac{\sigma_v^2}{\sigma_\alpha^2} + \mathbf{s}_M^H(\omega)\mathbf{s}_M(\omega)}\mathbf{s}_M(\omega)\mathbf{s}_M^H(\omega)\end{aligned}$$

we also note that

$$\mathbf{s}_M^H(\omega)\mathbf{s}_M(\omega) = M$$

Hence,

$$\mathbf{R}_M^{-1} = \frac{1}{\sigma_v^2}\mathbf{I}_M - \frac{\dfrac{1}{\sigma_v^2}}{\dfrac{\sigma_v^2}{\sigma_\alpha^2} + M}\mathbf{s}_M(\omega)\mathbf{s}_M^H(\omega) \tag{5}$$

Substituting Equations (3) and (5) into (2) yields

$$\begin{aligned}\mathbf{w}_0 &= \left(\frac{\sigma_\alpha^2}{\sigma_\nu^2}\right)\exp\left(-\mathrm{j}\,\omega\right)\mathbf{s}_M(\omega) - \frac{\left(\dfrac{\sigma_\alpha^2}{\sigma_\nu^2}\right)\exp\left(-\mathrm{j}\,\omega\right)}{\left(\dfrac{\sigma_\nu^2}{\sigma_\alpha^2}\right) + M}\mathbf{s}_M(\omega)\mathbf{s}_M^H(\omega)\mathbf{s}_M(\omega) \\ &= \left(\frac{\sigma_\alpha^2}{\sigma_\nu^2}\right)\exp\left(-\mathrm{j}\,\omega\right)\left[1 - \frac{M}{\left(\dfrac{\sigma_\nu^2}{\sigma_\alpha^2}\right) + M}\right]\mathbf{s}_M(\omega) \\ &= \left(\frac{\exp(-\mathrm{j}\,\omega)}{\left(\dfrac{\sigma_\nu^2}{\sigma_\alpha^2}\right) + M}\right)\mathbf{s}_M(\omega)\end{aligned} \tag{6}$$

Equations (1) and (6) define the tap-weight vector of the prediction-error filter. Moreover, the final value of the prediction-error power is

$$\begin{aligned} P_M =& r(0) - \mathbf{r}_m^H \mathbf{w}_0 \\ =& r(0) - \left(\frac{\sigma_\alpha^2}{\left(\frac{\sigma_\nu^2}{\sigma_\alpha^2}\right) + M} \right) \mathbf{s}_M^H(\omega)\mathbf{s}_M(\omega) \end{aligned} \tag{7}$$

Using (4) and the fact that

$$r(0) = \sigma_\alpha^2 + \sigma_\nu^2$$

we may rewrite Equation (7) as

$$\begin{aligned} P_M =& \sigma_\alpha^2 + \sigma_\nu^2 - \frac{M\sigma_\alpha^2}{\left(\frac{\sigma_\nu^2}{\sigma_\alpha^2}\right) + M} \\ =& \frac{\sigma_\nu^2 \left[1 + M + \left(\frac{\sigma_\nu^2}{\sigma_\alpha^2}\right)\right]}{\left(\frac{\sigma_\nu^2}{\sigma_\alpha^2}\right) + M} \end{aligned} \tag{8}$$

a.ii)

The mth refelction coefficient is [from Equation (3.56) of the text]

$$\kappa_m = -\frac{\Delta_{m-1}}{P_{m-1}}$$

where

$$\Delta_{m-1} = \mathbf{r}_m^{BT} \mathbf{a}_{m-1}$$

Hence,

$$\kappa_m = -\frac{\mathbf{r}_m^{BT} \mathbf{a}_{m-1}}{P_{m-1}} \tag{9}$$

From Equation (3) we deduce that

$$\begin{aligned} \mathbf{r}_m^B =& \sigma_\alpha^2 \exp(-\mathrm{j}\,\omega)\mathbf{s}_m^B(\omega) \\ =& \sigma_\alpha^2 \begin{bmatrix} \exp(-\mathrm{j}\,m\omega) \\ \exp(-\mathrm{j}(m-1)\omega) \\ \vdots \\ \exp(-\mathrm{j}\,\omega) \end{bmatrix} \end{aligned} \tag{10}$$

From Equation (8) we have:

$$P_{m-1} = \frac{\sigma_\nu^2 \left[m + \left(\frac{\sigma_\nu^2}{\sigma_\alpha^2} \right) \right]}{\left(\frac{\sigma_\nu^2}{\sigma_\alpha^2} \right) + m - 1} \tag{11}$$

From Equations (1) and (6):

$$\mathbf{a}_{m-1} = \begin{bmatrix} 1 \\ -\left(\frac{\exp(-\mathrm{j}\,\omega)}{\left(\frac{\sigma_\nu^2}{\sigma_\alpha^2} \right) + m - 1} \right) \mathbf{s}_{m-1}(\omega) \end{bmatrix} \tag{12}$$

Hence, the combined use of Equations (9)-(12) yields

$$\begin{aligned} \kappa_m &= -\frac{\sigma_\alpha^2 \left[\sigma_\nu^2 + (m-1)\sigma_\alpha^2 \right] \exp\left(-\mathrm{j}\, m\omega\right)}{\sigma_\nu^2 \left(\sigma_\nu^2 + m\sigma_\alpha^2 - \sigma_\alpha^2 \right)} + \frac{\sigma_\alpha^4 \exp(-\mathrm{j}\, m\omega)(m-1)}{\sigma_\nu^2 \left(\sigma_\nu^2 + m\sigma_\alpha^2 - \sigma_\alpha^2 \right)} \\ &= -\frac{\sigma_\alpha^2 \exp(-\mathrm{j}\, m\omega)}{\sigma_\nu^2 + m\sigma_\alpha^2 - \sigma_\alpha^2} \end{aligned}$$

a.iii)

When we let the noise variance σ_ν^2 approach zero, the mth reflection coefficient κ_m approaches $\frac{1}{m-1} \exp(-\mathrm{j}\, m\omega)$, the magnitude of which, in turn approaches zero as m approaches infinity.

b.i)

The tap-weight vector of the prediction-error filter of order M is

$$\mathbf{a}_M = \begin{bmatrix} 1 \\ \alpha^* \exp(-\mathrm{j}\,\omega) \\ \mathbf{0}_{M-1} \end{bmatrix}$$

where $\mathbf{0}_{M-1}$ is a null vector of dimension M-1.

b.ii)

The reflection coefficients of a lattice predictor of order M are

$$\kappa_1 = \alpha^* \exp(-\mathrm{j}\,\omega)$$

$$\kappa_m = 0 \quad \text{for } m = 2, \ldots, M.$$

c)
We may consider $u_1(n) = u_2(n)$ under the limiting conditions:

$$|\alpha| \to 1$$

and

$$\sigma_\nu^2 \to 0$$

where σ_ν^2 refers to the noise in the AR process.

Problem 3.8

$$\mathbf{a}_m = \begin{bmatrix} \mathbf{a}_{m-1} \\ 0 \end{bmatrix} + \kappa_m \begin{bmatrix} 0 \\ \mathbf{a}_{m-1}^{B*} \end{bmatrix}$$

In expanded form, we have

$$\begin{bmatrix} a_{m,0} \\ a_{m,1} \\ \vdots \\ a_{m,m-1} \\ a_{m,m} \end{bmatrix} = \begin{bmatrix} a_{m-1,0} \\ a_{m-1,1} \\ \vdots \\ a_{m-1,m-1} \\ 0 \end{bmatrix} + \kappa_m \begin{bmatrix} 0 \\ a_{m-1,m-1}^* \\ \vdots \\ a_{m-1,1}^* \\ a_{m-1,0}^* \end{bmatrix}$$

or equivalently,

$$a_{m,k} = a_{m-1,k} + \kappa_m a_{m-1,m-k}^*, \qquad k = 0, 1, \ldots, M$$

Put $m - k = l$ or $k = m - l$. Then,

$$a_{m,m-l} = a_{m-1,m-l} + \kappa_m a_{m-1,l}^*, \qquad l = 0, 1, \ldots, m$$

Complex conjugate both sides:

$$a_{m,m-l}^* = a_{m-1,m-l}^* + \kappa_m^* a_{m-1,l}, \qquad l = 0, 1, \ldots, m$$

Rewrite this relationship in matrix form:

$$\begin{bmatrix} a_{m,m}^* \\ a_{m,m-1}^* \\ \vdots \\ a_{m-1}^* \\ a_{m-0}^* \end{bmatrix} = \begin{bmatrix} 0 \\ a_{m-1,m-1}^* \\ \vdots \\ a_{m-1,1}^* \\ a_{m-1,0}^* \end{bmatrix} + \kappa_m^* \begin{bmatrix} a_{m-1,0} \\ a_{m-1,1} \\ \vdots \\ a_{m-1,m-1} \\ 0 \end{bmatrix}$$

or equivalently

$$\mathbf{a}_m^{B*} = \begin{bmatrix} 0 \\ \mathbf{a}_{m-1}^{B*} \end{bmatrix} + \kappa_m^* \begin{bmatrix} \mathbf{a}_{m-1} \\ 0 \end{bmatrix}$$

Note that (for all m)

$$a_{m,k} = \begin{cases} 1, & k = 0 \\ 0, & k > m \end{cases}$$

Problem 3.9

We start with the formula

$$\Delta_{m-1} = \sum_{k=0}^{m-1} a_{m-1,k} r(k-m) \tag{1}$$

The autocorrelation function $r(k-m)$ is by definition:

$$r(k-m) = \mathbb{E}\left[u(n-m)u^*(n-k)\right] \tag{2}$$

Hence, substituting Equation (2) into (1)

$$\begin{aligned} \Delta_{m-1} =& \sum_{k=0}^{m-1} a_{m-1,k} \mathbb{E}\left[u(n-m)u^*(n-k)\right] \\ =& \mathbb{E}\left[u(n-m)\sum_{k=0}^{m-1} a_{m-1,k} u^*(n-k)\right] \end{aligned} \tag{3}$$

But, by definition:

$$f_{m-1}(n) = \sum_{k=0}^{m-1} a_{m-1,k}^* u(n-k)$$

Hence, we may rewrite Equation (3) as

$$\Delta_{m-1} = \mathbb{E}\left[u(n-m)f_{m-1}^*(n)\right] \tag{4}$$

Next, we note that

$$u(n-m) = \hat{u}(n-m|U_{n-1}) + b_{m-1}(n-1) \tag{5}$$

where U_{n-1} is the space spanned by $u(n-1), \ldots, u(n-m+1)$, and b_{m-1} is the backward prediction error produced by a predictor of order $m-1$. The estimate $\hat{u}(n-m|U_{n-1})$ is therefore

$$\hat{u}(n-m|U_{n-1}) = \sum_{k=1}^{m-1} w_{b,k}^* u(n-k) \tag{6}$$

Accordingly, using Equations (6) and (5) in (3):

$$\begin{aligned}\Delta_{m-1} =& \mathbb{E}\left[b_{m-1}(n-1) f_{m-1}^*(n)\right] + \mathbb{E}\left[\sum_{k=1}^{m-1} w_{b,k}^* u(n-k) f_{m-1}^*(n)\right] \\ =& \mathbb{E}\left[b_{m-1}(n-1) f_{m-1}^*(n)\right] + \sum_{k=1}^{m-1} w_{b,k}^* \mathbb{E}\left[u(n-k) f_{m-1}^*(n)\right] \end{aligned} \tag{7}$$

But

$$\mathbb{E}\left[f_{m-1}(n) u^*(n-k)\right] = 0, \qquad 1 \leq k \leq m-1$$

Hence, Equation (7) simplifies to

$$\Delta_{m-1} = \mathbb{E}\left[b_{m-1}(n-1) f_{m-1}^*(n)\right]$$

Problem 3.10

The polynomial $x(z)$ is given by

$$\begin{aligned} x(z) =& a_{M,M} z^M + a_{M,M-1} z^{M-1} + \ldots + a_{M,0} \\ =& \sum_{k=0}^{M} a_{M,k} z^k \\ =& z^M H_{b,M}(z) \end{aligned}$$

where $H_{b,M}(z)$ is the transfer function of a backward prediction-error filter of order M. The reciprocal polynomial $x'(z)$ is defined by

$$\begin{aligned} x'(z) =& a_{M,M}^* + a_{M,M-1}^* z + \ldots + a_{M,0}^* z^M \\ =& z^M H_{f,M}(z) \end{aligned}$$

where $H_{f,M})(z)$ is the transfer function of a forward prediction-error filter of order M.

Next, we note, by definition, that

$$\begin{aligned} T\left[x(z)\right] =& a^*_{M,0}x(z) - a_{M,M}x'(z) \\ =& \sum_{k=0}^{M} a_{M,k}z^k - a_{M,M}\sum_{k=0}^{M} a^*_{M,M-k}z^k \\ =& \sum_{k=0}^{M} \left(a_{M,k} - a_{M,M}a^*_{M,M-k}\right) z^k \end{aligned}$$

But, from the inverse recursion:

$$a_{M-1,k} = \frac{a_{M,k} - a_{M,M}a^*_{M,M-k}}{1 - \left|a_{M,M}\right|^2}, \qquad k = 0, 1, \ldots, M$$

Therefore,

$$\begin{aligned} T\left[x(z)\right] &= \left(1 - \left|a_{M,M}\right|^2\right) \sum_{k=0}^{M} a_{M-1,k}z^k \\ &= \left(1 - \left|a_{M,M}\right|^2\right) \left[z^{M-1}H_{b,M-1}(z)\right] \end{aligned}$$

where we have used the fact that $a_{M-1,M}$ is zero. This shows that $T\left[x(z)\right]$ is of order M-1, one less than the order of the original polynomial $x(z)$.

Similarly, we have

$$T^2\left[x(z)\right] = \left(1 - \left|a_{M,M}\right|^2\right) \left(1 - \left|a_{M-1,M-1}\right|^2\right) \left[z^{M-2}H_{b,M-2}(z)\right]$$

We also note that

$$T^1\left[x(0)\right] = 1 - \left|a_{M,M}\right|^2$$

$$T^2\left[x(0)\right] = \left(1 - \left|a_{M,M}\right|^2\right) \left(1 - \left|a_{M-1,M-1}\right|^2\right)$$

Thus:

$$T^0\left[x(z)\right] = T^0\left[x(0)\right] z^M H_{b,M}(z)$$

$$T^1\left[x(z)\right] = T^1\left[x(0)\right] z^{M-1} H_{b,M-1}(z)$$

$$T^2\left[x(z)\right] = T^2\left[x(0)\right] z^{M-2} H_{b,M-2}(z)$$

where, in the first line, we have

$$T^0\left[x(z)\right] = x(z)$$

and

$$x(0) = 1$$

We may generalize these results by writing

$$T^i\left[x(z)\right] = T^i\left[x(0)\right] z^{M-i} H_{b,M-i}(z)$$

where

$$T^i\left[x(0)\right] = \prod_{j=0}^{i-1}\left(1 - |a_{M-j,M-j}|^2\right), \qquad 1 \leq i \leq M$$

Problem 3.11

a)

The AR parameters are as follows:

$$a_1 = -1$$

$$a_2 = \frac{1}{2}$$

Since $\nu(n)$ has zero mean, the average power of $u(n)$ is

$$\begin{aligned} P_0 =& r(0) \\ =& \left(\frac{1+a_2}{1-a_2}\right) \frac{\sigma_\nu^2}{[(1+a_2) - a_1^2]} \\ =& 1.2 \end{aligned}$$

b)

For prediction order M=2, the prediction-error filter coefficients equal the AR parameters:

$$a_{2,1} = a_1 = -1$$

$$a_{2,2} = a_2 = \frac{1}{2}$$

The use of the inverse Levinson-Durbin recursion for real-valued data yields

$$a_{m-1,k} = \frac{a_{m,k} - a_{m,m}\, a_{m,m-k}}{1 - |a_{m,m}|^2}, \qquad k = 0, \ldots, M$$

For $m = 2$, we have

$$a_{1,k} = \frac{a_{2,k} - a_{2,2}\, a_{2,2-k}}{1 - |a_{2,2}|^2}, \qquad k = 0, 1$$

Hence,

$$a_{1,0} = 1$$

$$\begin{aligned} a_{1,1} =& \frac{a_{2,1} - a_{2,2}\, a_{2,1}}{1 - |a_{2,2}|^2} \\ =& -\frac{2}{3} \end{aligned}$$

The reflection coefficients are thus as follows

$$\kappa_1 = a_{1,1} = -\frac{2}{3}$$

$$\kappa_2 = a_{2,2} = \frac{1}{2}$$

c)

Using the formula (for real-valued data)

$$P_m = P_{m-1}\left(1 - \kappa_m^2\right)$$

yields the following values for the average prediction-error powers:

$$P_1 = P_0\left(1 - \kappa_1^2\right) = \frac{2}{3}$$

$$P_2 = P_1\left(1 - \kappa_2^2\right) = \frac{1}{2}$$

Problem 3.12

For real data, we have

$$r(m) = -\kappa_m P_{m-1} - \sum_{k=1}^{m-1} a_{m-1,k}\, r(m-k)$$

For m=1,

$$\begin{aligned} r(1) &= -\kappa_1 P_0 \\ &= 0.8 \end{aligned}$$

For m=2,

$$\begin{aligned} r(2) &= -\kappa_2 P_1 - a_{1,1}\, r(1) \\ &= 0.2 \end{aligned}$$

Problem 3.13

a)

The transfer function of the forward prediction-error filter is

$$H_{f,M}(z) = \left(1 - z_i z^{-1}\right) c_i(z)$$

where

$$z_i = \rho_i \exp\left(\mathrm{j}\,\omega_i\right)$$

The power spectrum density of the prediction error $f_M(n)$ is

$$S_f(\omega) = \left|H_{f,M}\left(\exp\left(\mathrm{j}\,\omega\right)\right)\right|^2 S(\omega)$$

where $S(\omega)$ is the power spectral density of the input process $u(n)$. Hence, the mean-square value of the prediction error $f_M(n)$ is

$$\begin{aligned} \varepsilon &= \int_{-\pi}^{\pi} \left|H_{f,M}\left(\exp\left(\mathrm{j}\,\omega\right)\right)\right|^2 S(\omega)\,\mathrm{d}(\omega) \\ &= \int_{-\pi}^{\pi} \left|1 - \rho_i \exp\left(\mathrm{j}\,\omega_i\right)\exp\left(-\mathrm{j}\,\omega\right)\right|^2 \left|C_i\left(\exp\left(\mathrm{j}\,\omega\right)\right)\right|^2 S(\omega)\,\mathrm{d}(\omega) \\ &= \int_{-\pi}^{\pi} \left[1 - 2\rho_i \cos\left((\omega - \omega_i) + \rho_i^2\right)\right] \left|C_i\left(\exp\left(\mathrm{j}\,\omega\right)\right)\right|^2 S(\omega)\,\mathrm{d}(\omega) \end{aligned}$$

b)
Differentiating ε with respect to ρ_i:

$$\frac{\partial \varepsilon}{\partial \rho_i} = 2 \int_{-\pi}^{\pi} \left[-\cos\left(\omega - \omega_i\right) + \rho_i \right] \left| C_i \left(\exp\left(\mathrm{j}\, \omega \right) \right) \right|^2 S(\omega)\, \mathrm{d}(\omega)$$

If z_i lies outside the unit circle, $\rho_i > 1$, we then note that (regardless of ρ_i)

$$-1 \leq \cos(\omega - \omega_i) \leq 1, \qquad -\pi \leq (\omega, \omega_i) \leq \pi$$

Hence,

$$-\cos(\omega - \omega_i) + \rho_i > 0, \qquad \text{if } \rho_i > 1$$

Since $|C_i(\exp(\mathrm{j}\,\omega))|^2$ and $S(\omega)$ are both positive, if follows that

$$\frac{\partial \varepsilon}{\partial \rho_i} > 0, \qquad \text{if } \rho_i > 1$$

If the prediction-error filter is to be optimum, its parameters (and therefore the ρ_i) must be chosen in such a way that $\dfrac{\partial \varepsilon}{\partial \rho_i} = 0$.

Hence it is not possible for $\rho_i > 1$ and yet satisfy the optimality condition $\dfrac{\partial \varepsilon}{\partial \rho_i} = 0$.

The conclusion to be drawn is that the transfer function of a forward prediction-error filter (that is optimum) cannot have any of its zeros outside of the unit circle. In other words, a forward prediction-error filter is necessarily minimum phase

Problem 3.14

An AR process $u(n)$ of order M is described by the difference equation

$$u(n) = -a_1^* u(n-1) - \ldots - a_M^* u(n-M) + \nu(n)$$

Equivalently, in the z-domain we have

$$U(z) = \frac{1}{1 + a_1^* z^{-1} + \ldots + a_M^* z^{-M}} V(z)$$

When the process $u(n)$ is applied to a forward prediction-error filter described by the transfer function

$$H_f(z) = 1 + a_1^* z^{-1} + \ldots + a_M^* z^{-M},$$

the z-transformation of the resulting output is

$$H_f(z)U(z) = V(z)$$

In other words, the output consists of a white noise sequence $\nu(n)$.

Suppose next the process $u(n)$ is applied to a backward prediction-error filter described by the transfer function

$$H_b(z) = a_M + a_{M-1} z^{-1} + \ldots + a_1 z^{-M-1} + z^{-M}$$

The z-transform of the resulting output is

$$H_b(z)U(z) = \frac{a_M + a_{M-1} z^{-1} + \ldots + a_1 z^{-M+1} + z^{-M}}{1 + a_1^* z^{-1} + \ldots + a_M^* z^{-M}}$$

This rational function is recognized to be an all-pass (non-minimum phase) function with unit magnitude but non-zero phase. Equivalently, we may state that the corresponding output sequence is an anti-causal realization of white noise.

Problem 3.15

Let

$$I = \frac{1}{2\pi \,\mathrm{j}} \oint_C \frac{1}{z} \phi_m(z) \phi_k\left(z^{-1}\right) S(z) \,\mathrm{d}\, z$$

On the unit circle,

$$z = \exp\left(\mathrm{j}\,\omega\right)$$

$$\mathrm{d}\, z = \mathrm{j} \exp\left(\mathrm{j}\,\omega\right) \mathrm{d}\,\omega$$

Hence,

$$I = \frac{1}{2\pi} \int_{-\pi}^{\pi} \phi_m(\exp\left(\mathrm{j}\,\omega\right)) \phi_k\left(\exp\left(-\mathrm{j}\,\omega\right)\right) S(\omega) \,\mathrm{d}\,\omega \tag{1}$$

From the definition of $\phi_m(z)$, we have

$$\phi_m(\exp(\mathrm{j}\,\omega)) = \frac{1}{\sqrt{P_m}} \sum_{i=0}^{m} a_{m,i} \exp\left(\mathrm{j}\,\omega\,(m-i)\right)$$

Hence, we may rewrite Equation (1) as

$$\begin{aligned} I =& \frac{1}{2\pi\sqrt{P_m P_k}} \int_{-\pi}^{\pi} \sum_{i=0}^{m} \sum_{l=0}^{k} a_{m,i} a_{k,l} \exp(\mathrm{j}\,\omega(m-i)) \exp(-\mathrm{j}\,\omega(k-l)) S(\omega)\,\mathrm{d}\,\omega \\ =& \frac{1}{2\pi\sqrt{P_m P_k}} \sum_{i=0}^{m} \sum_{l=0}^{k} a_{m,i} a_{k,l} \int_{-\pi}^{\pi} S(\omega) \exp(\mathrm{j}\,\omega(m-k+l-i))\,\mathrm{d}\,\omega \end{aligned} \tag{2}$$

From the Einstein-Wiener-Khintchine relations:

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} S(\omega) \exp(\mathrm{j}\,\omega(m-k+l-i))\,\mathrm{d}\,\omega = r(m-k+l-i)$$

Accordingly, we may simplify Equation (2) as

$$I = \frac{1}{\sqrt{P_m P_k}} \sum_{i=0}^{m} \sum_{l=0}^{k} a_{m,i} a_{k,l} r(m-k+l-i) \tag{3}$$

From the augmented Wiener-Hopf equation for linear prediction we have

$$\sum_{l=0}^{k} a_{m,i} r(m-k+l-i) = \begin{cases} P_k, & \text{if } m = k \text{ and } l = 0 \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

Substituting Equation (4) into (3), we get

$$I = \begin{cases} 1, & \text{if } m = k \\ 0, & \text{otherwise} \end{cases}$$

where it is noted that $a_{k,l} = 1$ for $l = 0$. Equivalently,

$$I = \delta_{mk}$$

as required.

Problem 3.16

From Equations (3.81) and (3.82) in the textbook

$$H_{f,m}(z) = \prod_{i=1}^{m} \left(1 - z_i z^{-1}\right)$$

$$\begin{aligned} H_{b,m}(z) =& \prod_{i=1}^{m} \left(z^{-1} - z_i^*\right) \\ =& \prod_{i=1}^{m} \left(\frac{z^{-1} - z_i^*}{1 - z_i z^{-1}}\right) \left(1 - z_i z^{-1}\right) \\ =& H_{f,m}(z) \prod_{i=1}^{m} \left(\frac{z^{-1} - z_i^*}{1 - z_i z^{-1}}\right) \end{aligned}$$

The factor $\left(z^{-1} - z_i^*\right) / \left(1 - z_i z^{-1}\right)$ represents an all-pass structure with a magnitude response equal to unity for $z = \exp(\mathrm{j}\,\omega)$ at all ω. Hence,

$$\left|H_{b,m}\left(\exp\left(\mathrm{j}\,\omega\right)\right)\right| = \left|H_{f,m}\left(\exp\left(\mathrm{j}\,\omega\right)\right)\right|$$

Given an input $u(n)$ of power spectral density $S_u(\omega)$ applied to both $H_{f,m}\left(\exp\left(\mathrm{j}\,\omega\right)\right)$ and $H_{b,m}\left(\exp\left(\mathrm{j}\,\omega\right)\right)$, we immediately see that

$$\begin{aligned} S_{f,m}(\omega) =& S_u(\omega) \left|H_{f,m}\left(\exp\left(\mathrm{j}\,\omega\right)\right)\right|^2 \\ =& S_u(\omega) \left|H_{b,m}\left(\exp\left(\mathrm{j}\,\omega\right)\right)\right|^2 \\ =& S_{b,m}(\omega) \quad \text{for all } m. \end{aligned}$$

Problem 3.17

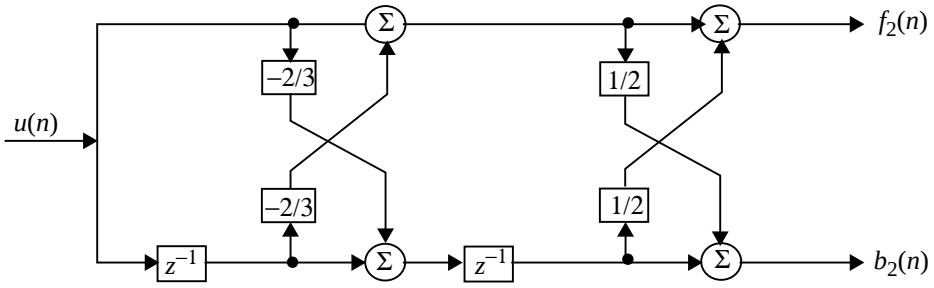
a)

The reflection coefficients of the two-stage lattice predictor are given to be:

$$\kappa_1 = -\frac{2}{3}$$

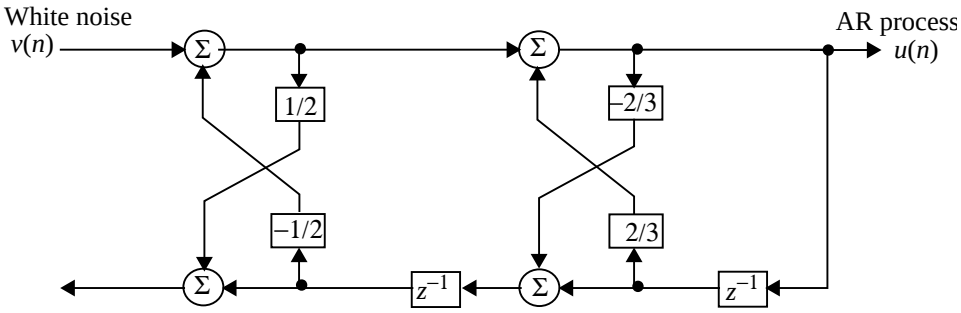
$$\kappa_2 = \frac{1}{2}$$

Hence, the structure of this lattice predictor for real data is as follows:



b)

The inverse lattice filter for generating the second-order AR process from a white noise process is as follows (see Fig 3.11 in the textbook):



From this latter structure, we readily see that

$$u(n) = \frac{2}{3}u(n-1) + \left(-\frac{2}{3}\right)\left(-\frac{1}{2}\right)u(n-1) + \left(-\frac{1}{2}\right)u(n-2) + \nu(n)$$

That is,

$$u(n) = u(n-1) - 0.5u(n-2) + \nu(n)$$

which is exactly the same as the difference equation specified in Problem 3.11.

Problem 3.18

a)

From Fig. 3.10, in the textbook, $f_m(n)$ is obtained by passing $u(n)$ through a minimum-phase prediction-error filter of order M, whereas $b_M(n)$ is obtained by passing $u(n)$

through a maximum-phase prediction-error filter of order M. Hence, going through the steps outlined in Problem 3.16, we readily see that in passing through the path from the input $f_M(n)$ to the output $b_M(n)$, we will have gone through all-pass filter of order M.

b)

In going from the input $f_M(n)$ to the output $u(n)$, we will have passed through the inverse of a forward prediction-error filter of order M. Since such a filter is minimum phase with all its zeros confined to the interior of the unit circle, it follows that its inverse is an all-pole filter with all its poles confined to the interior of the unit circle; hence its physical realizability is assured.

Problem 3.19

a)

The $(M+1)$-by-$(M+1)$ lower triangular matrix $\mathbf{L}$ is defined by

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ a_{1,1} & 1 & \dots & 0 \\ a_{2,1} & a_{1,1} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m-1,1} & \dots & 1 \end{bmatrix}$$

Let

$$\mathbf{Y} = \mathbf{LR}$$

where the $(M+1)$-by-$(M+1)$ correlation matrix $\mathbf{R}$ is defined by

$$\mathbf{R} = \begin{bmatrix} r(0) & r(1) & \dots & r(M) \\ r(-1) & r(0) & \dots & r(M-1) \\ \vdots & \vdots & \ddots & \vdots \\ r(-M) & r(-M+1) & \dots & r(0) \end{bmatrix}$$

Hence, the kmth element of the matrix product $\mathbf{LR}$ is given by

$$y_{km} = \sum_{\ell=0}^{k} a_{k,k-\ell} r(m-\ell), \qquad k = 0, 1, \dots, M \tag{1}$$

For $k = m$, we thus have

$$y_{mm} = \sum_{\ell=0}^{m} a_{m,m-\ell} r(m-\ell), \qquad m = 0, 1, \ldots, M$$

However, from the augmented Wiener-Hopf equation for backward prediction we have

$$\sum^{m} a^*_{m,m-\ell} r(\ell - i) = \begin{cases} 0, & i = 0, 1, \ldots, m-1 \\ P_m, & i = m \end{cases}$$

We therefore find that y_{mm} is real-valued and that

$$y_{mm} = P_m, \qquad m = 0, 1, \ldots, M.$$

b)

The mth column of matrix $\mathbf{Y}$ is given by

$$\begin{bmatrix} y_{0m} \\ y_{1m} \\ \vdots \\ y_{mm} \\ \vdots \\ y_{Mm} \end{bmatrix} = \begin{bmatrix} r(k) \\ \sum_{\ell=0}^{1} a_{1,1-\ell} r(m-\ell) \\ \vdots \\ \sum_{\ell=0}^{m} a_{m,m-\ell} r(m-\ell) \\ \mathbf{0} \end{bmatrix}$$

The kth element of this column vector equals y_{km} that is defined by Equation (1). The element y_{km} is recognized as the output produced by a backward prediction-error filter with tap weights $a^*_{k,k}, a^*_{k,k-1}, \ldots, a^*_{k,0}$ and correspondingly tap inputs $r(m), r(m-1), \ldots, r(m-k)$, respectively. By summing the inner products of the respective tap weights and tap inputs, we get y_{km}. Hence, the mth column of matrix $\mathbf{Y}$ is obtained by passing the autocorrelation sequence $\{r(0), r(1), \ldots, r(m)\}$ through the sequence of backward prediction-error filters whose transfer functions equal $H_{b,0}(z), H_{b,1}(z), \ldots, H_{b,m}(z)$.

c)

Apply the autocorrelation sequence $\{r(0), r(1), \ldots, r(m)\}$ to the input of a lattice predictor of order m. Denote the variables appearing at the various points on the lower part of the predictor as $x_0, x_1, \ldots, x_m$, as shown here:

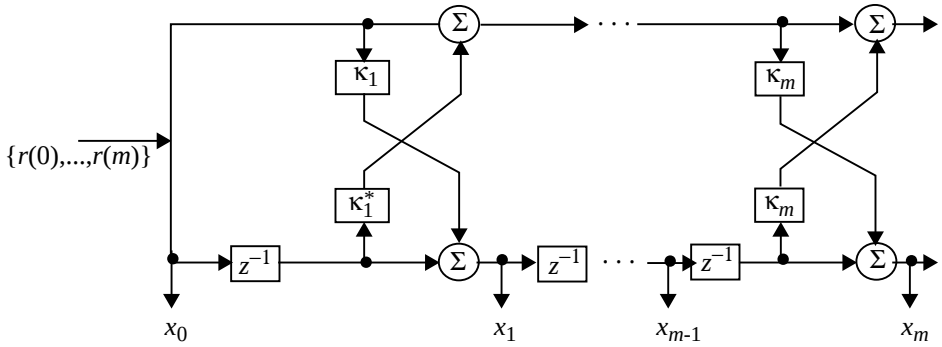


Figure 1

At time m we may express the resulting values of these outputs as follows

$$
\begin{aligned}
x_0 &= r(m) \\
x_1 &= \text{output of backward prediction-error filter of order 1} \\
&\quad \text{and with tap inputs } r(m-1), r(m) \\
&= \sum_{\ell=0}^{1} a_{1,1-\ell} r(m-\ell) \\
&\vdots \\
x_m &= \text{output of backward prediction-error filter of order} k \\
&\quad \text{and with tap inputs } r(0), r(1), \ldots, r(m) \\
&= \sum_{\ell=0}^{m} a_{m,m-\ell} r(m-\ell)
\end{aligned}
$$

The various sets of prediction-error filter coefficients are related to the reflection coefficients $\kappa_1, \kappa_2, \ldots, \kappa_m$ in accordance with the Levinson-Durbin recursion. Hence, the variables appearing at the various points on the lower line of the lattice predictor in Fig. 1 at time m equal the elements of the mth column of matrix $\mathbf{Y}$.

d)

The lower output of stage m at time m equals the mmth element y_{mm} of matrix $\mathbf{Y}$. This output equals P_m, as shown in part **a)**. The upper output of the stage m in the lattice predictor is equivalent ot the output of a forward prediction-error filter of order m. Hence, this output at time $m+1$, in response to the autocorrelation sequence $\{r(1), r(2), \ldots, r(m+1)\}$

used as input, equals

$$\sum_{\ell=0}^{m} a_{m,\ell}^{*} r(m+1-\ell) = \Delta_m^{*}$$

Thus, we deduce that (except for a minus sign) the ratio of the upper output of stage m in the lattice predictor of Figure 1 at time $m+1$ to the lower output of this stage at time m equals the complex conjugate of the reflection coefficient κ_{m+1} for stage $m+1$ in the lattice predictor.

e)
Using the autocorrelation sequence $\{r(0), r(1), \ldots, r(m)\}$ as inputs into the lattice predictor of Fig. 1, we may thus compute the corresponding sequence of reflection coefficients of the predictor as follows:

(i) At the input of the lattice predictor (i.e., $m = 0$), the upper output equals $r(1)$ at time 1, and the lower input equals $r(0)$ at time 0. Hence, the ratio of these two outputs equals $[r(1)/r(0)] = -\kappa_1^{*}$.

(ii) The upper output of stage 1 at time 2 equals Δ_1^{*}, and the lower output of this stage at time 1 equals P_1. Hence, the ratio of these two outputs equals $(\Delta_1^{*}/P_1) = -\kappa_2^{*}$.

(iii) The ratio of the upper output of stage 2 at time 3 to the lower output of this stage at time 2 equals $(\Delta_2^{*}/P_2) = -\kappa_3^{*}$, and so on for the higher stages of the lattice predictor.

Problem 3.20

A lattice filter exhibits some interesting correlation properties between the forward and backward prediction errors developed at the various stages of the filter. Basically, these properties are consequences of the principle of orthogonality, as described below:

Property 1. *The forward prediction error $f_m(n)$ and the input signal $u(n)$ are orthogonal:*

$$\mathbb{E}\left[f_m(n) u^{*}(n-k)\right] = 0, \qquad 1 \le k \le m \tag{1}$$

Similarly, the backward prediction error $b_m(n)$ and the input signal $u(n)$ are orthogonal:

$$\mathbb{E}\left[b_m(n)u^*(n-k)\right] = 0, \qquad 1 \leq k \leq m-1 \tag{2}$$

Note the difference between the ranges of the index k in Equations (1) and (2).

Equations (1) and (2) are both restatements of the principle of orthogonality. By definition, the forward prediction error $f_m(n)$ equals the difference between $u(n)$ and the prediction of $u(n)$, given the tap inputs $u(n-1), u(n-2), \ldots, u(n-m)$. By the principle of orthogonality, the error $f_m(n)$ is orthogonal to $u(n-k)$. $k = 1, 2, \ldots, m$. This proves Equation (1).

The backward prediction error, by definition, equals the difference between $u(n-m)$ and the prediction of $u(n-m)$, given the tap inputs $u(n), u(n-1) \ldots, u(n-m+1)$. Here, again, by the principle of orthogonality, the error $b_m(n)$ is orthogonal to $u(n-k)$, $k = 0, 1, \ldots, m-1$. This proves Equation (2)

Property 2. *The cross-correlation of the forward precition error $f_m(n)$ and the input $u(n)$ equals the cross-correlation of the backward prediction error $b_m(n)$ and the time-shifted input $u(n-m)$, as shown by*

$$\mathbb{E}\left[f_m(n)u^*(n)\right] = \mathbb{E}\left[b_m(n)u^*(n)\right] = P_m \tag{3}$$

where P_m is the corresponding prediction-error power.

To prove the first part of this property, we note that $u(n)$ equals the forward prediction error $f_m(n)$ plus the prediction of $u(n)$, given the samples $u(n-1), u(n-2), \ldots, u(n-m)$. Since this prediction is orthogonal to the error $f_m(n)$ [which is a corollary to the principle of orthogonality], it follows that

$$\begin{aligned}\mathbb{E}\left[f_m(n)u^*(n)\right] =& \mathbb{E}\left[f_m(n)f_m^*(n)\right] \\ =& P_m\end{aligned}$$

To prove the second part of the property, we note that $u(n-m)$ equals the backward prediction error $b_m(n)$ plus the prediction of $u(n-m)$, given the samples $u(n), u(n-1), \ldots, u(n-m+1)$. Since this prediction is orthogonal to the error $b_m(n)$, it follows that

$$\begin{aligned}\mathbb{E}\left[b_m(n)u^*(n-m)\right] =& \mathbb{E}\left[b_m(n)b_m^*(n)\right] \\ =& P_m\end{aligned}$$

This completes the proof of Equation (3).

Property 3. *The backward prediction errors are orthogonal to each other, as shown by*

$$\mathbb{E}\left[b_m(n)b_i^*(n)\right] = \begin{cases} P_m, & m = i \\ 0, & m \neq i \end{cases}$$

The forward prediction errors do not, however, exhibit the same orthogonality property as the backward prediction errors; rather they are correlated as shown by

$$\mathbb{E}\left[f_m(n)f_i^*(n)\right] = P_m \qquad m \geq i \tag{4}$$

Without loss of generality, we may assume that $m \geq i$. To prove Equation (4) we express the backward prediction error $b_i(n)$ in terms of the input $u(n)$ as the convolution sum

$$b_i(n) = \sum_{k=0}^{i} a_{i,i-k}u(n-k) \tag{5}$$

where $a_{i,i-k}^*$, $k = 0, 1, \ldots, i$, are the coefficients of a backward prediction-error filter of order i. Hence, we may write

$$\mathbb{E}\left[b_m(n)b_i^*(n)\right] = \mathbb{E}\left[b_m(n)\sum_{k=0}^{i} a_{i,i-k}^* u^*(n-k)\right] \tag{6}$$

Now, by property 1, we have

$$\mathbb{E}\left[b_m(n)b_i^*(n-k)\right] = 0 \qquad 0 \leq k \leq m-1$$

For the case when $m \geq i$, and with $0 \leq k \leq i$, we therefore find that all the expectation terms inside the summation on the right side of Equation (6) are zero. Correspondingly,

$$\mathbb{E}\left[b_m(n)b_i^*(n)\right] = 0 \qquad m \neq i$$

When $m = i$, Equation (6) reduces to

$$\begin{aligned}\mathbb{E}\left[b_m(n)b_i^*(n)\right] =& \mathbb{E}\left[b_m(n)b_m^*(n)\right] \\ =& P_m, \qquad m = i\end{aligned}$$

To Prove Equation (4), we express the forward prediction error $f_i(n)$ in terms of the input $u(n)$ as the convolution sum

$$f_i(n) = \sum_{k=0}^{i} a_{i,k}^* u(n-k) \tag{7}$$

where $a_{i,k}$, $k = 0, 1, \ldots, i$, are the coefficients of a forward prediction-error filter of order i. Hence,

$$\begin{aligned}\mathbb{E}\left[f_m(n)f_i^*(n)\right] =& \mathbb{E}\left[f_m(n)\sum_{k=0}^{i} a_{i,k}u^*(n-k)\right] \\ =& \mathbb{E}\left[f_m(n)u^*(n)\right] + \sum_{k=0}^{i} a_{i,k}\mathbb{E}\left[f_m(n)u^*(n-k)\right] \end{aligned} \tag{8}$$

where we have used the fact that $a_{i,0} = 1$. However, by Property 1, we have

$$\mathbb{E}\left[f_m(n)u^*(n-k)\right] = 0 \qquad 1 \le k \le m$$

Also, by Property 2, we have

$$\mathbb{E}\left[f_m(n)u^*(n)\right] = P_m$$

Therefore, Equation (8) reduces to

$$\mathbb{E}\left[f_m(n)f_i^*(n)\right] = P_m \qquad m \ge 1$$

This completes the proof of Equation (4).

Property 4. *The time-shifted versions of the forward and backward prediction errors are orthogonal, as shown by, respectivly,*

$$\mathbb{E}\left[f_m(n+\ell)f_i^*(n)\right] = \mathbb{E}\left[f_m(n-\ell)f_i^*(n)\right] = 0 \qquad \begin{array}{l}1 \le \ell \le m-i \\ m > i\end{array} \tag{9}$$

$$\mathbb{E}\left[b_m(n)b_i^*(n-\ell)\right] = \mathbb{E}\left[b_m(n+\ell)b_i^*(n)\right] = 0 \qquad \begin{array}{l}1 \le \ell \le m-i-1 \\ m > i\end{array} \tag{10}$$

where ℓ is an integer lag.

To prove Equation (9), we use Equation (7) to write

$$\begin{aligned}\mathbb{E}\left[f_m(n)f_i^*(n-\ell)\right] =& \mathbb{E}\left[f_m(n)\sum_{k=0}^{i} a_{i,k}u^*(n-\ell-k)\right] \\ =& \sum_{k=0}^{i} a_{i,k}\mathbb{E}\left[f_m(n)u^*(n-\ell-k)\right] \end{aligned} \tag{11}$$

By Property 1, we have

$$\mathbb{E}\left[f_m(n)u^*(n-\ell-k)\right]=0 \qquad 1\le \ell+k\le m \tag{12}$$

In the summation on the right side of Equation. (11) we have $0 \le k \le i$. For the orthogonality relationship of Equation (12) to hold for all values of k inside this range, the lag ℓ must correspondingly satisfy the conditions $1 \le \ell \le m-i$. Thus, with the lag ℓ bounded in this way, and with $m > i$, all the expectation terms inside the summation on the right side of Equation (11) are zero. We therefore have

$$\mathbb{E}\left[f_m(n)f_i^*(n-\ell)\right]=0 \qquad \begin{array}{l} 1\le \ell\le m-i \\ m>i \end{array}$$

By definition, we have

$$\mathbb{E}\left[f_m(n)f_i^*(n-\ell)\right]=\mathbb{E}\left[f_m(n+\ell)f_i^*(n)\right]$$

Therefore, if the expectation $\mathbb{E}\left[f_m(n)f_i^*(n-\ell)\right]$ is zero, then so is the expectation $\mathbb{E}\left[f_m(n+\ell)f_i^*(n)\right]$. This completes the proof of Equation (9)

To prove Equation (10), we use Equation (5) to write

$$\begin{aligned}\mathbb{E}\left[b_m(n)b_i^*(n-\ell)\right] =& \mathbb{E}\left[b_m\sum_{k=0}^{i}a_{i,i-k}^*u^*(n-\ell-k)\right] \\ =& \sum_{k=0}^{i}a_{i,i-k}^*\mathbb{E}\left[b_mu^*(n-\ell-k)\right]\end{aligned} \tag{13}$$

But, by Property 1, we have

$$\mathbb{E}\left[b_m(n)u^*(n-\ell-k)\right]=0 \qquad 0\le \ell+k\le m-1 \tag{14}$$

For the orthogonality relationship to hold for $0 \le k \le i$, the lag ℓ must satisfy the condition $0 \le \ell \le m-i-1$. Then, with $m > i$, we find that all the expectations inside the summation on the right side of Equation (13) are zero. We therefore have

$$\mathbb{E}\left[b_m(n)b_i^*(n-\ell)\right]=0 \qquad \begin{array}{l} 0\le \ell\le m-i-\ell \\ m>i \end{array}$$

By definition, we have

$$\mathbb{E}\left[b_m(n)b_i^*(n-\ell)\right]=\mathbb{E}\left[b_m(n+\ell)b_i^*(n)\right]$$

Hence, if the expectation $\mathbb{E}[b_m(n)b_i^*(n-\ell)]$ is zero, then so is the expectation $\mathbb{E}[b_m(n+\ell)b_i^*(n)]$. This completes the proof of Equation (10).

Property 5. *The time-shifted forward prediction errors $f_m(n+m)$ and $f_i(n+i)$ are orthogonal, as shown by*

$$\mathbb{E}[f_m(n-m)f_i^*(n+i)] = \begin{cases} P_m, & m = i \\ 0, & m \neq i \end{cases} \tag{15}$$

The corresponding time-shifted backward prediction errors $b_m(n+m)$ and $b_m(n+i)$, on the other hand are correlated as shown by

$$\mathbb{E}[b_m(n+m)b_i^*(n+i)] = P_m, \qquad m \geq i \tag{16}$$

Equations (15) and (16) are the duals of Equations (3) and (4), respectively

Without loss of generality, we may assume $m \geq i$. To prove Equation (15), we first recognize that

$$\begin{aligned} \mathbb{E}[f_m(n+m)f_i^*(n+i)] &= \mathbb{E}[f_m(n)f_i^*(n-m+i)] \\ &= \mathbb{E}[f_m(n)f_i^*(n-\ell)] \end{aligned} \tag{17}$$

where

$$\ell = m - i$$

Therefore, with $m > i$, we find from Property 4 that the expectation in Equation (17) is zero. When, however, $m = i$, the lag ℓ is zer, and this expectation equals P_m, the mean-square value of $f_m(n)$. This completes the proof of Equation (15).

To prove Equation (16) we recognize that

$$\begin{aligned} \mathbb{E}[b_m(n+m)b_i^*(n+i)] &= \mathbb{E}[b_m(n)b_i^*(n-m+i)] \\ &= \mathbb{E}[b_m(n)b_i^*(n-\ell)] \end{aligned} \tag{18}$$

where

$$\ell = m - i$$

The value of ℓ lies outside the range for which the expectation in Equation (18) is zero [see Equation (10)]. This means that $b_m(n+m)$ and $b_i(n+i)$ are correlated. To determine this

correlation, we use Equation (5) to write

$$\begin{aligned}\mathbb{E}\left[b_m(n+m)b_i^*(n+i)\right] =&\mathbb{E}\left[b_m(n+m)\sum_{k=0}^{i} a_{i,i-k}^* u^*(n+i-k)\right]\\ =&\mathbb{E}\left[b_m(n)u^*(n)\right]+\sum_{k=0}^{i-1} a_{i,i-k}^*\mathbb{E}\left[b_m(n+m)u^*(n+i-k)\right]\end{aligned} \tag{19}$$

where we have used $a_{i,0}=1$. By Property 1, we have

$$\begin{aligned}\mathbb{E}\left[b_m(n+m)u^*(n+i-k)\right] =&\mathbb{E}\left[b_m(n)u^*(n+i-k-m)\right]\\ =&0, \qquad 0\le k+m-i\le m-1\end{aligned} \tag{20}$$

The orthogonality relationship of Equation (2) holds for $i-m\le k\le i-1$. The summation on the right side of Equation (19) applies for $0\le k\le i-1$. Hence, with $m\ge i$, all the expectation terms inside the summation are zero. Correspondingly, Equation (19) reduces to

$$\begin{aligned}\mathbb{E}\left[b_m(n+m)b_i^*(n+i)\right] =&\mathbb{E}\left[b_m(n+m)u^*(n)\right]\\ =&\mathbb{E}\left[b_m(n)u^*(n-m)\right]\\ =&P_m, \qquad m\ge i\end{aligned}$$

where we have made use of Property 2. This completes the proof of Equation (16)

Property 6. *The forward and backward prediction erros exhibit the following crosscorrelation property:*

$$\mathbb{E}\left[f_m(n)b_i^*(n)\right]=\begin{cases}\kappa_i^* P_m, & m\ge i\\ 0, & m<i\end{cases} \tag{21}$$

To prove this property, we use Equation (5) to write

$$\begin{aligned}\mathbb{E}\left[f_m(n)b_i^*(n)\right] =&\mathbb{E}\left[f_m(n)\sum_{k=0}^{i} a_{i,i-k}^* u^*(n-k)\right]\\ =&a_{i,i}^*\mathbb{E}\left[f_m(n)u^*(n)\right]+\sum_{k=1}^{i} a_{i,i-k}^*\mathbb{E}\left[f_m(n)u^*(n-k)\right]\end{aligned} \tag{22}$$

By Property 1, we have

$$\mathbb{E}\left[f_m(n)u^*(n-k)\right] = 0, \qquad 1 \leq k \leq m$$

Assuming that $m \geq 1$, we therefore find that all the expectation terms in the second term (summation) on the right side of Equation (22) are zero. Hence,

$$\sum_{k=1}^{i} a_{i,i-k}^* \mathbb{E}\left[f_m(n)u^*(n-k)\right] = 0 \qquad \text{for } m \geq i$$

By Property 2, we have

$$\mathbb{E}\left[f_m(n)u^*(n)\right] = P_m$$

Therefore, with $a_{i,i} = \kappa_i$, we find that Equation (22) reduces to

$$\mathbb{E}\left[f_m(n)b_i^*(n)\right] = \kappa_i^* P_m, \qquad m \geq i$$

For the case when $m < i$, we adapt Equation (7) to write

$$\begin{aligned}\mathbb{E}\left[f_m(n)b_i^*(n)\right] =& \mathbb{E}\left[\sum_{k=0}^{m} a_{m,k}^* u(n-k)b_i^*(n)\right] \\ =& \sum_{k=0}^{m} a_{m,k}^* \mathbb{E}\left[u(n-k)b_i^*(n)\right] \end{aligned} \tag{23}$$

By Property 1, we have

$$\mathbb{E}\left[b_i(n)u^*(n-k)\right] = 0, \qquad 0 \leq k \leq i-1$$

Therefore, with $m < i$, we find that all the expectations terms inside the summation on the right side of Equation (23) are zero. Thus,

$$\mathbb{E}\left[f_m(n)b_i^*(n)\right] = 0 \qquad m < i$$

This completes the proof of Property 6.

Problem 3.21

The entropy of the input vector $\mathbf{u}(n)$ is defined by the multiple integral

$$H_u = -\int_{-\infty}^{\infty} f_U(\mathbf{u}) \ln\left[f_U(\mathbf{u})\right] \mathrm{d}\,\mathbf{u} \tag{1}$$

But,

$$\mathbf{u}(n) = \mathbf{L}^{-1}\mathbf{b}(n) \tag{2}$$

where $\mathbf{L}$ is a lower triangular matrix defined by a sequence of backward prediction-error filters. We may therefore express the joint probability density function $f_B(\mathbf{b})$ of the backward prediction-error vector $\mathbf{b}(n)$ in terms of $f_U(\mathbf{u})$ as

$$f_B(\mathbf{b}) = \left|\det\left(\mathbf{L}^{-1}\right)\right| f_U\left(\mathbf{L}^{-1}\mathbf{b}\right) \tag{3}$$

where the term inside the absolute value signs is the Jacobian of the transformation described in Equation (2). We also note:

$$\mathrm{d}(\mathbf{u}) = \left|\det\left(\mathbf{L}^{-1}\right)\right| \mathrm{d}\,\mathbf{b} \tag{4}$$

Next, we note that the entropy of the backward prediction-error vector $\mathbf{b}(n)$ is

$$\begin{aligned} H_b &= -\int_{-\infty}^{\infty} f_B(\mathbf{b}) \ln\left[f_B(\mathbf{b})\right] \mathrm{d}\,\mathbf{b} \\ &= -\int_{-\infty}^{\infty} \left|\det\left(\mathbf{L}^{-1}\right)\right| f_U\left(\mathbf{L}^{-1}\mathbf{b}\right) \ln\left[\left|\det\left(\mathbf{L}^{-1}\right)\right| f_U\left(\mathbf{L}^{-1}\mathbf{b}\right)\right] \mathrm{d}\,\mathbf{b} \\ &= -\int_{-\infty}^{\infty} f_U(\mathbf{u}) \ln\left[\left|\det\left(\mathbf{L}^{-1}\right)\right| f_U(\mathbf{u})\right] \mathrm{d}\,\mathbf{u} \end{aligned}$$

where in the second line we have used Equation (3) and in the third line we have used Equations (2) and (4). The transformation matrix $\mathbf{L}$ has unity for all its diagonal elements; hence, the Jacobian of the transformation is unity. Accordingly, we may simplify the entropy H_b as

$$\begin{aligned} H_b &= -\int_{-\infty}^{\infty} f_U(\mathbf{u}) \ln\left[f_U(\mathbf{u})\right] \mathrm{d}\,\mathbf{u} \\ &= H_u \end{aligned}$$

This shows that the backward prediction-error vector $\mathbf{b}(n)$ has the same entropy, and therefore contains the same amount of information, as the input vector $\mathbf{u}(n)$.

Problem 3.22

a)

The index of performance to be optimized for stage m is given by

$$J_m = a\mathbb{E}\left[|f_m(n)|^2\right] + (1-a)\mathbb{E}\left[|b_m(n)|^2\right] \tag{1}$$

where a is a constant that determines the mix of contributions from the forward and backward prediction errors of the index of performance. The forward prediction error is

$$f_m(n) = f_{m-1}(n) + \kappa_m^* b_{m-1}(n-1) \tag{2}$$

and the backward prediction error is

$$b_m(n) = b_{m-1}(n) + \kappa_m f_{m-1}(n-1) \tag{3}$$

Substituting Equations (2) and (3) in (1) yields

$$\begin{aligned} J_m =& a\mathbb{E}\left[|f_{m-1}(n)|^2\right] + (1-a)\,\mathbb{E}\left[|b_{m-1}(n-1)|^2\right] \\ &+ |\kappa_m|^2 \left\{(1-a)\,\mathbb{E}\left[|f_{m-1}(n)|^2\right] + a\mathbb{E}\left[|b_{m-1}(n-1)|^2\right]\right\} \\ &+ 2\kappa_m^*\mathbb{E}\left[f_{m-1}^*(n)b_{m-1}(n-1)\right] + 2\kappa\mathbb{E}\left[f_{m-1}(n)b_{m-1}^*(n-1)\right] \end{aligned}$$

Hence, differentiating J_m with respect to κ_m and setting the result equal to zero, we find that the optimum value of κ_m is given by

$$\kappa_{m,0}(a) = -\frac{\mathbb{E}\left[f_{m-1}^*(n)b_{m-1}(n-1)\right]}{(1-a)\mathbb{E}\left[|f_{m-1}(n)|^2\right] + a\mathbb{E}\left[|b_{m-1}(n-1)|^2\right]}$$

b)

The three special cases of interest are

i) $a = 1$

The index of performance takes the first form:

$$J_m = \mathbb{E}\left[|f_m(n)|^2\right]$$

Correspondingly, the optimum value of the reflection coefficients is given by

$$\kappa_{m,0}(1) = -\frac{\mathbb{E}\left[f_{m-1}^*(n)b_{m-1}(n-1)\right]}{\mathbb{E}\left[|b_{m-1}(n-1)|^2\right]}$$

We refer to this method of optimization as the *forward method.*

ii) $a = 0$

The index of performance takes the second form:

$$J_m = \mathbb{E}\left[|b_{m-1}(n)|^2\right]$$

Correspondingly, the optimum value of the reflection coefficients is given by

$$\kappa_{m,0}(1) = -\frac{\mathbb{E}\left[f_{m-1}^*(n)b_{m-1}(n-1)\right]}{\mathbb{E}\left[|f_{m-1}(n)|^2\right]}$$

We refer to this method of optimization as the *backward method.*

iii) $a = 1/2$

The index of performance takes the third form:

$$J_m = \frac{1}{2}\left\{\mathbb{E}\left[|f_m(n)|^2\right] + \mathbb{E}\left[|b_{m-1}(n)|^2\right]\right\}$$

Correspondingly, the optimum reflection coefficients yields the Burg formula:

$$\kappa_{m,0}(\frac{1}{2}) = -\frac{2\mathbb{E}\left[f_{m-1}^*(n)b_{m-1}(n-1)\right]}{\mathbb{E}\left[\left|f_{m-1}^*(n)\right|^2\right] + \mathbb{E}\left[|b_{m-1}(n-1)|^2\right]}$$

We refer to this method of optimization as the *forward-backward method.*

Problem 3.23

a)

The harmonic mean of the optimum values of the reflection coefficient produced by the forward and backward methods is defined by

$$\begin{aligned}\frac{1}{2}\left[\frac{1}{\kappa_{m,0}(1)} + \frac{1}{\kappa_{m,0}(0)}\right] &= -\frac{\mathbb{E}\left[|f_{m-1}(n)|^2\right] + \mathbb{E}\left[|b_{m-1}(n-1)|^2\right]}{2\mathbb{E}\left[f_{m-1}^*(n)b_{m-1}(n-1)\right]} \\ &= \frac{1}{\kappa_{m,0^{(1/2)}}}\end{aligned}$$

where $\kappa_{m,0}$ is the optimum value of the reflection coefficient produced by the Burg formula.

b)

Define the correlation coefficient or the normalized value of the cross-correlation function

between the forward prediction error $f_{m-1}(n)$ and the delayed backward prediction error $b_{m-1}(n-1)$ as

$$\rho = \frac{\mathbb{E}\left[f_{m-1}^{*}(n)b_{m-1}(n-1)\right]}{\sqrt{\mathbb{E}\left[\left|f_{m-1}^{*}(n)\right|^2\right] + \mathbb{E}\left[\left|b_{m-1}(n-1)\right|^2\right]}} \tag{1}$$

Hence, we may redefine the optimum value $\kappa_{m,0}(1)$ and $\kappa_{m,0}(0)$ for the reflection coefficient that are produced by the forward and backward methods, respectively, as follows

$$\kappa_{m,0}(1) = -\alpha\rho$$

and

$$\kappa_{m,0}(0) = -\frac{\rho}{\alpha}$$

where the parameter α is itself defined by

$$\alpha = \sqrt{\frac{\mathbb{E}\left[\left|f_{m-1}(n)\right|^2\right]}{\mathbb{E}\left[\left|b_{m-1}(n-1)\right|^2\right]}}$$

Accordingly, using the result of part (a), we may redefine the optimum value of the reflection coefficient produced by the Burg formula as

$$\frac{1}{\kappa_{m,0}} = -\frac{1}{2\rho}\left(\frac{1}{\alpha} + \alpha\right)$$

We note that the parameter α is a nonnegative real-valued scalar that lies in the range $0 \leq \alpha \leq \infty$. Moreover, the factor

$$\frac{1}{\alpha} + \alpha$$

attains its minimum value of 2 at $\alpha = 1$. We may therefore write

$$\frac{1}{\alpha} + \alpha \geq 2$$

The correlation coefficient ρ is likewise a nonnegative scalar that lies in the range $0 \leq \rho \leq 1$. That is, $1 \leq (1/\rho) \leq \infty$. We conclude therefore that

$$\frac{1}{|\kappa_{m,0}|} \geq 1$$

or, equivalently,

$$|\kappa_{m,0}| \leq 1 \text{ for all } m.$$

c)

The mean-square value of the forward prediction error $f_m(n)$ is $\mathbb{E}\left[|f_m(n)|^2\right]$, where

$$f_m(n) = f_{m-1}(n) + \kappa_{m,0}^* b_{m-1}(n-1) \tag{2}$$

where the reflection coefficient is assigned the optimum value $\kappa_{m,0}$ in accordance with the Burg formula. Based on (2), we write

$$\begin{aligned}\mathbb{E}\left[|f_m(n)|^2\right] =& \mathbb{E}\left[|f_{m-1}(n)|^2\right] + |\kappa_{m,0}|^2 \mathbb{E}\left[|b_{m-1}(n-1)|^2\right] \\ &+ \kappa_{m,0}^* \mathbb{E}\left[f_{m-1}^*(n) b_{m-1}(n-1)\right] \\ &+ \kappa_{m,0} \mathbb{E}\left[f_{m-1}(n) b_{m-1}^*(n-1)\right]\end{aligned}$$

Using the Burg formula

$$\kappa_{m,0} = -\frac{2\mathbb{E}\left[f_{m-1}^*(n) b_{m-1}(n-1)\right]}{\mathbb{E}\left[|f_{m-1}(n)|^2\right] + \mathbb{E}\left[|b_{m-1}(n-1)|^2\right]}$$

To eliminate the factor $\mathbb{E}\left[f_{m-1}^*(n) b_{m-1}(n-1)\right]$, we may thus rewrite the mean-square value of $f_m(n)$ as follows:

$$\begin{aligned}\mathbb{E}\left[|f_m(n)|^2\right] &= \mathbb{E}\left[|f_{m-1}(n)|^2\right] + |\kappa_{m,0}|^2 \mathbb{E}\left[|b_{m-1}(n-1)|^2\right] \\ &\quad - |\kappa_{m,0}|^2 \left\{\mathbb{E}\left[|f_{m-1}(n)|^2\right] + \mathbb{E}\left[|b_{m-1}(n)|^2\right]\right\} \\ &= \left(1 - |\kappa_{m,0}|^2\right) \mathbb{E}\left[|f_{m-1}(n)|^2\right]\end{aligned} \tag{3}$$

Similarly, we may express the mean-square value of the backward prediction error $b_{m-1}(n)$ as follows

$$\begin{aligned}\mathbb{E}\left[|b_{m-1}(n)|^2\right] =& \mathbb{E}\left[|b_{m-1}(n-1)|^2\right] + |\kappa_{m,0}|^2 \mathbb{E}\left[|f_{m-1}(n)|^2\right] \\ &+ \kappa_{m,0} \mathbb{E}\left[f_{m-1}(n) b_{m-1}^*(n-1)\right] \\ &+ \kappa_{m,0}^* \mathbb{E}\left[f_{m-1}^*(n) b_{m-1}(n-1)\right] \\ =& \mathbb{E}\left[|b_{m-1}(n-1)|^2\right] + |\kappa_{m,0}|^2 \mathbb{E}\left[|f_{m-1}(n)|^2\right] \\ &- |\kappa_{m,0}|^2 \left\{\mathbb{E}\left[|f_{m-1}(n)|^2\right] + \mathbb{E}\left[|b_{m-1}(n-1)|^2\right]\right\} \\ =& \left(1 - |\kappa_{m,0}|^2\right) \mathbb{E}\left[|b_{m-1}(n-1)|^2\right]\end{aligned} \tag{4}$$

Problem 3.24

Assuming ergodicity, we may set up the following table for time series of length N.

Ensemble average	Time-average
$\dfrac{\mathbb{E}\left[b_{m-1}(n-1)f^*_{m-1}(n)\right]}{\mathbb{E}\left[\lvert b_{m-1}(n-1)\rvert^2\right]}$	$\dfrac{\sum_{n=1}^{N} b_{m-1}(n-1)f^*_{m-1}(n)}{\sum_{n=1}^{N}\lvert b_{m-1}(n-1)\rvert^2}$
$\dfrac{\mathbb{E}\left[b_{m-1}(n-1)f^*_{m-1}(n)\right]}{\mathbb{E}\left[\lvert f_{m-1}(n)\rvert^2\right]}$	$\dfrac{\sum_{n=1}^{N} b_{m-1}(n-1)f^*_{m-1}(n)}{\sum_{n=1}^{N}\lvert f_{m-1}(n)\rvert^2}$
$\dfrac{\mathbb{E}\left[b_{m-1}(n-1)f^*_{m-1}(n)\right]}{\left(\mathbb{E}\left[\lvert f_{m-1}(n)\rvert^2\right]\mathbb{E}\left[\lvert b_{m-1}(n-1)\rvert^2\right]\right)^{-1}}$	$\dfrac{\sum_{n=1}^{N} b_{m-1}(n-1)f^*_{m-1}(n)}{\left(\sum_{n=1}^{N}\lvert f_{m-1}(n)\rvert^2\sum_{n=1}^{N}\lvert b_{m-1}(n-1)\rvert^2\right)}$

Applying the Cauchy-Schwartz inequality, we now see that only the last entry in the table assures the condition $|\kappa_m| \leq 1$ for all m:

$$\left|\sum_{n=1}^{N} b_{m-1}(n-1)f^*_{m-1}(n)\right|^2 \leq \left(\sum_{n=1}^{N}\lvert f_{m-1}(n)\rvert^2\right)\left(\sum_{n=1}^{N}\lvert b_{m-1}(n-1)\rvert^2\right)$$

Problem 3.25

a)

$$G(z) = \frac{z^2}{\left(1 - \frac{z}{0.4}\right)\left(1 + \frac{z}{0.8}\right)}$$

$$= \frac{-0.4 \times 0.8}{\left(1 - \frac{0.4}{z}\right)\left(1 + \frac{0.8}{z}\right)}$$

We may therefore set

$$a_{21} = 0.4, \qquad a_{22} = -0.32$$

$$\begin{aligned} a_1 &= \frac{a_{21} - a_{22} \cdot a_{21}}{1 - a_{22}^2} \\ &= \frac{0.4 + 0.32 \cdot 0.4}{1 - 0.32^2} \\ &= 0.5882 \end{aligned}$$

The reflection coefficients are therefore

$$\kappa_2 = -0.32$$

$$\kappa_1 = 0.5882$$

b)

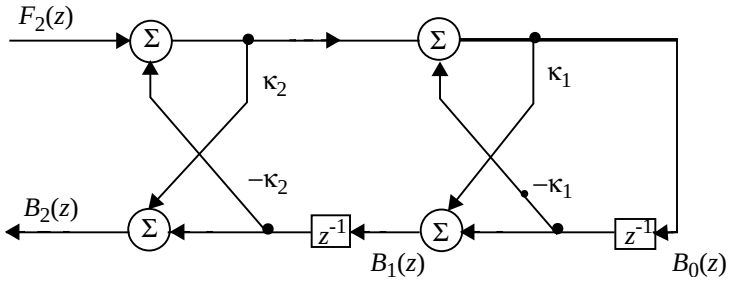


Figure 1

$$B_2(z) = \left[-\kappa_2 B_1(z) z^{-1} + F_2(z)\right] \kappa_2 + B_1(z) z^{-1} \tag{1}$$

$$\frac{B_0(z)}{F_2(z)} = \frac{1}{1 + a_{21} z^{-1} + a_{22} z^{-2}}$$

$$B_1(z) = B_0(z) z^{-1} + B_0(z) \kappa_1$$

$$\frac{B_1(z)}{F_2(z)} = \frac{\kappa_1 + z^{-1}}{1 + a_{21} z^{-1} + a_{22} z^{-2}}$$

$$\begin{aligned}
\frac{B_2(z)}{F_2(z)} &= \frac{B_1(z)}{F_2(z)} \left[z^{-1} - \kappa_2^2 z^{-1}\right] + \kappa_2 \\
&= \frac{\left[\kappa_1 + z^{-1}\right]\left[z^{-1} - \kappa_2^2 z^{-1}\right]}{1 + a_{21} z^{-1} + a_{22} z^{-2}} + \kappa_2 \\
&= \frac{\kappa_1 z^{-1} + \kappa_1 \kappa_2 z^{-1} + \kappa_2 + z^{-2}}{1 + \left(\kappa_1 + \kappa_1 + \kappa_2\right) z^{-1} + \kappa_2 z^{-2}} \\
&= \frac{z^{-2} + 0.4 z^{-1} - 0.32}{1 + 0.4 z^{-1} - 0.32 z^{-2}}
\end{aligned}$$

Therefore, the all pass transfer function realized by the filter of Figure 1 is given by

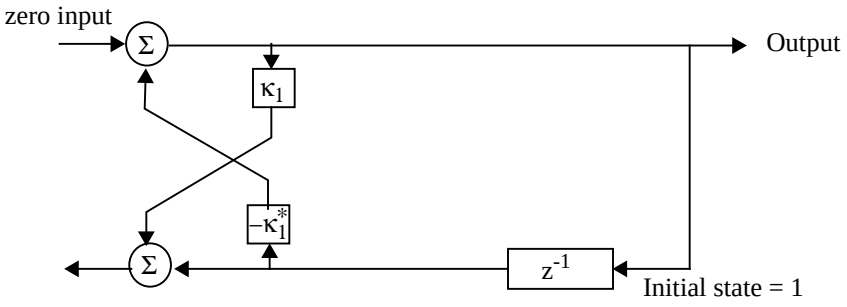
$$\frac{z^{-2} + 0.4 z^{-1} - 0.32}{1 + 0.4 z^{-1} - 0.32 z^{-2}}$$

Problem 3.26

a)

Prediction order M=1

The inverse lattice filter consists of a single stage. By initializing the single stage of the filter to 1 and operating it with zero input, we have the following configuration to consider:

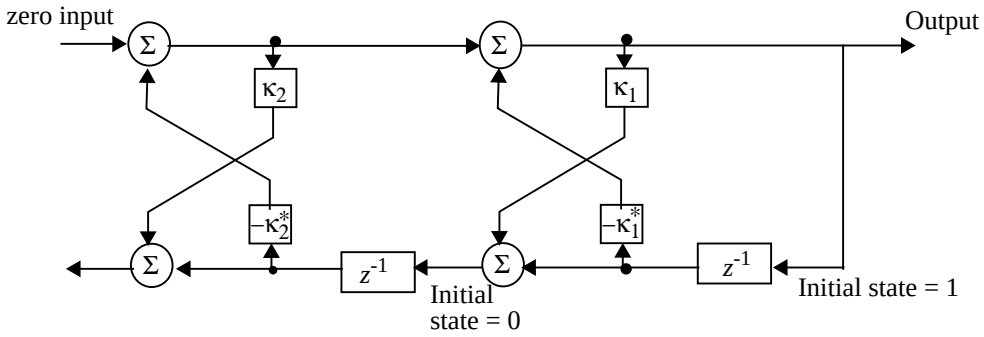


$$\text{Output for lag of one } = -\kappa_1^* = \frac{r(1)}{r(0)}$$

b)

Prediction order M=2

In this case the inverse lattice filter consists of two stages. By initializing the two states of the filter to 1, 0 and operating it with zero input, we have the following configuration to consider:



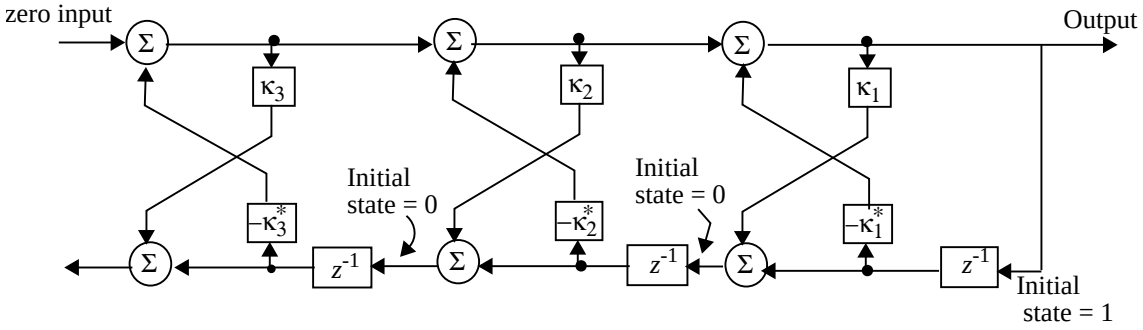
$$\text{Output for lag of one } = -\kappa_1^* = \frac{r(1)}{r(0)}$$

$$\text{Output for lag of two } = (-\kappa_2^*) + (-\kappa_1^*)(\kappa_1)(-\kappa_2^*) + (-\kappa_1^*)(-\kappa_1^*) = \frac{r(2)}{r(0)}$$

c)

Prediction order M=3

In this case, the inverse lattice filter consists of three stages. By initializing the three state of the filter to 1, 0, 0 and operating it with zero input, we have the following configuration to consider:



$$\begin{aligned}\text{Output for lag of one } &= -\kappa_1^* \\ &= \frac{r(1)}{r(0)}\end{aligned}$$

$$\begin{aligned}\text{Output for lag of two } &= (-\kappa_2^*) + (-\kappa_1^*)(\kappa_1)(-\kappa_2^*) + (-\kappa_1^*)(-\kappa_1^*) \\ &= \frac{r(2)}{r(0)}\end{aligned}$$

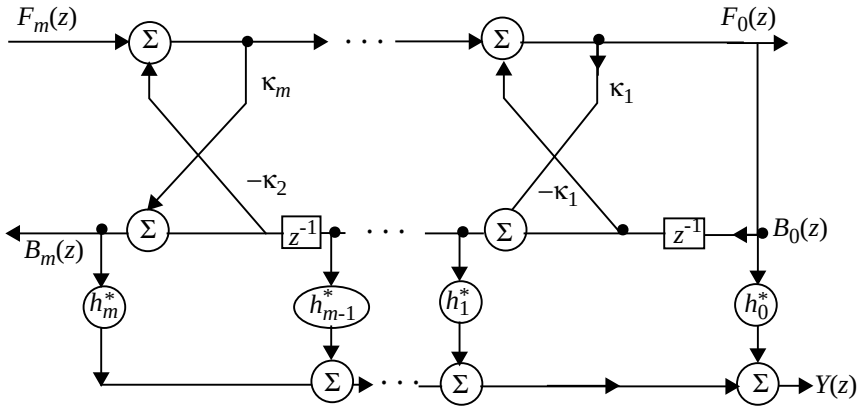
$$\begin{aligned}\text{Output for lag of three } &= (-\kappa_3^*) + (-\kappa_2^*)(\kappa_2)(-\kappa_3^*) + (-\kappa_1^*)(\kappa_1)(-\kappa_3^*) \\ &\quad + (-\kappa_1^*)(\kappa_1)(-\kappa_2^*)(\kappa_2)(-\kappa_3^*) + (-(\kappa_1^*))(-\kappa_2^*) \\ &\quad + (-\kappa_2^*)(\kappa_1^*) + (-\kappa_2^*)(\kappa_1)(-\kappa_2^*) \\ &\quad + (-\kappa_1^*)(-\kappa_1^*)(\kappa_1)(-\kappa_2^*) \\ &\quad + (-\kappa_1^*)(\kappa_1)(-\kappa_2^*)(\kappa_1)(-\kappa_2^*) \\ &\quad + (-\kappa_1^*)(-\kappa_1^*)(-\kappa_1^*) + (-\kappa_2^*)(-\kappa_1^*) \\ &\quad + (-\kappa_1^*)(-\kappa_1^*)(\kappa_1)(-\kappa_2^*) \\ &= \frac{r(3)}{r(0)}\end{aligned}$$

Note that a lag of one corresponds to signal flow through a single delay element z^{-1}, a lag of two corresponds to signal flow through two delay elements, and a lag of three corresponds to signal flow through three delay elements, and so on.

Problem 3.27

Backward prediction errors are orthogonal to each other. On the other hand, forward prediction errors are correlated, which follows from Property 3 discussed in Problem 3.20.

Problem 3.28



$$
\begin{aligned}
\frac{Y(z)}{F_m(z)} =& h_0\frac{B_0(z)}{F_m(z)} + h_1\frac{B_1(z)}{F_m(z)} + \ldots + h_m\frac{B_m(z)}{F_m(z)} \\
=& \frac{h_0 + h_1\left[a_{11} + a_{10}z^{-1}\right] + \ldots + h_m\left[a_{mm} + \ldots a_{m0}z^{-m}\right]}{1 + a_{m1}z^{-1} + \ldots + a_{mm}z^{-m}}, \quad a_{m0} = 1 \\
=& \frac{\left(h_0 + h_1a_{11} + \ldots h_ma_{mm}\right) + \left(h_1a_{10} + h_2a_{21} + \ldots h_ma_{mm-1}\right)z^{-1} + \ldots + h_mz^{-m}}{1 + a_{m1}z^{-1} + \ldots + a_{mm}z^{-m}} \\
=& \frac{P_0 + P_1z^{-1} + \ldots P_mz^{-m}}{1 + d_1z^{-1} + \ldots d_mz^{-m}}, \qquad d_i = a_{mi} \\
=& P_0\frac{\left[1 + (P_1/P_0)z^{-1} + \ldots + (P_m/P_0)z^{-m}\right]}{1 + d_1z^{-1} + \ldots d_mz^{-m}} \qquad (1)
\end{aligned}
$$

$$
\begin{aligned}
=& G_0\frac{\prod_{i=1}^{m}(1 - z/z_i)}{\prod_{i=1}^{m}(1 - z/z_p)} \\
=& \frac{G_0p_1\ldots p_m}{z_1\ldots z_m}\frac{\prod_{i=1}^{m}(1 - z_i/z)}{\prod_{i=1}^{m}(1 - z_p/z)} \qquad (2)
\end{aligned}
$$

$$= \frac{G_0 p_1 \dots p_m}{z_1 \dots z_m} \times \left[\frac{1 + (-1)^1 \sum_{i=1}^{m} z_i z^{-1} + (-1)^2 \sum_{i=1}^{m} \sum_{\substack{k=1 \\ i \neq k}}^{m} z_i z_k z^{-2} + \dots + z^{-m} (-1)^m z_1 \dots z_m}{1 + (-1)^1 \sum_{i=1}^{m} p_i z^{-1} + (-1)^2 \sum_{i=1}^{m} \sum_{\substack{k=1 \\ i \neq k}}^{m} p_i p_k z^{-2} + \dots + z^{-m} (-1)^m p_1 \dots p_m} \right]$$

a)

By equating the denominators of Equations (1) and (2):

$$1 + a_{m1} z^{-1} + \dots a_{mm} z^{-m} = \prod_{i=1}^{m} (1 - p_i / z)$$

$$= 1 + (-1)^1 \sum_{i=1}^{m} p_i z^{-1} + \sum_{\substack{i=1 \\ i \neq k}}^{m} \sum_{k=1}^{m} p_1 p_k z^{-2} + \dots + (-1)^m p_1 \dots p_m z^{-m}$$

By equating the coefficients of z^{-i}, $a_{m,i}$ can be calculated for $i = 1 \dots m$. Then by using the equation

$$a_{m-1,k} = \frac{a_{mk} - a_{mm} a^*_{m,m-k}}{1 - |a_m m|^2}, \qquad k = 0, 1, \dots, m$$

$a_{i,i}, \qquad i = 1, \dots, m$ can be calculated.

Reflection coefficients $k_i - a_{ii} \qquad i = 1, \dots, m$

b)

By equating the numerator of Equations (1) and (2):

$$h_0 + h_1 a_{11} + \dots h_m a_{mm} + \dots + h_m z^{-m} = \frac{G_0 p_1 \dots p_m}{z_1 \dots z_m} \prod_{i=1}^{m} (1 - z_1 / z)$$

$$= \frac{G_0 p_1 \dots p_m}{z_1 \dots z_m} \left[1 + (-1)^1 \sum_{i=1}^{m} z_i z^{-1} + \dots (-1)^m z_1 \dots z_m z^{-m} \right]$$

By equating the coefficients of z^{-i}, from z^{-m} to z^{-1}, the regression coefficients $h_m, \dots, h_0$ can be calculated

$$h_0 + h_1 a_{11} + \dots h_m a_{m1} = \frac{G_0 p_1 \dots p_m}{z_1 \dots z_m}$$

Coefficient of z^{-m} is

$$\begin{aligned} h_m =& \frac{G_0 p_1 \dots p_m}{z_1 \dots z_m}(-1)^m z_1 \dots z_m \\ =& G_0 p_1 \dots p_m \end{aligned}$$

By equating the coefficients of z^{-m+1}

$$h_{m-1}a_{m-1,0} + h_{m-1}a_{m-1} = \frac{G_0 p_1 \dots p_m}{z_1 \dots z_m} \left\{\text{coefficients of } z^{-m+1}\right\}$$

By back substitution, h_{m-1} can be calculated in the same way as $h_{m-2}, \dots, h_1, h_0$ can be calculated.

c)

The lattice filter of Fig. 3.3 is configured for an all-pole filter with no restrictions imposed on the zeros. As such, this lattice filter is an example of a minimum phase filter, and therefore includes the all pole, phase filter of Fig 3.10 as a special case. In other words the lattice structure of Fig p 3.3 is applicable to a non-minimum phase transfer function provided the conditions $z_i^* = p_i$ is satisfied for all i.

Problem 3.29

a)

$$\begin{aligned} G(z) =& 10\frac{\left(1+\frac{z}{0.1}\right)\left(1+\frac{z}{0.6}\right)}{\left(1-\frac{z}{0.4}\right)\left(1+\frac{z}{0.8}\right)} \\ =& \frac{10 \times 0.4 \times 0.8}{0.1 \times 0.6}\frac{\left(\frac{0.1}{z}+1\right)\left(\frac{0.6}{z}+1\right)}{\left(\frac{0.4}{z}-1\right)\left(1+\frac{0.8}{z}\right)} \\ =& -\frac{32}{6} \times 10\frac{\left(\frac{0.1}{z}+1\right)\left(\frac{0.6}{z}+1\right)}{\left(1-\frac{0.4}{z}\right)\left(1+\frac{0.8}{z}\right)} \\ =& -\frac{32}{6} \times 10\frac{[1+0.7z^{-1}+0.006z^{-2}]}{[1+0.4z^{-1}-0.32z^{-2}]} \end{aligned}$$

Therefore, $a_{21} = 0.4, \qquad a_{22} = -0.32$

$$\begin{aligned} a_{11} =& \frac{a_{21} - a_{22}a_{21}}{1 - a_{22}^2} \\ =& \frac{0.4 + 0.32 \times 0.4}{1 - 0.32^2} \\ =& \frac{0.528}{0.8976} \\ =& 0.5882 \end{aligned}$$

Hence, the reflection coefficients are

$$\kappa_1 = 0.5882$$

and

$$\kappa_2 = -0.32$$

The regression coefficients are calculated as follows:

$$\begin{aligned} h_2 =& \frac{-32}{6} \times 10 \times 0.06 \\ =& -3.2 \end{aligned}$$

$$h_1 a_{10} + h_2 a_{21} = 0.7 \times \frac{-32}{6}$$

Therefore $h_1 = -34.13$

$$h_0 + h_1 a_{11} + h_2 a_{22} = \frac{-32}{6} \times 10$$

Therefore $h_0 = -34.28$

b)

$$\begin{aligned} G(z) =& \frac{10\,(1 + z + z^2)}{\left(1 - \dfrac{z}{0.4}\right)\left(1 + \dfrac{z}{0.8}\right)} \\ =& -(10 \times 0.4 \times 0.8)\,\frac{[1 + z + z^2]}{\left(1 - \dfrac{0.4}{z}\right)\left(1 + \dfrac{0.8}{z}\right)} \end{aligned}$$

From section **a)**

$$a_{21} = 0.4, \qquad a_{22} = -0.32$$

$$a_{11} = 0.5882, \qquad \kappa_1 = 0.5882, \qquad \kappa_2 = -0.32$$

$$\begin{aligned} h_2 &= -10 \times 0.4 \times 0.8 \\ &= -3.2 \end{aligned}$$

$$\begin{aligned} h_1 a_{10} + h_2 a_{21} &= -10 \times 0.4 \times 0.8 \\ h_1 &= -1.92 \end{aligned}$$

$$\begin{aligned} h_0 + h_1 a_{11} + h_2 a_{22} &= -10 \times 0.4 \times 0.8 \\ h_0 &= -3.095 \end{aligned}$$

c)

$$\begin{aligned} G(z) &= \frac{10\left(1+\frac{z}{0.6}\right)\left(1+\frac{z}{1.5}\right)}{\left(1-\frac{z}{0.4}\right)\left(1+\frac{z}{0.8}\right)} \\ &= -\left(\frac{10 \times 0.4 \times 0.8}{0.6 \times 1.5}\right) \frac{\left(\frac{0.6}{z}+1\right)\left(\frac{1.5}{z}+1\right)}{\left(1-\frac{0.4}{z}\right)\left(1+\frac{0.8}{z}\right)} \end{aligned}$$

From section **a)**

$$a_{21} = 0.4, \qquad a_{22} = -0.32$$

$$a_{11} = 0.5882, \qquad \kappa_1 = 0.5882, \qquad \kappa_2 = -0.32$$

$$h_2 = \frac{-10 \times 32}{90} \times 0.6 \times 1.5 = -3.2$$

$$h_1 a_{10} + h_2 a_{21} = 2.1 \times \left(\frac{-10 \times 32}{80}\right)$$

Therefore, $h_1 = -6.19$

$$h_0 + h_1 a_{11} + h_2 a_{22} = \frac{-10 \times 32}{90}$$

and $h_0 = -0.938$

d)

$$G(z) = \frac{10\,(1 + z + 0.5z^2)}{\left(1 - \frac{z}{0.4}\right)\left(1 + \frac{z}{0.8}\right)}$$
$$= -\,10 \times 0.4 \times 0.8 \frac{z^{-2} + z^{-1} + 0.5}{\left(1 - \frac{0.4}{z}\right)\left(1 + \frac{0.8}{z}\right)}$$

From section **a)**

$$a_{21} = 0.4, \qquad a_{22} = -0.32$$

$$a_{11} = 0.5882, \qquad \kappa_1 = 0.5882, \qquad \kappa_2 = -0.32$$

$$h_2 = 10 \times 0.4 \times 0.8 = -3.2$$

$$h_1 a_{10} + h_2 a_{21} = -10 \times 0.4.8$$

Therefore $h_1 = -1.92$

$$h_0 + h_1 a_{11} + h_2 a_{22} = -0.5 \times 10 \times 0.4 \times 0.8$$

and $h_0 = -1.495$

Problem 3.30

a)

$$A\left(e^{\mathrm{j}\,\omega}\right) = \sum_{k=0}^{M} a_k \exp\left(-\,\mathrm{j}\,\omega_k\right)$$

$$\hat{H}\left(e^{\mathrm{j}\,\omega}\right) = \sum_{i=0}^{N} \hat{h}(i) \exp\left(-\,\mathrm{j}\,\omega_i\right)$$

$$\hat{H}\left(e^{\mathrm{j}\,\omega_m}\right) A\left(e^{\mathrm{j}\,\omega_m}\right) = 1 \tag{1}$$

$$\hat{p}(\omega) = \left|\hat{H}\left(\omega\right)\right|^2$$

$$\hat{H}^*\left(e^{\mathrm{j}\,\omega}\right)\hat{H}\left(e^{\mathrm{j}\,\omega}\right)A\left(e^{\mathrm{j}\,\omega}\right)=\hat{H}^*\left(e^{\mathrm{j}\,\omega}\right)$$

$$\hat{p}(\omega_m)A\left(e^{\mathrm{j}\,\omega_m}\right)=\hat{H}^*\left(e^{\mathrm{j}\,\omega_m}\right)$$

$$\sum_{k=0}^{p}\hat{p}\left(e^{\mathrm{j}\,\omega_m}\right)a_k\exp\left(-\mathrm{j}\,\omega_m k\right)=\hat{H}^*\left(e^{\mathrm{j}\,\omega_m}\right)$$

$$\sum_{k=1}^{p}\sum_{m=1}^{N}\hat{p}\left(e^{\mathrm{j}\,\omega_m}\right)a_k\exp\left(-\mathrm{j}\,\omega_{mk}\right)\exp\left(-\mathrm{j}\,\omega_{mi}\right)=\sum_{m=1}^{N}\hat{H}^*\left(e^{\mathrm{j}\,\omega_m}\right)\exp\left(-\mathrm{j}\,\omega_{mi}\right)$$

From the definition

$$\hat{R}(i)=\frac{1}{N}\sum_{m=1}^{N}\hat{p}\left(\omega_m\right)\cos(\omega_{mi})$$

and averaging over $\omega_m\in\Omega$, we have

$$\begin{aligned}\sum_{k=0}^{p}a_k\hat{R}(i-k)=&\frac{1}{N}\sum_{m=1}^{N}\hat{H}\left(\omega_m\right)\exp\left(-\mathrm{j}\,\omega_{mi}\right)\\=&\hat{h}(-1)\text{ for all }i\\=&\frac{1}{N}\sum_{m=1}^{N}\left[\frac{\exp\left(\mathrm{j}\,\omega_{mi}\right)}{\sum_{k=0}^{M}a_k\exp\left(-\mathrm{j}\,\omega_{mk}\right)}\right]\end{aligned}$$

b)

$$D(w)=\sum_{k=0}^{M}d_k\cos(\omega_k)$$

$$\begin{aligned}D_{IS}=&\sum_{k=0}^{N-1}\left[\frac{|U_k|^2}{S_k(a)}-\ln\frac{|U_k|^2}{S_k(a)}-1\right]\\=&\sum_{k=0}^{N-1}\left[|U_k|^2\sum_{k=0}^{M}\cos\left(\omega_k\right)-\ln\left[|U_k|^2\sum_{k=0}^{M}d_k\cos\left(\omega_k\right)-1\right]\right]\end{aligned}$$

$$\frac{\partial D_{JS}}{\partial d_i} = 0$$

$$= |U_i|^2 \cos(\omega_i) - \frac{|U_i|^2 \cos(\omega_i)}{|U_i|^2 \sum_{k=0}^{M} d_k \cos(\omega_k)}$$

Therefore $|U_i|^2 \sum_{k=0}^{M} d_k \cos(\omega_k) = 1 \qquad$ for $j = 0, \ldots, N-1$

$$|U_i|^2 = \left| \hat{H}\left(e^{\mathrm{j}\,\omega_m}\right) \right|^2$$

$$p(w_m) A(e^{\mathrm{j}\,\omega_m}) = \hat{H}^*\left(e^{\mathrm{j}\,\omega_m}\right)$$

as in part **a)**, it can be shown:

$$\sum_{k=0}^{p} a_k R(i-k) = \hat{h}(-i)$$

where $R(i) = \frac{1}{N} \sum_{m=1}^{N} p(w_m) \cos(\omega_{mi})$ By comparing with part **a)** the results are:

$$\sum_{k=0}^{p} a_k \hat{R}(j-k) = \sum_{k=0}^{p} a_k \hat{R}(j-k) \qquad \text{for } 0 \le i \le p$$

Therefore $\hat{R}(i) = \hat{R}(i)$

Chapter 4

Problem 4.1

a)

For convergence of the steepest-descent algorithm:

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

where λ_{max} is the largest eigenvalue of the correlation matrix $\mathbf{R}$. We are given

$$\mathbf{R} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

The two eigenvalue of $\mathbf{R}$ are

$$\lambda_1 = 0.5$$

$$\lambda_2 = 1.5$$

Hence $\lambda_{\text{max}} = 1.5$. The step-size parameter μ must therefore satisfy the condition

$$0 < \mu < \frac{2}{1.5} = 1.334$$

We may thus choose $\mu = 1.0$

b)

From Equation (4.10) of the textbook:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\left[\mathbf{p} - \mathbf{R}\mathbf{w}(n)\right]$$

with $\mu = 1$ and

$$\mathbf{p} = \begin{bmatrix} 0.5 \\ 0.25 \end{bmatrix}$$

we therefore have

$$\begin{aligned} \mathbf{w}(n+1) =& \mathbf{w}(n) + \left\{ \begin{bmatrix} 0.5 \\ 0.25 \end{bmatrix} - \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix} \mathbf{w}(n) \right\} \\ =& \left\{ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix} \right\} \mathbf{w}(n) + \begin{bmatrix} 0.5 \\ 0.25 \end{bmatrix} \\ =& \begin{bmatrix} 0 & -0.5 \\ -0.5 & 0 \end{bmatrix} \mathbf{w}(n) + \begin{bmatrix} 0.5 \\ 0.25 \end{bmatrix} \end{aligned}$$

That is,

$$\begin{bmatrix} w_1(n+1) \\ w_2(n+1) \end{bmatrix} = \begin{bmatrix} 0 & -0.5 \\ -0.5 & 0 \end{bmatrix} \begin{bmatrix} w_1(n) \\ w_2(n) \end{bmatrix} + \begin{bmatrix} 0.5 \\ 0.25 \end{bmatrix}$$

Equivalently, we may write

$$w_1(n+1) = -0.5 w_2(n) + 0.5$$

$$w_2(n+1) = -0.5 w_1(n) + 0.25$$

c)

To investigate the effect of varying The step-size paramter μ on the trajectory, we find it convenient to work with $\nu(n)$ rather than with $w(n)$. The kth natural mode of the steepest descent algorithm is described by

$$\nu_k(n+1) = (1 - \mu\lambda_k)\, \nu_k(n), \qquad k = 1, 2$$

Specifically,

$$\nu_1(n+1) = (1 - 0.5\mu)\, \nu_1(n)$$

$$\nu_2(n+1) = (1 - 1.5\mu)\, \nu_2(n)$$

For the initial condition, we have

$$\begin{aligned} \mathbf{v}(0) =& \begin{bmatrix} \nu_1(0) \\ \nu_2(0) \end{bmatrix} \\ =& \mathbf{Q}^H \mathbf{w}_0 \end{aligned}$$

From the solution to Problem 2.2:

$$\mathbf{w}_0 = \begin{bmatrix} 0.5 \\ 0 \end{bmatrix}$$

$$\mathbf{Q} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$$

Hence,

$$\begin{aligned} \mathbf{v}_0 =& \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0 \end{bmatrix} \\ =& \frac{1}{\sqrt{2}} \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \end{aligned}$$

That is,

$$\nu_1(0) = \nu_2(0) = \frac{0.5}{\sqrt{2}}$$

For $n \geq 0$, we have

$$\nu_k(n) = (1 - \mu\lambda_k)^n \, \nu_k(0), \qquad k = 1, 2$$

Hence,

$$\nu_1(n) = (1 - 0.5\mu)^n \, \nu_1(0)$$

$$\nu_2(n) = (1 - 1.5\mu)^n \, \nu_2(0)$$

$$\boxed{\mu = 1}$$

$$\nu_1(n) = (0.5)^n \, \nu_1(0)$$

$$\nu_2(n) = (-0.5)^n \, \nu_2(0)$$

Solution- This represents an oscillatory trajectory.

$$\boxed{\mu = 0.1}$$

$$\nu_1(n) = (0.59)^n \, \nu_1(0)$$

$$\nu_2(n) = (0.85)^n \, \nu_2(0)$$

Solution- This represents a damped trajectory.

The transition from a damped to an oscillatory trajectory occurs at

$$\mu = \frac{1}{1.5} = 0.667$$

Specifically, for $0 < \mu < 0.667$, the trajectory is damped. On the other hand, for $0.667 < \mu < 1.334$ the trajectory is oscillatory.

Problem 4.2

We are given
$J(w) = J_{\text{min}} + r(0)(w - w_0)^2$

a)
The correlation matrix $\mathbf{R} = r(0)$,

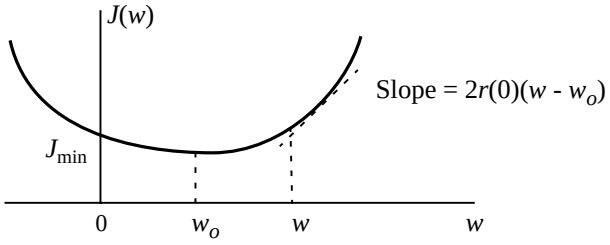
Hence, $\lambda_{\text{max}} = r(0)$

Correspondingly, $\mu_{\text{max}} = \dfrac{2}{\lambda_{\text{max}}} = \dfrac{2}{r(0)}$

b)
The time constant of the filter is

$$\tau_1 \approx \frac{1}{\mu\lambda_1} = \frac{1}{\mu r(0)}$$

c)



$$\frac{\partial J}{\partial w} = 2r(0)\,(w - w_0)$$

Problem 4.3

a)
There is a single mode with eigenvalue $\lambda = r(0)$, and $q_1 = 1$, Hence,

$$J(n) = j_{\text{min}} + \lambda_1 \left|\nu_1(n)\right|^2$$

where

$$\nu_1(n) = q_1\,(w_0 - w(n)) = (w_0 - w(n))$$

b)

$$\frac{\partial J}{\partial \lambda_1} = |\nu_1(n)|^2 = (w_0 - w(n))^2$$

Problem 4.4

The estimation error $e(n)$ equals

$$e(n) = d(n) - \mathbf{w}^H(n)\mathbf{u}(n)$$

where $d(n)$ is the desired response, $\mathbf{w}(n)$ is the tap-weight vector, and $\mathbf{u}(n)$ is the tap-input vector. Hence, the gradient of the instantaneous squared error equals

$$\begin{aligned}\hat{\Delta}J(n) =& \frac{\partial}{\partial \mathbf{w}}\left[|e(n)|^2\right] = \frac{\partial}{\partial \mathbf{w}}\left[e(n)e^*(n)\right] \\ =& e^*(n)\frac{\partial e(n)}{\partial \mathbf{w}} + e(n)\frac{\partial e^*(n)}{\partial \mathbf{w}} \\ =& -2e^*(n)\mathbf{u}(n) \\ =& 2\mathbf{u}(n)d^*(n) + 2\mathbf{u}(n)\mathbf{u}^H(n)\mathbf{w}(n)\end{aligned}$$

Problem 4.5

Consider the approximation to the inverse of the correlation matrix:

$$\mathbf{R}^{-1}(n+1) \approx \mu \sum_{k=0}^{n} (\mathbf{I} - \mu\mathbf{R})^k$$

where μ is a positive constant bounded in value as

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

where λ_{max} is the largest eigenvalue of $\mathbf{R}$. Note that according to this approximation, we have $\mathbf{R}^{-1}(1) = \mu\mathbf{I}$. Correspondingly, we may approximate the optimum Wiener solution

as follows:

$$\begin{aligned}\mathbf{w}(n+1) =& \mathbf{R}^{-1}(n+1)\mathbf{p} \\ \approx& \mu \sum_{k=0}^{n} (\mathbf{I} - \mu\mathbf{R})^k \mathbf{p} \\ =& \mu\mathbf{p} + \mu \sum_{k=1}^{n} (\mathbf{I} - \mu\mathbf{R})^k \mathbf{p}, \qquad \mathbf{I} = \text{ identity matrix}\end{aligned}$$

In the second term, put $k = i + 1$ or $i = k - 1$

$$\begin{aligned}\mathbf{w}(n+1) =& \mu\mathbf{p} + \mu (\mathbf{I} - \mu\mathbf{R}) \sum_{k=1}^{n} (\mathbf{I} - \mu\mathbf{R})^k \mathbf{p} \\ =& \mu\mathbf{p} + (\mathbf{I} - \mu\mathbf{R}) \mathbf{w}(n) \qquad (1)\end{aligned}$$

where, in the second line, we have used the fact that

$$\sum_{k=1}^{n} (\mathbf{I} - \mu\mathbf{R})^k \mathbf{p} = \mathbf{w}(n)$$

Hence, rearranging Equation (1)

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu [\mathbf{p} - \mathbf{R}\mathbf{w}(n)]$$

which is the standard formula for the steepest descent algorithm.

Problem 4.6

$$J(\mathbf{w}(n+1)) = J(\mathbf{w}(n)) - \frac{1}{2}\mu \|\mathbf{g}(n)\|^2$$

For stability of the steepest-descent algorithm, we require

$$J(\mathbf{w}(n+1)) < J(\mathbf{w}(n))$$

To satisfy this requirement, the step-size parameter μ should be positive, since

$$\|\mathbf{g}(n)\|^2 > 0$$

Hence, the steepest-descent algorithm becomes unstable when the step-size parameter is negative.

Problem 4.7

Let

$$\begin{aligned}\delta\mathbf{w}(n+1) = &\ \mathbf{w}(n+1) - \mathbf{w}(n) \\ = &\ \mu\left[\mathbf{p} - \mathbf{R}\mathbf{w}(n)\right]\end{aligned}$$

By definition,

$$\mathbf{p} = \mathbb{E}\left[\mathbf{u}(n)d^*(n)\right]$$

$$\mathbf{R} = \mathbb{E}\left[\mathbf{u}(n)\mathbf{u}^H(n)\right]$$

Hence,

$$\begin{aligned}\delta\mathbf{w}(n+1) = &\ \left\{\mu\mathbb{E}\left[\mathbf{u}(n)\mathbf{u}^H(n)\right] - \mathbb{E}\left[\mathbf{u}(n)d^*(n)\right]\mathbf{w}(n)\right\} \\ = &\ \mu\mathbb{E}\left[\mathbf{u}(n)\left\{d^*(n) - \mathbf{u}^H(n)\mathbf{w}(n)\right\}\right]\end{aligned}$$

But the estimation error is

$$e(n) = d(n) - \mathbf{w}^H(n)\mathbf{u}(n)$$

Therefore,

$$\delta\mathbf{w}(n+1) = \mu\mathbb{E}\left[\mathbf{u}(n)e^*(n)\right]$$

At the minimum point of the error-performance surface, the expression $\delta\mathbf{w}(n+1)$ is zero. Hence, at this point, we have

$$\mathbb{E}\left[\mathbf{u}(n)e^*(n)\right] = 0$$

which is a restatement of the principle of orthogonality.

Problem 4.8

We are given

$$J_{\text{approx}}(n) = \left[J(0) - J(\infty)\right]\exp\left(-n/\tau\right) + J(\infty)$$

For $n = 1$, we have

$$\begin{aligned}J_{\text{approx}}(1) = &J(1) \\ &= \left[J(0) - J(\infty)\right]\exp\left(-1/\tau\right) + J(\infty)\end{aligned}$$

Solving for $\exp(-1/\tau)$, we get

$$\exp(-1/\tau) = \frac{J(0) - J(\infty)}{J(1) - J(\infty)}$$

Taking natural logarithms:

$$\frac{1}{\tau} = \ln\left[\frac{J(0) - J(\infty)}{J(1) - J(\infty)}\right]$$

or

$$\tau = \ln\left[\frac{J(1) - J(\infty)}{J(0) - J(\infty)}\right]$$

Problem 4.9

a)

The cross-correlation between the "desired response" $u(n)$ and the tap input $u(n-1)$ is

$$p = \mathbb{E}[u(n)u(n-1)] = r(1)$$

The mean-square value of the tap input $u(n-1)$ is

$$\mathbb{E}\left[u^2(n-1)\right] = r(0)$$

Hence,

$$w(n+1) = w(n) + \mu\left(r\left(1\right) - r\left(0\right)w(n)\right)$$

For recursive computation of the parameter a, we note that $a(n) = -w(n)$; hence

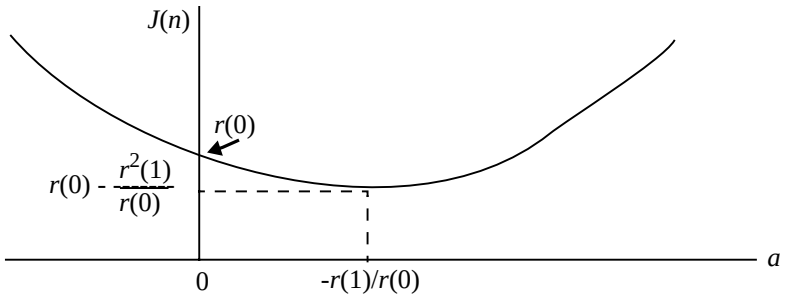
$$a(n+1) = a(n) - \mu\left(r\left(1\right) + r\left(0\right)a(n)\right)$$

b)

The error-performance surface is defined by

$$\begin{aligned} J(n) =& r(0) - 2r(1)w(n) + r(0)w^2(n) \\ =& r(0) + 2r(1)a(n) + r(0)a^2(n) \end{aligned}$$

The corresponding plot of the error performance surface is therefore



c)
The condition on the step-size parameter is

$$0 < \mu < \frac{2}{r(0)}$$

Problem 4.10

The second-order AR process $u(n)$ is described by the difference equation

$$u(n) = -0.5u(n-1) + u(n-2) + \nu(n) \tag{1}$$

Hence,

$$w_1 = -0.5 \qquad w_2 = 1$$

and the AR parameters equal

$$a_0 = 1, \qquad a_1 = 0.5, \qquad a_2 = -1$$

Accordingly, we write the Yule-Walker equations as

$$\begin{bmatrix} r(0) & r(1) \\ r(1) & r(0) \end{bmatrix} \begin{bmatrix} -0.5 \\ 1 \end{bmatrix} = \begin{bmatrix} r(1) \\ r(2) \end{bmatrix} \tag{2}$$

$$\sigma_\nu^2 = \sum_{k=0}^{2} a_k r(k)$$

$$\begin{aligned} 1 =& a_0 r(0) + a_1 r(1) + a_2 r(2) \\ =& r(0) + 0.5r(1) - r(2) \end{aligned} \tag{3}$$

Equations (1), (2),and (3) yield

$$r(0) = 0$$

$$r(1) = 1$$

$$r(2) = -1/2$$

Hence,

$$\mathbf{R} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

The eigenvalues of $\mathbf{R}$ are -1, +1. For convergence of the steepest descent algorithm:

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

where λ_{max} is the largest eigenvalue of the correlation matrix $\mathbf{R}$. Hence, with $\lambda_{\text{max}} = 1$.

$$0 < \mu < 2$$

Problem 4.11

$$u(n-2) = 0.5u(n-1) + u(n) - \nu(n) \tag{1}$$

Hence,

$$w_1 = 1 \qquad w_2 = 0.5$$

Accordingly, we may write $\mathbf{R}\mathbf{w}_b = \mathbf{r}^{B*}$ as

$$\begin{bmatrix} r(0) & r(1) \\ r(1) & r(0) \end{bmatrix} \begin{bmatrix} 1 \\ 0.5 \end{bmatrix} = \begin{bmatrix} r(2) \\ r(1) \end{bmatrix}$$

$$\sigma_\nu^2 = \sum_{k=0}^{2} a_k r(k)$$

$$1 = r(0) - r(1) - 0.5r(2) \tag{2}$$

$$r(0) = 0$$

$$r(1) = -2/3$$

Therefore,

$$\mathbf{R} = \begin{bmatrix} 0 & -2/3 \\ -2/3 & 0 \end{bmatrix}$$

The eigenvalues of $\mathbf{R}$ are $\lambda = -2/3, 2/3$

For convergence of the steepest descent algorithm:

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

$$0 < \mu < \frac{2}{2/3}$$

$$0 < \mu < 3$$

Problem 4.12

The third-order AR process $u(n)$ is described by the difference equation

$$u(n) = -0.5u(n-1) - 0.5u(n-2) + 0.5u(n-3) + \nu(n)$$

Hence,

$$w_1 = -0.5, \qquad w_2 = -0.5, \qquad w_3 = 0.5$$

and the AR parameters are as follows:

$$a_1 = 0.5, \qquad a_2 = 0.5, \qquad a_3 = -0.5$$

Accordingly, we write the Yule-Walker equations as

$$\begin{bmatrix} r(0) & r(1) & r(2) \\ r(1) & r(0) & r(1) \\ r(2) & r(1) & r(0) \end{bmatrix} \begin{bmatrix} -0.5 \\ -0.5 \\ 0.5 \end{bmatrix} = \begin{bmatrix} r(1) \\ r(2) \\ r(3) \end{bmatrix}$$

$$\sigma_\nu^2 = \sum_{k=0}^{3} a_k r(k)$$

$$1 = r(0) + a_1 r(1) + a_2 r(1) + a_3$$

We may therefore determine:

$$r(0) = 1$$

$$r(1) = \frac{-1}{2}$$

$$r(2) = \frac{-1}{2}$$

$$r(3) = 1$$

a)
The correlation matrix is

$$\mathbf{R} = \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{2} & 1 \end{bmatrix}$$

b)
The eigenvalues of $\mathbf{R}$ are 0, 1.5, 1.5

c)
For convergence of the steepest descent algorithm, we require

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

Hence, with $\lambda_{\text{max}} = 1.5$, we have

$$0 < \mu < \frac{2}{1.5}$$

Problem 4.13

a)
The first-order MA process is described by the difference equation

$$u(n) = \nu(n) - 0.2\nu(n-1)$$

Taking the z-transform of both sides:

$$U(z) = \left(1 - 0.2z^{-1}\right) V(z)$$

$$\frac{U(z)}{V(z)} = \frac{1}{(1-0.2z^{-1})^{-1}}$$
$$\approx \frac{1}{1-0.2z^{-1}+0.04z^{-2}}$$

Accordingly,

$$u(n) + 0.2u(n-1) + 0.04u(n-2) = \nu(n)$$
$$u(n) = -0.2u(n-1) - 0.04u(n-2) + \nu(n)$$

for which we infer the following values

$$w_1 = -0.2, \qquad w_2 = -0.04$$

Correspondingly, the AR parameters equal

$$a_1 = 0.2, \qquad a_2 = 0.04$$

Yule-Walker equations are

$$\begin{bmatrix} r(0) & r(1) \\ r(1) & r(0) \end{bmatrix} \begin{bmatrix} -0.2 \\ -0.04 \end{bmatrix} = \begin{bmatrix} r(1) \\ r(2) \end{bmatrix} \tag{1}$$

$$\sigma_\nu^2 = \sum_{k=0}^{2} a_k r(k) \tag{2}$$

$$1 = r(0) + a_1 r(1) + a_2 r(2) \tag{3}$$

Equations (1), (2) and (3) yield

$$r(0) = 1.0401$$
$$r(1) = -0.2000$$
$$r(2) = -0.0016$$

The correlation matrix is

$$\mathbf{R} = \begin{bmatrix} r(0) & r(1) \\ r(1) & r(0) \end{bmatrix} = \begin{bmatrix} 1.0401 & -0.2000 \\ -0.2000 & 1.0401 \end{bmatrix}$$

The eigenvalues of $\mathbf{R}$ are $1.2401, 0.8401$, and therefore $\lambda_{\text{max}} = 1.2401$

For convergence of the steepest descent algorithm:

$$0 < \mu < \frac{2}{1.2401}$$

That is

$$0 < \mu < 1.6127$$

b)

$$\begin{aligned}\frac{U(z)}{V(z)} =& \frac{1}{(1-0.2z^{-1})^{-1}} \\ \approx& \frac{1}{1+0.2z^{-1}+0.04z^{-2}+0.008z^{-3}}\end{aligned}$$

Accordingly,

$$u(n) + 0.2u(n-1) + 0.04u(n-2) + 0.008u(n-3) = \nu(n)$$

$$w_1 = -0.2, \qquad w_2 = -0.04, \qquad w_3 = -0.008$$

The AR parameters are

$$a_1 = 0.2, \qquad a_2 = 0.04, \qquad a_3 = 0.008$$

The Yule-Walker equations are

$$\begin{bmatrix} r(0) & r(1) & r(2) \\ r(1) & r(0) & r(1) \\ r(2) & r(1) & r(0) \end{bmatrix} \begin{bmatrix} -0.2 \\ -0.04 \\ 0.008 \end{bmatrix} = \begin{bmatrix} r(1) \\ r(2) \\ r(3) \end{bmatrix} \tag{4}$$

$$\sigma_\nu^2 = \sum_{k=0}^{3} a_k r(k) \tag{5}$$

$$1 = r(0) + a_1 r(1) + a_2 r(2) + a_3 r(3) \tag{6}$$

Equations (4), (5) and (6) yield

$$r(0) = 1.04$$

$$r(1) = -0.2$$

$$r(2) = 0.0$$

$$r(3) = -0.003$$

The correlation matrix is

$$\mathbf{R} = \begin{bmatrix} 1.04 & -0.2 & 0 \\ -0.2 & 1.04 & -0.2 \\ 0 & -0.2 & 1.04 \end{bmatrix}$$

and its eigenvalues are

$$\lambda = 0.7572, 1.04, 1.3228, \qquad \lambda_{\text{max}} = 1.3228$$

For convergence of the steepest descent algorithm:

$$0 < \mu < \frac{2}{1.3228}$$
$$0 < \mu < 1.5119$$

c)

When the system is approximated by a second-order AR model,

$$0 < \mu < 1.6127$$

When it is approximated by a third-order AR model,

$$0 < \mu < 1.5119$$

These results show that the upper bound on the step-size parameters becomes tighter as the approximating model order is increased.

Problem 4.14

a)

$$u(n) = -0.5u(n-1) + \nu(n) - 0.2\nu(n-1)$$

Take the z-transform of both sides:

$$U(z) = -0.5z^{-1}U(z) + V(z) - 0.2z^{-1}V(z)$$

$$\begin{aligned} \frac{U(z)}{V(z)} &= \frac{1 - 0.2z^{-1}}{1 + 0.5z^{-1}} \\ &= \frac{1}{(1 + 0.5z^{-1})\,(1 - 0.2z^{-1})^{-1}} \end{aligned}$$

Approximation to third-order yields

$$\frac{U(z)}{V(z)} \approx \frac{1}{(1+0.5z^{-1})\,(1+0.2z^{-1}+0.04z^{-2}+0.08z^{-3})}$$
$$\approx \frac{1}{1+0.7z^{-1}+0.14z^{-2}+0.028z^{-3}}$$

Accordingly,

$$u(n)+0.7u(n-1)+0.14u(n-2)+0.028u(n-3)=\nu(n)$$

$$u(n)=-0.7u(n-1)-0.14u(n-2)-0.028u(n-3)+\nu(n)$$

$$w_1=-0.7, \qquad w_2=-0.14, \qquad w_3=-0.028$$

The AR parameters are

$$a_1=0.7, \qquad a_2=0.14, \qquad a_3=0.028$$

The Yule-Walker equations are

$$\begin{bmatrix} r(0) & r(1) & r(2) \\ r(1) & r(0) & r(1) \\ r(2) & r(1) & r(0) \end{bmatrix} \begin{bmatrix} -0.7 \\ -0.14 \\ 0.028 \end{bmatrix} = \begin{bmatrix} r(1) \\ r(2) \\ r(3) \end{bmatrix} \tag{1}$$

$$\sigma_\nu^2 = \sum_{k=0}^{3} a_k r(k) \tag{2}$$

$$r(0)+a_1 r(1)+a_2 r(2)+a_3 r(3)=1 \tag{3}$$

Equations (4), (5) and (6) yield

$$r(0)=1.6554$$

$$r(1)=-1.0292$$

$$r(2)=0.5175$$

$$r(3)=-0.2646$$

The correlation matrix is

$$\mathbf{R}=\begin{bmatrix} r(0) & r(1) & r(2) \\ r(1) & r(0) & r(1) \\ r(2) & r(1) & r(0) \end{bmatrix} = \begin{bmatrix} 1.6554) & -1.0292 & 0.5175 \\ -1.0292 & 1.6554 & -1.0292 \\ 0.5175 & -1.0292 & 1.6554 \end{bmatrix}$$

b)
The eigenvalues of $\mathbf{R}$ are 0.4358, 1.1379, 3.3925, $\lambda_{\max} = 3.3925$

c)
For convergence of the steepest descent algorithm we require

$$0 < \mu < \frac{2}{\lambda_{\max}}, \qquad \lambda_{\max} = 3.3925$$

$$0 < \mu < \frac{2}{3.3925}$$

$$0 < \mu < 0.589$$

Problem 4.15

a)

$$\begin{aligned}\mathbf{w}(n+1) &= (1-\mu)\,\mathbf{w}(n) + \mu\mathbf{R}^{-1}\mathbf{p} \\ &= (1-\mu)\,\mathbf{w}(n) + \mu\mathbf{w}_0, \qquad \left(\mathbf{R}^{-1}\mathbf{p} = \mathbf{w}_0\right)\end{aligned}$$

Subtract $\mathbf{w}_0$ from both sides

$$\begin{aligned}\mathbf{w}(n+1) - \mathbf{w}_0 &= (1-\mu)\,\mathbf{w}(n) + \mu\mathbf{w}_0 - \mathbf{w}_0 \\ &= (1-\mu)\,(\mathbf{w}(n) - \mathbf{w}_0)\end{aligned}$$

Starting with the initial value $\mathbf{w}(0)$ and iterating:

$$\mathbf{w}(n) - \mathbf{w}_0 = (1-\mu)^n\,(\mathbf{w}(0) - \mathbf{w}_0)$$

From Equation (2.52) of the textbook:

$$\begin{aligned}J(n) =& J_{\min} + (\mathbf{w} - \mathbf{w}_0)^H\,\mathbf{R}\,(\mathbf{w} - \mathbf{w}_0) \qquad (1)\\ =& J_{\min} + (1-\mu)^n\,(\mathbf{w}(0) - \mathbf{w}_0)^H\,\mathbf{R}\,(1-\mu)^n\,(\mathbf{w}(0) - \mathbf{w}_0) \\ =& J_{\min} + (1-\mu)^{2n}\,(\mathbf{w}(0) - \mathbf{w}_0)^H\,\mathbf{R}\,(\mathbf{w}(0) - \mathbf{w}_0) \\ =& J_{\min} + (1-\mu)^{2n}\,(J(0) - J_{\min})\end{aligned}$$

The transient behavior of Newton's algorithm is thus characterized by a single exponential whose corresponding time constant is

$$(1-\mu)^{2k} = \exp(-k/\tau) \qquad (2)$$

where $\mu << 1$.

b)

Taking the logarithm of both sides of Equation (2):

$$2k \ln (1 - \mu) = -k/\tau$$

When $\mu << 1$, $\ln (1 - \mu) \approx -\mu$. Hence

$$-2k\mu \approx -k/\tau$$

from which we readily deduce the time constant

$$\tau = \frac{1}{2\mu}, \qquad \mu << 1$$

Chapter 5

Problem 5.1

For the sake of convenience, assume that the optimal solution of the graphs plotted in Figure P 5.1 is positioned along the horizontal (i.e., x) axis to be at $x = 0$. We may then make the following two observations:

a) In part (a) of Fig. P 5.1, we may describe the gradient as follows: the gradient has a positive slope, and it is located on the positive side of the graph.

b) In direct contrast, the gradient in part (b) of Fig. P 5.1 has a negative slope, and it is located on the negative side of the graph.

The two negatives in part (b) make up in a positive result, thereby confirming that the gradient has a positive value, aimed at the same minimal point in the graph

Problem 5.2

From Figure 6.1 (of the textbook) we see that the LMS algorithm requires $2M+1$ complex multiplications and $2M$ complex additions per iterations, where M is the number of tap weights used in the adaptive finite impulse filter. Therefore, the computational complexity of the LMS algorithm is $O(M)$

Problem 5.3

a)

We are given the real-valued update formula:

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu \mathbf{u}(n-i)\,\mathrm{sgn}\left[e(n)\right], \qquad i = 0, 1, \ldots, M-1$$

The computational complexity of this algorithm is $2M$, compared with $2M + 1$ of the computational complexity of the conventional LMS algorithm. It follows therefore that the rational behind the sign-error LMS algorithm is its computational simplicity.

b)
Following the conventional complex-valued LMS algorithm, the complex-valued version of the sign-error LMS algorithm is as follows:

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu \mathbf{u}(n-i)\,\mathrm{sgn}\,[e^*(i)]$$

According to the linearity property of the signum function, we may express $\mathrm{sgn}\,[e^*(n)]$ as follows:

$$\begin{aligned}\mathrm{sgn}\,[e^*(n)] &= \mathrm{sgn}\,[e_R(n) + \mathrm{j}\, e_I(n)]^*\,, \\ &= \mathrm{sgn}\,[e_R(n) - \mathrm{j}\, e_I(n)] \\ &= \mathrm{sgn}\,[e_R(n)] - \mathrm{j}\,\mathrm{sgn}\,[e_I(n)]\end{aligned} \qquad \begin{cases} e_R(n) = \text{real part of } e(n) \\ e_I(n) = \text{imaginary part of } e(n) \end{cases}$$

Problem 5.4

Referring back to Chapter 4 on the method of Steepest Descent, Equation (4.10) is reproduced here as follows:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\,[\mathbf{p} - \mathbf{R}\mathbf{w}(n)]\,, \qquad n = 0, 1, 2, \ldots \tag{1}$$

where the cross-correlation vector,

$$\mathbf{p} = \mathbb{E}\,[\mathbf{u}(n) d^*(n)] \tag{2}$$

and $\mathbf{R}$ is the correlation matrix

$$\mathbf{R} = \mathbb{E}\left[\mathbf{u}(n)\mathbf{u}^H(n)\right] \tag{3}$$

Substituting Equations (2) and (3) into (1), we write

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\,[\mathbb{E}\,[\mathbf{u}(n) d^*(n)] - \mathbb{E}\,[\mathbf{u}(n)\mathbf{u}^*(n)]\,\mathbf{w}(n)] \tag{4}$$

Let $\hat{\mathbf{w}}$ denote the modified value of $\mathbf{w}(n)$ that results from eliminating the expectation operator so as to simplify the algorithm. We may then, correspondingly, modify (4) into

the simplified form:

$$\begin{aligned}\hat{\mathbf{w}}(n+1) =& \hat{\mathbf{w}}(n) + \mu\left[\mathbf{u}(n)d^*(n) - \mathbf{u}(n)\mathbf{u}^H(n)\mathbf{w}(n)\right] \\ =& \hat{\mathbf{w}}(n) + \mu\mathbf{u}(n)\left[d^*(n) - \mathbf{u}^H(n)\mathbf{w}(n)\right] \\ =& \hat{\mathbf{w}}(n) + \mu\mathbf{u}(n)\left[d(n) - \mathbf{u}^T(n)\mathbf{w}^*(n)\right]^* \\ =& \hat{\mathbf{w}}(n) + \mu\mathbf{u}(n)\left[d(n) - \mathbf{w}^H(n)\mathbf{u}(n)\right]^* \\ =& \hat{\mathbf{w}}(n) + \mu\mathbf{u}(n)e^*(n) \end{aligned} \tag{5}$$

where $e(n)$ is the error signal defined by

$$e(n) = d(n) - \mathbf{w}^H(n)\mathbf{u}(n) \tag{6}$$

Equation (5) defines the same update formula for the LMS algorithm as that defined in Equation (5.6) from the textbook.

Problem 5.5

a)

We are given

$$\begin{aligned} J(n) =& |e(n)|^2 + \alpha\|\mathbf{w}(n)\|^2 \\ =& e(n)e^*(n) + \alpha\mathbf{w}^H(n)\mathbf{w}(n) \end{aligned}$$

Hence, using the material described in appendix B:

$$\begin{aligned}\nabla J(n) =& \frac{\partial J(n)}{\partial \mathbf{w}^*(n)} \\ =& \frac{\partial e(n)}{\partial \mathbf{w}^*(n)}e^*(n) + \frac{\partial e^*(n)}{\partial \mathbf{w}^*(n)} + 2\alpha\mathbf{w}(n) \end{aligned} \tag{1}$$

By definition,

$$e(n) = d(n) - \mathbf{w}^H(n)\mathbf{u}(n)$$

$$e^*(n) = d^*(n) - \mathbf{u}^H(n)\mathbf{w}(n)$$

Therefore,

$$\frac{\partial e(n)}{\partial \mathbf{w}^*(n)} = -\mathbf{u}(n) \text{ and } \frac{\partial e^*(n)}{\partial \mathbf{w}^*(n)} = \mathbf{0}$$

Thus, Equation (1) reduces to

$$\nabla J(n) = -2\mathbf{u}(n)e^*(n) + 2\alpha\mathbf{w}$$

The update for the tap-weight vector in the leaky LMS algorithm is therefore as follows:

$$\begin{aligned}\hat{\mathbf{w}}(n+1) =& \hat{\mathbf{w}}(n) - \frac{1}{2}\mu\nabla J(n)\\ =& \hat{\mathbf{w}}(n) - \frac{1}{2}\mu\left(\mathbf{u}(n)e^*(n) - \alpha\mathbf{w}(n)\right)\\ =& (1-\mu\alpha)\hat{\mathbf{w}}(n) - \mu\mathbf{u}(n)e^*(n)\end{aligned} \tag{2}$$

b)

Let the input vector applied to the standard LMS algorithm be written as

$$\mathbf{u}_{\text{eq}}(n) = \mathbf{u}(n) + \mathbf{v}(n) \tag{3}$$

where each element of $\mathbf{v}(n)$ is a white noise process of zero mean and variance α. The correlation matrix of $\mathbf{u}_{\text{eq}}(n)$ is therefore

$$\mathbf{R}_{\text{eq}} = \mathbf{R} + \alpha\mathbf{I}$$

Correspondingly, we may write

$$\begin{aligned}\lim_{n\to\infty}\hat{\mathbf{w}}(n) =& \mathbf{R}_{\text{eq}}^{-1}\mathbf{p}\\ =& (\mathbf{R}+\alpha\mathbf{I})^{-1}\mathbf{p}\end{aligned}$$

Thus, the standard LMS algorithm is equivalent to a leaky LMS algorithm with its input vector $\mathbf{u}_{\text{eq}}(n)$ related to the input vector $\mathbf{u}(n)$ of the leaky LMS algorithm in the manner described in equation (3)

Problem 5.6

*

*Correction: The instantaneous cost function of the fourth-least-mean (FLM) algorithm is meant to have the weights term bolded indicating a vector

a)[†]

The instantaneous cost function is defined by

$$J_s(\mathbf{w}) = |e(n)|^4 \tag{1}$$

Following (5.3) and (5.4) in the textbook, we have

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}_k(n) - \frac{1}{2}\mu \nabla J_{s,k}(n) \tag{2}$$

where

$$\begin{aligned}\nabla J_{s,k}(n) =& \frac{\partial J_s(\mathbf{w})}{\partial w_k^H} \\ =& \frac{\partial J_{s,k}(\mathbf{w})}{\partial e(n)} \cdot \frac{\partial e(n)}{\partial \mathbf{w}^H}\end{aligned} \tag{3}$$

For the first partial derivative in (3), we use (1) to write

$$\begin{aligned}\frac{\partial J_s(\mathbf{w})}{\partial e(n)} =& \frac{\partial}{\partial e(n)} |e_n|^4 \\ =& 2\left(e(n)e^*(n)\right) \frac{\partial}{\partial e(n)} \left(e(n)e^*(n)\right)\end{aligned} \tag{4}$$

Involving the Wirtinger calculus in Appendix B, we differentiate with respect to $e(n)$ and treat $e^*(n)$ as a constant. Hence,

$$\frac{\partial}{\partial e(n)} \left(e(n)e^*(n)\right) = e^*(n) \tag{5}$$

Accordingly,

$$\begin{aligned}\frac{\partial J_s(\mathbf{w})}{\partial e(n)} =& 2\left(e(n)e^*(n)\right) e^*(n) \\ =& 2\left|e(n)\right|^2 e^*(n)\end{aligned} \tag{6}$$

For the second partial derivative in (3), we want

$$\begin{aligned}\frac{\partial e(n)}{\partial \mathbf{w}^H(n)} =& \frac{\partial}{\partial \mathbf{w}^H(n)} \left(d(n) - \mathbf{w}^H(n)\mathbf{u}(n)\right) \\ =& \frac{\partial}{\partial \mathbf{w}^H(n)} \left(d(n) - \mathbf{u}^T(n)\mathbf{w}^*(n)\right) \\ =& -\mathbf{u}(n)\end{aligned} \tag{7}$$

[†]Correction: The update formula should be given as: $\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + 2\mu\mathbf{u}(n)e^*(n)\left|e(n)\right|^2$ not $\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu\mathbf{u}(n-i)e^*(n)\left|e(n)\right|^2 \quad i = 0, 1, \ldots, M-1$

Thus, using (6) and (7) in (3), we get

$$\frac{\partial J_s(\mathbf{w})}{\partial \mathbf{w}^H} = -2\,|e(n)|^2\, e^*(n) \cdot \mathbf{u}(n) \tag{8}$$

Finally, substituting (8) into (2), we get the desired FLM algorithm expressed as follows:

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + 2\mu\,|e(n)|^2\, e^*(n)\mathbf{u}(n) \tag{9}$$

b)
The corresponding LMS algorithm is defined as follows

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu\,|e(n)|^2\, e^*(n)\mathbf{u}(n) \tag{10}$$

Putting aside the multiplying factor 2 in (4), the major difference between the FLM and LMS algorithm is the presence of the squared absolute value of $e(n)$ in the adjustment term of the FLM algorithm. Naturally, the presence of the additional term makes the FLM algorithm computationally more demanding than the LMS algorithm.

c)
The practical advantage of the FLM algorithm compared to the LMS algorithm is a faster rate of convergence, attributed to the additional term $|e(n)|^2$.

Problem 5.7

The normalized LMS filter (which is to be discussed in detail in chapter 7) is a manifestation of the *principle of minimal disturbance*, which may be stated as follows:

> From one iteration to the next, the weight vector of an adaptive filter should be changed in a minimal manner, subject to a constraint imposed on the updated filter's output.

To cast the principle in mathematical terms, let $\hat{\mathbf{w}}(n)$ denote the old weight vector of the filter at iteration n and $\hat{\mathbf{w}}(n+1)$ denote its updated weight vector at iteration $n+1$. We may then formulate the criterion for designing the normalized LMS filter as that of constrained optimization: Given the tap-input vector $\mathbf{u}(n)$ and the desired response $d(n)$, determine the updated tap-weight vector $\hat{\mathbf{w}}(n+1)$ so as to minimize the squared Euclidean norm of the change,

$$\delta\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n+1) - \hat{\mathbf{w}}(n)$$

subject to the constraint

$$\hat{\mathbf{w}}^H(n+1)\mathbf{u}(n) = d(n)$$

The update solution that satisfies this optimization problem comes up with the following statement:

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{\tilde{\mu}}{\|\mathbf{u}(n)\|^2}\mathbf{u}(n)e^*(n) \tag{1}$$

Consider now the following equivalence between paramters in the generalized LMS algorithm and the corresponding ones in Figure 5.4;

generalized LMS	Backward prediction
$\hat{\mathbf{w}}_m(n)$	$\hat{h}_m(n)$
$\mathbf{u}_m(n)$	$b_m(n)$
$\|\mathbf{u}_m(n)\|^2$	$\|b_m(n)\|^2$
$e_m(n)$	$e_m(n)$

Accordingly, we may write

$$\hat{h}_m(n+1) = \hat{h}_m(n) + \frac{\tilde{\mu}}{\|b_m(n)\|^2}b_m(n)e^*(n)$$

which is a restatement of Equation (5.27) A similar restatement can be made for Equation (5.28).

Problem 5.8

a)

Suppose we write

$$\begin{aligned} f_m(i) =& f_{m-1}(i) + \hat{\kappa}_m(n)b_{m-1}(i-1) \\ b_m(i) =& b_{m-1}(i) + \hat{\kappa}_m(n)f_{m-1}(i) \end{aligned}$$

Then, substituting these relations in Burg's formula:

$$\hat{\kappa} = -\frac{2\sum_{i=1}^{n} b_{m-1}(i-1) f_{m-1}^{*}(i)}{\sum_{i=1}^{n} |f_{m-1}(i)|^2 + |b_{m-1}(i-1)|^2}$$

$$-\frac{2\sum_{i=1}^{n} b_{m-1}(i-1) f_{m-1}^{*}(i) + 2b_{m-1}(n-1) f_{m-1}^{*}(n)}{E_{m-1}(n-1) + |f_{m-1}(i)|^2 + |b_{m-1}(i-1)|^2}$$

Cross-multiplying and proceeding in a manner similar to that described after Equations (5.20) and (5.21) in the text, we finally get

$$\hat{\kappa}_m(n) = \hat{\kappa}_m(n-1) - \frac{f_{m-1}^{*}(n) b_m(n) + b_{m-1}(n-1) f_m^{*}(n)}{E_{m-1}(n-1)}$$

b)

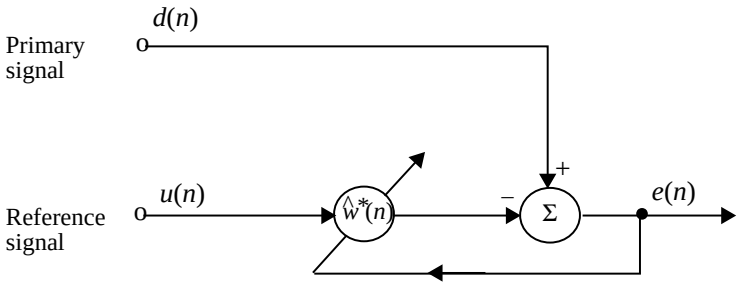
The algorithm so formulated is impractical because to compute the updated forward prediction error $f_m(n)$ and backward prediction error $b_m(n)$, we need to know the updated $\hat{\kappa}_m(n)$. This is not possible to do because the correction term for $\hat{\kappa}_m(n)$ requires knowledge of $f_m(n)$ and $b_m(n)$.

Problem 5.9

The GAL and LMS algorithm do share a common mathematical basis: The method of stochastic gradient processing. As such, they both follow a linear law of computational complexity. However, the structural complexity of the GAL algorithm is more elaborate than that of LMS algorithm. This can clearly be seen by, examining Table 5.2 which summarizes the GAL algorithm and comparing it with Table 5.1 which summarizes the LMS algorithm. Upon comparing the two algorithms we immediately see that, computationally speaking, the GAL algorithm is more demanding than the LMS algorithm.

Chapter 6

Problem 6.1



$$e(n) = d(n) - \hat{\mathbf{w}}_1^*(n)\mathbf{u}(n)$$
$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu\mathbf{u}(n)e^*(n)$$

Problem 6.2

For backward prediction, we have (assuming real data)

$$\hat{u}(n-n_0) = \hat{\mathbf{w}}^T(n)\mathbf{u}(n)$$

$$y(n) = u(n) - \hat{u}(n)$$
$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu\left[u(n-n_0) - \hat{u}(n-n_0)\right]\mathbf{u}(n)$$

Problem 6.3

The adaptive line enhancer minimizes the mean-square error, $\mathbb{E}[|e(n)|^2]$. For the problem at hand, the cost function $J = \mathbb{E}[|e(n)|^2]$ consists of the sum of three components:

(1) The average power of the primary input noise, denoted by σ_ν^2
(2) The average power of the noise at the filter output
(3) The average power of the contribution produced by the sinusoidal components at the input and the output of the filter

Let the peak value of the transfer function be denoted by a. We may then approximate the peak value of the weights as $2a/M$, where M is the length of the filter. On this basis, we may approximate the average power of the noise at the filter output as $(2a^2/M)\sigma_\nu^2$, which takes care of the term (2). For term (3), we assume that the input and output sinusoidal components subtract coherently, thereby yielding the average power $(A^2/2)(1-a)^2$. Hence, we may express the cost function J as

$$J(a) \approx \sigma_\nu^2 + \left(\frac{2a^2}{M}\right)\sigma_\nu^2 + \frac{A^2}{2}(1-a)^2$$

Differentiating J with respect to a and setting the result equal to zero yields the optimum scale factor

$$\begin{aligned} a_{\text{opt}} &= \frac{A^2}{A^2 + 4(\sigma_\nu^2/M)} \\ &= \frac{(A^2/2\sigma_\nu^2)(M/2)}{1 + (A^2/2\sigma_\nu^2)(M/2)} \\ &= \frac{(M/2)\,\text{SNR}}{1 + (M/2)\,\text{SNR}} \end{aligned}$$

where $\text{SNR} = A^2/2\sigma_\nu^2$.

Problem 6.4

The index of performance is

$$J(\mathbf{w}, K) = \mathbb{E}[\exp(2K)(n)], \qquad K = 1, 2, 3, \ldots$$

The estimation error $e(n)$ is

$$e(n) = d(n) - \mathbf{w}^T(n)\mathbf{u}(n) \tag{1}$$

where $d(n)$ is the desired response, $\mathbf{w}(n)$ is the tap-weight vector of the finite impulse response filter, and $\mathbf{u}(n)$ is the tap-input vector. In accordance with he multiple linear regression model for $d(n)$, we have

$$d(n) = \mathbf{w}_0^T(n)\mathbf{u}(n) \tag{2}$$

where $\mathbf{w}_0$ is the parameter vector, and $\nu(n)$ is a white-noise process of zero mean and variance σ_ν^2.

a)

The instantaneous gradient vector is

$$\begin{aligned}\hat{\nabla}(n,K) =& \frac{\partial}{\partial \mathbf{w}} J(\mathbf{w},K)\\ =& \frac{\partial}{\partial \mathbf{w}}[\exp(2K)(n)]\\ =& 2K\exp(2K-1)(n)\frac{\partial e(n)}{\partial \mathbf{w}}\\ =& -2K\exp(2K-1)(n)\mathbf{u}(n)\end{aligned}$$

Hence, we may express the new adaption rule for the estimate of the tap-weight vector as

$$\begin{aligned}\hat{\mathbf{w}}(n+1) =& \hat{\mathbf{w}}(n) - \frac{1}{2}\mu(-\hat{\nabla}(n,K))\\ =& \hat{\mathbf{w}}(n) + \mu K\mathbf{u}(n)\exp\left[(2K-1)(n)\right]\end{aligned} \tag{3}$$

b)

Eliminate $d(n)$ between Equations (1) and (2), with the estimate $\hat{\mathbf{w}}(n)$ used in place of $\mathbf{w}(n)$:

$$\begin{aligned}e(n) &= (\mathbf{w}_0 - \hat{\mathbf{w}}(n))^T\mathbf{u}(n) + \nu(n)\\ e(n) &= \epsilon^T(n)\mathbf{u}(n) + \nu(n)\\ e(n) &= \mathbf{u}^T(n)\epsilon(n) + \nu(n)\end{aligned} \tag{4}$$

Subtracting $\mathbf{w}_0$ from both sides of Equation (3)

$$\epsilon(n+1) = \epsilon(n) - \mu K\mathbf{u}(n)\exp(2K-1)(n) \tag{5}$$

where

$$\varepsilon(n) = \mathbf{w}_0 - \hat{\mathbf{w}}(n)$$

For the case when $\epsilon(n)$ is close to zero (i.e., $\hat{\mathbf{w}}(n)$ is close to $\mathbf{w}_0$), we may use Equation (4) to write:

$$\begin{aligned}\exp(2K-1)(n) &= [\mathbf{u}^T(n)\epsilon(n) + \nu(n)]^{2K-1} \\ &= \nu^{2K-1}(n)\left[1 + \frac{\mathbf{u}^T(n)\epsilon(n)}{\nu(n)}\right]^{2K-1} \\ &\approx \nu^{2K-1}(n)\left[1 + (2K-1)\frac{\mathbf{u}^T(n)\epsilon(n)}{\nu(n)}\right] \\ &= \nu^{2K-1}(n) + (2K-1)\mathbf{u}^T(n)\epsilon(n)\nu^{2(k-1)}(n) \qquad (6)\end{aligned}$$

Substitute Equation (6) into Equation (5):

$$\epsilon(n+1) \approx [\mathbf{I} - \mu K(2K-1)\nu^{2(K-1)}(n)\mathbf{u}(n)\mathbf{u}^T(n)]\epsilon(n) - \mu K\nu^{2K-1}\mathbf{u}(n)$$

Taking the expectation of both sides of this relation and recognizing that:
1) $\epsilon(n)$ is independent of $\mathbf{u}(n)$ by low pass filtering of the filter,
2) $\mathbf{u}(n)$ is independent of $\nu(n)$ by assumption, and
3) $\mathbf{u}(n)$ has zero mean,
we get

$$\mathbb{E}[\epsilon(n+1)] = \left\{\mathbf{I} - \mu K(2K-1)\mathbb{E}[\nu^{2(K-1)}(n)]\mathbf{R}\right\}\mathbb{E}[\epsilon(n)] \qquad (7)$$

where

$$\mathbf{R} = \mathbb{E}[\mathbf{u}(n)\mathbf{u}^T(n)]$$

c)
Let

$$\mathbf{R} = \mathbf{Q}\Lambda\mathbf{Q}^T \qquad (8)$$

Where Λ is the diagonal matrix of eigenvalues of $\mathbf{R}$, and $\mathbf{Q}$ is a matrix whose column vector equal the associated eigenvectors. Hence, substituting Equation (8) in Equation (7) and using

$$v(n) = \mathbf{Q}^T\mathbb{E}[\epsilon(n)]$$

we get

$$v(n+1) = \left\{\mathbf{I} - \mu K(2K-1)\mathbb{E}[\nu^{2(K-1)}(n)]\Lambda\right\}v(n)$$

That is, the i^{th} element of this equation is

$$v_i(n+1) = \left\{1 - \mu K(2K-1)\mathbb{E}[\nu^{2(K-1)}(n)]\lambda_i\right\} v_i(n) \tag{9}$$

Where $v_i(n)$ is the i^{th} element of $v(n)$, and $i = 1, 2, \ldots, M$. Solving the first-order difference Equation (9):

$$v_i(n) = \left\{1 - \mu K(2K-1)\mathbb{E}[\nu^{2(K-1)}(n)]\lambda_i\right\}^{n-1} v_i(0)$$

where $v_i(0)$ is the initial value of $v_i(n)$. Hence, for $v_i(n)$ to converge, we require that

$$|1 - \mu K(2K-1)\mathbb{E}[\nu^{2(K-1)}(n)]|\lambda_{\text{max}} < 1$$

where λ_{max} is the largest eigenvalue of $\mathbf{R}$. This condition on μ may be rewritten as

$$0 < \mu < \frac{2}{K(2K-1)\lambda_{\text{max}}\mathbb{E}[\nu^{2(K-1)}(n)]} \tag{10}$$

When this condition is satisfied, we find that

$$v_i(\infty) \to 0 \quad \text{for all } i$$

which prompts us to write $\epsilon(\infty) \to 0$ and, correspondingly, $\hat{\mathbf{w}}(\infty) \to \mathbf{w}_0$

d)

For $K = 1$, the results described in Equations (3), (7), and (10) reduce as follows, respectively,

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu\mathbf{u}(n)e(n)$$

$$\mathbb{E}[\epsilon(n+1)] = (\mathbf{I} - \mu\mathbf{R})\mathbb{E}[\epsilon(n)]$$

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

These results are recognized to be the same as those for the conventional LMS algorithm for real-valued data.

Problem 6.5

a)

We start with the equation

$$\mathbb{E}[\epsilon(n+1)] = (\mathbf{I} - \mu\mathbf{R})\mathbb{E}[\epsilon(n)] \tag{1}$$

where

$$\epsilon(n) = \mathbf{w}_0 - \hat{\mathbf{w}}(n)$$

We note that

$$\mathbf{m}(n) = \mathbb{E}[\hat{\mathbf{w}}(n)]$$

$$\mathbf{w}_0 = \mathbf{m}(\infty)$$

Hence, we may rewrite Equation (1) as

$$\mathbf{m}(n+1) - \mathbf{m}(\infty) = (\mathbf{I} - \mu\mathbf{R})(\mathbf{m}(n) - \mathbf{m}(\infty))$$

This is a first-order difference equation in $\mathbf{m}(n) - \mathbf{m}(\infty)$. Solving it, we get

$$\mathbf{m}(n) - \mathbf{m}(\infty) = (\mathbf{I} - \mu\mathbf{R})^n(\mathbf{m}(0) - \mathbf{m}(\infty))$$

where $\mathbf{m}(0)$ is the initial value of $\mathbf{m}(n)$. Hence,

$$\mathbf{m}(n) = \mathbf{m}(\infty) + (\mathbf{I} - \mu\mathbf{R})^n(\mathbf{m}(0) - \mathbf{m}(\infty)) \tag{2}$$

b)

Using the expansion

$$\mathbf{R} = \mathbf{Q}^H\Lambda\mathbf{Q}$$

and the fact that

$$\mathbf{Q}^H\mathbf{Q} = \mathbf{I},$$

then Equation (2) may be rewritten as:

$$\mathbf{m}(n) = \mathbf{m}(\infty) + (\mathbf{Q}^H\mathbf{Q} - \mu\mathbf{Q}^H\Lambda\mathbf{Q})^n(\mathbf{m}(0) - \mathbf{m}(\infty)) \tag{3}$$

Let

$$v(n) = \mathbf{Q}(\mathbf{m}(n) - \mathbf{m}(\infty))$$

$$v(0) = \mathbf{Q}(\mathbf{m}(0) - \mathbf{m}(\infty))$$

We may then rewrite Equation (3) as

$$v(n) = (\mathbf{I} - \mu\Lambda)^n v(0)$$

For an arbitrary $v(0)$, $v(n) \to 0$ and the LMS algorithm is therefore convergent in the mean value if and only if

$$|1 - \mu\lambda_{\text{max}}| < 1$$

that is

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

Problem 6.6

According to Equation (6.105) in the textbook, the time constant of the LMS algorithm is defined by

$$\tau_{\text{mse,k}} = \frac{1}{2\mu\lambda_k}, \qquad k = 1, 2, \ldots, m \tag{1}$$

where μ is the step-size parameter, λ_k is the k^{th} eigenvalue of the correlation matrix $\mathbf{R}$, and m is the number of taps in the LMS algorithm.

a)

Define the average eigenvalue of the LMS algorithm, as shown by

$$\lambda_{\text{av}} = \frac{1}{m}\sum_{k=1}^{m}\lambda_k \tag{2}$$

Then, using this formula in Equation (1), we may define the corresponding average time constant of the LMS algorithm as follows:

$$\tau_{\text{mse,av}} = \frac{1}{2\mu\lambda_{\text{av}}} \tag{3}$$

The practical condition for which Equation (3) holds reasonably well is when all the m eigenvalues of the LMS algorithm (i.e., the correlation matrix of the tap-inputs vectors, $\mathbf{R}$) are closely clustered together.

b)

When the smallest eigenvalue of the LMS algorithm is well separated from the remaining cluster of eigenvalues, the formula (3) loses its practical value. In such a scenario, the sensible approach is to pay more attention to the smallest eigenvalue rather than the other cluster of eigenvalues. Accordingly, the preferred approach is to define the time constant of the LMS algorithm as follows:

$$\tau_{\text{mse,max}} = \frac{1}{2\mu\lambda_{\text{min}}}$$

where λ_{min} is the smallest eigenvalue of the correlation matrix $\mathbf{R}$

Problem 6.7

a)

For convergence of the LMS algorithm in the mean value, we require that the step-size parameter μ satisfy the following condition:

$$0 < \mu < \frac{2}{\lambda_{\text{max}}} \tag{1}$$

where λ_{max} is the largest eigenvalue of the correlation matrix of the input vector $\mathbf{u}(n)$. For a white noise input of zero mean and variance σ^2, we have

$$\mathbf{R} = \sigma^2\mathbf{I}$$

For such an input, all the eigenvalues assume the common value σ^2. Correspondingly, Equation (1) takes the form

$$0 < \mu < \frac{2}{\sigma^2}$$

b)

The excess mean-square error produced by the LMS algorithm is

$$J_{\text{ex}} = J(\infty) - J_{\text{min}}$$

$$J_{\text{ex}} = \frac{2}{\sum_{i=1}^{M} \lambda_i} \tag{2}$$

With

$$\lambda_i = \sigma^2 \text{ for all, } i,$$

Equation (2) takes the form

$$J_{\text{ex}} = \frac{2}{M\sigma^2}$$

where M is the number of taps on the finite impulse response filter.

Problem 6.8

Case 1)

For this case, the true tap-weight vector of the finite-impulse response filter is

$$\mathbf{w}_0 = \mathbf{q}_2 = [-1, 1]^T$$

Here, under input $u_a(n)$, the convergence of the LMS algorithm is along a "slow" trajectory, traversing about halfway to the true parameterization of the filter in 200 iterations of the algorithm, starting from $\mathbf{w}(0) = [0, 0]^T$, as shown in Figure 1.

Next, the input signal is chosen as $u_b(n)$, for which the eigenvalue spread is evidenced by an increased eccentricity of the error-surface contours, as portrayed in Figure 2. Comparing Figure 2 with Figure 1, we see that in Case 1 the convergence of the LMS algorithm has been decelerated by an increase in the eigenvalue spread $\chi(\mathbf{R})$.

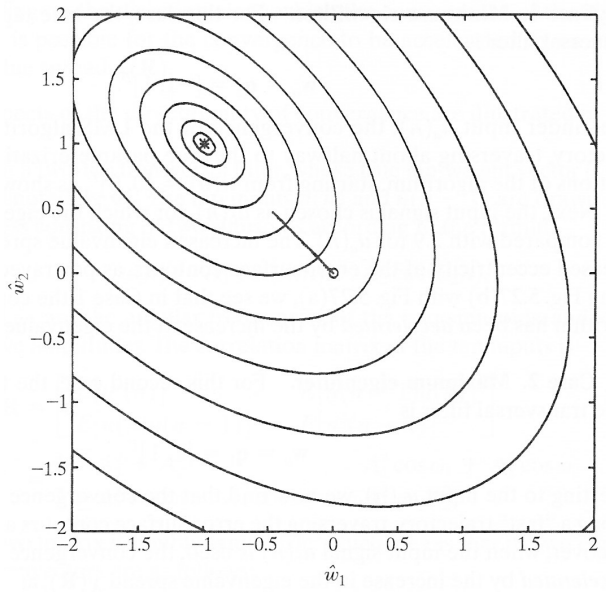


Figure 1: Convergence of the LMS algorithm, for a deterministic Sinusoidal process along "slow" eigenvector for input $u_a(n)$

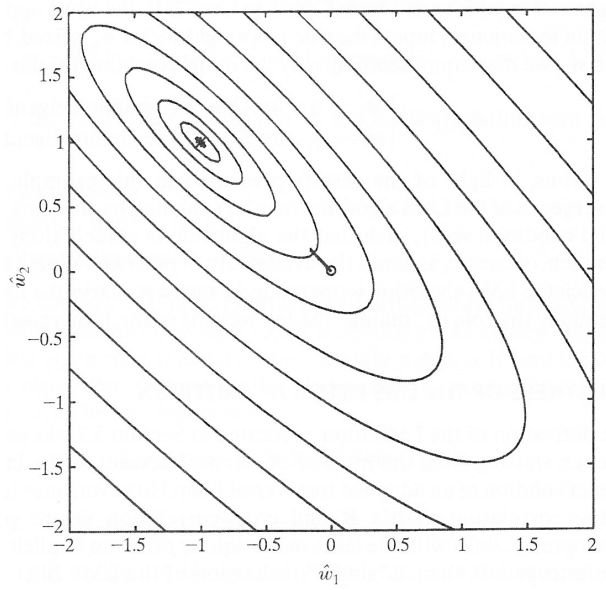


Figure 2: Convergence of the LMS algorithm, for a deterministic Sinusoidal process along "slow" eigenvector for input $u_b(n)$

Case 2)

For this second case, the true tap-weight vector of the finite-impulse response filter is

$$\mathbf{w}_0 = \mathbf{q}_1 = [1, 1]^T$$

Reverting to the input $u_a(n)$, we now find that convergence of the LMS algorithm is along a "fast" trajectory, traversing the error-surface contours as shown in Figure 3. Moreover, when the input signal $u_b(n)$ is used, convergence of the LMS algorithm is accelerated by the increase in eigenvalue spread $\chi(\mathbf{R})$, as shown in Figure 4.

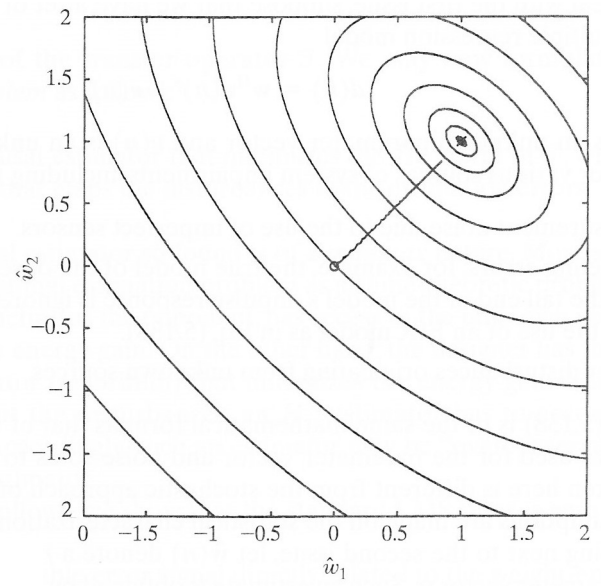


Figure 3: Convergence of the LMS algorithm, for a deterministic Sinusoidal process along "fast" eigenvector for input $u_a(n)$

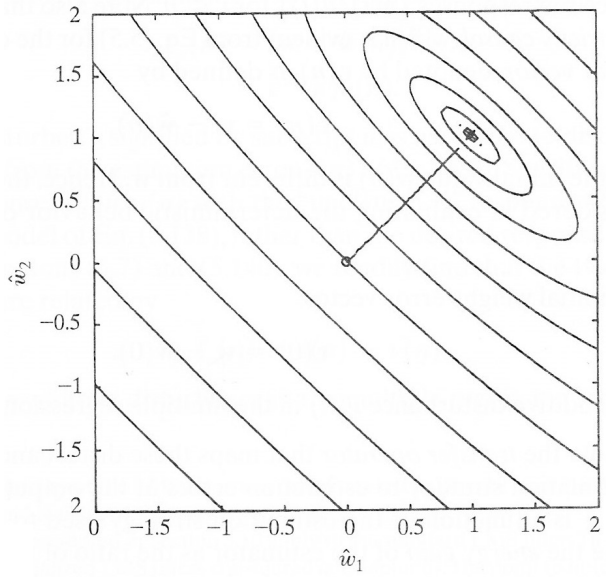


Figure 4: Convergence of the LMS algorithm, for a deterministic Sinusoidal process along "fast" eigenvector for input $u_b(n)$

In the example described here, the initial condition $\hat{\mathbf{w}}(0)$ is fixed, but the true tap-weight vector $\mathbf{w}_0$ is varied from Case 1 to Case 2. In the usual application of the LMS algorithm to stationary inputs, the true tap-weight vector $\mathbf{w}_0$ is fixed, but unknown. For some fixed

$\mathbf{w}_0$, we may equivalently specify the initial conditions for this example as follows:

$$\hat{\mathbf{w}}(0) = \begin{cases} \mathbf{w}_0 - \mathbf{q}_2 \text{ for Case 1 (minimum eigenfilter)} \\ \mathbf{w}_0 - \mathbf{q}_1 \text{ for Case 2 (maximum eigenfilter)} \end{cases}$$

Thus, in light of the results presented in this example, the directionality of convergence of the LMS algorithm may be exploited by choosing a suitable value for the initial condition $\hat{\mathbf{w}}(0)$, such that the algorithm is guided along a fast trajectory. This approach, of course, assumes the availability of prior knowledge about the environment in which the LMS algorithm is operating. In such a scenario, the LMs algorithm performs essentially the role of "tuning" the tap weights of the finite impulse-response filter.

Problem 6.9

We start with

$$\mathbf{K}(n+1) = (\mathbf{I} - \mu\mathbf{R})\mathbf{K}(n)(\mathbf{I} - \mu\mathbf{R}) + \mu^2 J_{\text{min}}\mathbf{R} \tag{1}$$

We are given

$$\mathbf{R} = \sigma_\nu^2\mathbf{I} \tag{2}$$

Hence, Equation (1) reduces to

$$\mathbf{K}(n+1) = (1 - \mu\sigma_\nu^2)^2\mathbf{K}(n) + \mu^2\sigma_\nu^2 J_{\text{min}}\mathbf{I}$$

For $n \to \infty$, we thus have

$$\mathbf{K}(\infty) = (1 - \mu\sigma_\nu^2)^2\mathbf{K}(\infty) + \mu^2\sigma_\nu^2 J_{\text{min}}\mathbf{I}$$

That is,

$$(2 - \mu\sigma_\nu^2)\mathbf{K}(\infty) = \mu J_{\text{min}}\mathbf{I} \tag{3}$$

For convergence in the mean square, we choose the step-size parameter μ to satisfy the condition

$$0 < \mu < \frac{2}{\sum_{i=1}^{M} \lambda_i}$$

For an input process $\mathbf{u}(n)$ whose correlation matrix $\mathbf{R}$ is described in Equation (2), we have

$$\lambda_i = \sigma_\nu^2 \text{ for } i = 1, 2, \ldots, M$$

where M is the number of taps in the finite impulse response filter. Hence,

$$0 < \mu < \frac{2}{M\sigma_\nu^2}$$

For large M, it follows that $(2-\mu\sigma_\nu^2)$ is closely approximated by 2, in which case Equation (3) takes on the approximate form

$$K(\infty) \approx \frac{\mu}{2} J_{\text{min}} \mathbf{I}$$

Problem 6.10

We start with Equation (6.58) from the textboook:

$$\epsilon_0(n+1) = (\mathbf{I} - \mu\mathbf{R})\epsilon_0(n) + \mu\mathbf{u}(n)e_0^*(n) \tag{1}$$

This difference equation has a solution in convolutional form

$$\epsilon_0(n) = \mathbf{H}(n) \star [\mu\mathbf{u}(n)e_0^*(n)] = \sum_{i=-\infty}^{\infty} \mathbf{H}(m)[\mu\mathbf{u}(n-m)e_0^*(n-m)]$$

where $\star$ denotes convolution, and the matrix

$$\mathbf{H}(n) = \begin{cases} (\mathbf{I} - \mu\mathbf{R})^{n-1} & \text{for } n > 0 \\ \mathbf{0} & \text{for } n \leq 0 \end{cases} \tag{2}$$

represents the causal symmetric $M \times M$ matrix impulse response of the system.

We are interested in the transmission of stationary stochastic signals, particularly the transformation of the input autocorrelation (use is made of the statistical independence of $\mathbf{u}(n)$ and $e_0^*(n)$)

$$\begin{aligned} \mathbf{F}^{(l)} &= \mathbb{E}\left[\mu\mathbf{u}(n)e_0(n)\mu\mathbf{u}^H(n-l)e_0^*(n-l)\right] \\ &= \mathbb{E}\left[e_0(n)e_0^*(n-l)\right]\mathbb{E}\left[\mu\mathbf{u}(n)\mu\mathbf{u}^H(n-l)\right] = \mu^2 J_{\text{min}}^{(l)}\mathbf{R}^{(l)} \end{aligned} \tag{3}$$

into the weight-error correlation matrix $\mathbf{K}_0 = \mathbb{E}\left[\epsilon_0(n)\epsilon_0^H(n)\right]$. Using standard results of linear systems theory, this is found as

$$\mathbf{K}_0 = \sum_m \sum_l \mathbf{H}(m)\mathbf{F}^{(l)}\mathbf{H}(m+l) = \sum_l \mathbf{T}^{(l)} \tag{4}$$

where

$$\mathbf{T}^{(l)} = \sum_m \mathbf{H}(m)\mathbf{F}^{(l)}\mathbf{H}(m+l)$$

In an attempt to evaluate the last sum in Equation (4), we encounter the difficulty that $\mathbf{H}(m)$ and $\mathbf{F}^{(l)}$, in general, do not commute, thus prohibiting the extraction of $\mathbf{F}^{(l)}$ from the sum. In fact, the last sum does not admit an explicit summation. Instead we show that for $\mu \to 0$, $\mathbf{T}^{(l)}$ satisfies the Lyapunov equation. To this end, we write the impulse response of Equation (2) in the recursive form [where the Dirac function $\delta(m)$ equals unity for $m = 0$ and zero elsewhere]

$$\begin{aligned}\mathbf{H}(m+1) &= (\mathbf{I} - \mu\mathbf{R})\mathbf{H}(m) + \delta(m)\mathbf{I} \\ &= \mathbf{H}(m)(\mathbf{I} - \mu\mathbf{R}) + \delta(m)\mathbf{I}\end{aligned} \tag{5}$$

which yeilds

$$\begin{aligned}\mathbf{T}^{(l)} &= \sum_m \mathbf{H}(m+l)\mathbf{F}^{(l)}\mathbf{H}(m+l+1) \\ &= \sum_m \left[(\mathbf{I} - \mu\mathbf{R})\mathbf{H}(m) + \delta(m)\mathbf{I}\right]\mathbf{F}^{(l)}[\mathbf{H}(m+l)(\mathbf{I} - \mu\mathbf{R}) + \delta(m+l)\mathbf{I}] \\ &= (\mathbf{I} - \mu\mathbf{R})\mathbf{T}^{(l)}(\mathbf{I} - \mu\mathbf{R}) + \mathbf{F}^{(l)}\mathbf{H}(l)(\mathbf{I} - \mu\mathbf{R}) + (\mathbf{I} - \mu\mathbf{R})\mathbf{H}(-l)\mathbf{F}^{(l)} + \delta(l)\mathbf{F}^{(l)}\end{aligned} \tag{6}$$

For small μ we may simplify Equation (6) by neglecting the terms $\mu\mathbf{R}\mathbf{T}^{(l)}\mu\mathbf{R}$, $-\mathbf{F}^{(l)}\mathbf{H}(l)\mu\mathbf{R}$, $-\mu\mathbf{R}\mathbf{H}(-l)\mathbf{F}^{(l)}$. and approximating the sum $\mathbf{F}^{(l)}\mathbf{H}(l)+\mathbf{H}(-l)\mathbf{F}^{(l)}+\delta(l)\mathbf{F}^{(l)}$ by $\mathbf{F}^{(l)}$. Thus we arrive at the Lyapunov equation

$$\mathbf{R}\mathbf{T}^{(l)} + \mathbf{T}^{(l)}\mathbf{R} = \mu^{-1}\mathbf{F}^{(l)} \tag{7}$$

and, after summation over l, using $\mathbf{K}_0 = \sum_l \mathbf{T}^{(l)}$, we get

$$\mathbf{R}\mathbf{K}_0 + \mathbf{K}_0\mathbf{R} = \sum_l \mu^{-1}\mathbf{F}^{(l)} = \mu \sum_l J_{\min}^{(l)}\mathbf{R}^{(l)} \tag{8}$$

which is the desired result.

Problem 6.11

Starting with Equation (6.80) in the textbook:

$$\nu_k(n) = (1-\mu\lambda_k)^n \nu_k(0) + \sum_{i=0}^{n-1} (1-\mu\lambda_k)^{n-1-i} \phi_k(i)$$

Applying the Expectation operator and noting that $\mathbb{E}[\phi_k(n)] = 0$ for all k, we get

$$\mathbb{E}[\nu_k(n)] = (1-\mu\lambda_k)^n \nu_k(0), \qquad k = 1, 2, \ldots, M$$

which is the desired result for the mean.

The mean-squared value of $\nu_k(n)$ is

$$\mathbb{E}[|\nu_k(n)|^2] = (1-\mu\lambda_k)^{2n} |\nu_k(0)|^2 + \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (1-\mu\lambda_k)^{m-1-i} (1-\mu\lambda_k)^{m-1-j} \mathbb{E}[\phi_k(i)\phi_k^*(j)]$$

where we have ignored the cross-product terms as the initial value of $\nu_k(0)$ is independent of $\phi_k(i)$. Next we note that

$$\mathbb{E}[\phi_k(i)\phi_k^*(j)] = \begin{cases} \mu^2 J_{\min}\lambda_k, & i = j \\ 0, & \text{otherwise} \end{cases}$$

Hence,

$$\begin{aligned} \mathbb{E}[|\nu_k(n)|^2] &= (1-\mu\lambda_k)^{2n} |\nu_k(0)|^2 + \sum_{i=0}^{n-1} (1-\mu\lambda_k)^{2(n-1-i)} \mu^2 J_{\min}\lambda_k \\ &= (1-\mu\lambda_k)^{2n} |\nu_k(0)|^2 + \mu^2 J_{\min}\lambda_k (1-\mu\lambda_k)^{2n-2} \sum_{i=0}^{n-1} (1-\mu\lambda_k)^{-2i} \end{aligned} \quad (1)$$

The sum

$$\sum_{i=0}^{n-1} (1-\mu\lambda_k)^{-2i}$$

represents a geometric series with the first term equal to 1, a geometric ratio of $(1-\mu\lambda_k)^{-2}$, and a total number of terms equal to n. We thus have

$$\sum_{i=0}^{n-1} (1-\mu\lambda_k)^{-2i} = \frac{1-(1-\mu\lambda_k)^{-2n}}{1-(1-\mu\lambda_k)^{-2}}$$

$$\sum_{i=0}^{n-1}(1-\mu\lambda_k)^{-2i} = \frac{(1-\mu\lambda_k)^2(1-\mu\lambda_k)^{-2n+2}}{\mu\lambda_k(2-\mu\lambda_k)}$$

Hence, Equation (1) reduces to

$$\mathbb{E}[|\nu_k(n)|^2] = (1-\mu\lambda_k)^{2n}|\nu_k(0)|^2 - \mu J_{\min}(1-\mu\lambda_k)^{2n-2}\left(\frac{(1-\mu\lambda_k)^2(1-\mu\lambda_k)^{-2n+2}}{\mu\lambda_k(2-\mu\lambda_k)}\right)$$

$$\mathbb{E}[|\nu_k(n)|^2] = \frac{\mu}{2-\mu\lambda_k}J_{\min} + (1-\mu\lambda_k)^{2n}\left(|\nu_k(0)|^2 - \frac{\mu J_{\min}}{2-\mu\lambda_k}\right)$$

Problem 6.12

a)

The mean-square deviation is defined by

$$D(n) = \mathbb{E}[\|\epsilon(n)\|^2]$$

$$D(n) \approx \mathbb{E}[\|\epsilon(n)\|^2] \text{ for small step-size } \mu$$

$$D(n) = \mathbb{E}[\|\mathbf{Q}\mathbf{V}(n)\|^2], \quad \mathbf{Q} = \text{orthonormal matrix}$$

$$D(n) = \mathbb{E}[\|\mathbf{V}(n))\|^2]$$

For $n \to \infty$, we thus have

$$D(\infty) = \mathbb{E}[\|\mathbf{V}(\infty))\|^2]$$

$$D(\infty) = \mathbb{E}\left[\sum_{k=1}^{M}|\nu_k(\infty)|^2\right]$$

$$D(\infty) = \sum_{k=1}^{M}\mathbb{E}\left[|\nu_k(\infty)|^2\right]$$

Equation (6.82) from the textbook tells us,

$$\mathbb{E}[|\nu_k(n)|^2] = \frac{\mu}{2-\mu\lambda_k}J_{\min} + (1-\mu\lambda_k)^{2n}\left(|\nu_k(0)|^2 - \frac{\mu J_{\min}}{2-\mu\lambda_k}\right)$$

Hence, with μ small compared to $2/\lambda_{\max}$,

$$\mathbb{E}[|\nu_k(\infty)|^2] = \frac{\mu J_{\min}}{2-\mu\lambda_k}$$

$$\mathbb{E}[|\nu_k(\infty)|^2] = \frac{\mu J_{\text{min}}}{2} \quad \text{for small } \mu$$

b)
From the Lyapunov equation derived in Problem (6.10), we have

$$\mathbf{R}\mathbf{K}_0(n) + \mathbf{K}_0(n)\mathbf{R} \approx \mu J_{\text{min}}\mathbf{R}, \quad \text{when } \mu \text{ is small} \tag{1}$$

where only the first term of the summation in the right-hand side of Equation (8) in the solution to problem (6.10) is retained. Taking the trace of both sides of Equation (1), and recognizing that

$$\text{tr}[\mathbf{R}\mathbf{K}_0(n)] = \text{tr}[\mathbf{K}_0(n)\mathbf{R}]$$

we get for $n = \infty$:

$$2\text{tr}[\mathbf{R}\mathbf{K}_0(\infty)] \approx \mu J_{\text{min}}\text{tr}[\mathbf{R}]$$

From Equation (6.90) of the textbook:

$$J_{\text{ex}}(\infty) = \text{tr}[\mathbf{R}\mathbf{K}_0(n)] = \frac{\mu}{2} J_{\text{min}}\text{tr}[\mathbf{R}]$$

Hence, the misadjustment is

$$\mathscr{M} \approx \frac{J_{\text{ex}}(\infty)}{J_{\text{min}}}$$

$$\mathscr{M} = \frac{\mu}{2}\text{tr}[\mathbf{R}]$$

Problem 6.13

The error correlation matrix $\mathbf{K}(n)$ is defined by

$$\mathbf{K}(n) = \mathbb{E}[\epsilon(n)\epsilon^H(n)]$$

The trace of $\mathbf{K}(n)$ is given by

$$\begin{aligned}\text{tr}[\mathbf{K}(n)] =& \text{tr}\left[\mathbb{E}\left[\epsilon(n)\epsilon^H(n)\right]\right] \\ =& \mathbb{E}\left[\text{tr}\left[\epsilon(n)\epsilon^H(n)\right]\right]\end{aligned}$$

Since

$$\text{tr}\left[\epsilon(n)\epsilon^H(n)\right] = \text{tr}\left[\epsilon^H(n)\epsilon(n)\right]$$

we may express $\mathbf{K}(n)$ as

$$\begin{aligned}\text{tr}[\mathbf{K}(n)] &= \text{tr}\left[\mathbb{E}\left[\epsilon^H(n)\epsilon(n)\right]\right] \\ &= \mathbb{E}\left[\text{tr}\left[\epsilon^H(n)\epsilon(n)\right]\right]\end{aligned}$$

The inner product $\epsilon^H(n)\epsilon(n)$ equals the squared norm of $\epsilon(n)$ which is a scalar. Hence

$$\text{tr}[\mathbf{K}(n)] = \mathbb{E}[\|\epsilon(n)\|^2] \tag{1}$$

From convergence analysis of the LMS algorithm, we have (see (1) in Problem 6.9)

$$\mathbf{K}(n+1) = (\mathbf{I} - \mu\mathbf{R})\mathbf{K}(n)(\mathbf{I} - \mu\mathbf{R}) + \mu^2 J_{\text{min}}\mathbf{R} \tag{2}$$

Initially, $\|\epsilon\|$ is so large that we may justifiably ignore the term $\mu^2 J_m in\mathbf{R}$, in which case Equation (2) may be approximated as

$$\mathbf{K}(n+1) = (\mathbf{I} - \mu\mathbf{R})\mathbf{K}(n)(\mathbf{I} - \mu\mathbf{R}) \quad \text{when } n \text{ is small} \tag{3}$$

Assuming that

$$\mathbf{R} = \sigma_u^2\mathbf{I}$$

we may further reduce Equation (3) to

$$\mathbf{K}(n+1) \approx \left(1 - \mu\sigma_u^2\right)^2 \mathbf{K}(n)$$

Thus, in light of Equation (1) we may write

$$\mathbb{E}[\|\epsilon(n+1)\|^2] \approx \left(1 - \mu\sigma_u^2\right)^2 \mathbb{E}[\|\epsilon(n)\|^2], \quad \text{for small } n$$

The convergence ratio is therefore approximately

$$\mathscr{C} = \frac{\mathbb{E}[\|\epsilon(n+1)\|^2]}{\mathbb{E}[\|\epsilon(n)\|^2]}$$

$$\mathscr{C} = \left(1 - \mu\sigma_u^2\right)^2$$

Problem 6.14

a)

The update equations for the tandem configuration of the LMS filters are formulated as follows:

$$e_1(n) = d(n) - \hat{\mathbf{w}}_1^H(n)\mathbf{u}(n)$$

$$\hat{\mathbf{w}}_1^H(n+1) = \hat{\mathbf{w}}_1^H(n) + \mu\mathbf{u}(n)e_1(n)$$

$$e_2(n) = e_1(n) - \hat{\mathbf{w}}_2^H(n)\mathbf{u}(n)$$

$$\hat{\mathbf{w}}_2^H(n+1) = \hat{\mathbf{w}}_2^H(n) + \mu\mathbf{u}(n)e_2(n)$$

This assumes that both filters are updated with the same step-size μ.

b)

As it can be seen from Figure (P6.2) as well as from the equation in part **a)** of this problem, adaptation of $\hat{\mathbf{w}}_1$ follows the exponential convergence of the LMS algorithm and its behavior is unaffected by $\hat{\mathbf{w}}_2$. The adaptation of $\hat{\mathbf{w}}_1$ is, however dependent on $\hat{\mathbf{w}}_1$. Let $\mathbf{R} = [\mathbf{u}(n)\mathbf{u}^H(n)] = \mathbf{Q}\Lambda\mathbf{Q}^H$, where $\mathbf{R}$ is the correlation matrix of the input vector. We use the eigendecomposition of $\mathbf{R}$. If we consider the rotated representation of the weighted vectors

$$\hat{\mathbf{v}}_1^1 = \mathbf{Q}^H\hat{\mathbf{w}}_1$$

$$\hat{\mathbf{v}}_2^1 = \mathbf{Q}^H\hat{\mathbf{w}}_2$$

and keep in mind that the signal is modelled by $d(n) = \mathbf{w}_0^H\mathbf{u}(n) + \nu(n)$ where $\nu(n)$ is a zero mean random noise, we may show that the total MSE= MSE_{11}+MSE_{12}+MSE_{22}.

Hence, $\text{MSE}_{ij} = \mathbb{E}[\epsilon_i(n)\epsilon_j^H(n)]$ where $(i,j) = 1,2$ is the order number of the LMS filters, and $\epsilon_i(n)$ =weight-error vector of LMS filter i. This result clearly indicated that the tandem system converges in the mean-square sense if MSE_{11}, MSE_{12}, and MSE_{22} converge in the mean-square.

Problem 6.15

From Equation (2.77) of the textbook, the cost function for a MVDR beamformer is

$$J = \sum_{k=0}^{m-1}\sum_{i=0}^{m-1}\mathbf{w}_k^*\mathbf{w}_i r(i-k) + \text{Re}\left[\lambda^*\left(\sum_{k=0}^{m-1}\mathbf{w}_k^*\exp(-\,\text{j}\,\theta_0 k) - g\right)\right] \tag{1}$$

where λ is the Lagrange multiplier defined in Equation (2.84)

$$\lambda = -\frac{2g}{\mathbf{s}^H(\theta_0)\mathbf{R}^{-1}\mathbf{s}(\theta_0)} \tag{2}$$

where g is a specified complex-valued gain, $\mathbf{R}$ is the correlation matrix of the input vector $\mathbf{u}(n)$, and $\mathbf{s}(\theta_0)$ is the steering vector directed at the target of interest. Combining Equations (1) and (2) and rewriting it in matrix from:

$$J = \mathbf{w}^H\mathbf{R}\mathbf{w} - \left(k^*\mathbf{w}^H\mathbf{s}(\theta_0) + k\mathbf{s}^H(\theta_0)\mathbf{w} + \text{Re}\,[g]\right) \tag{3}$$

where

$$k = \frac{g}{\mathbf{s}^H(\theta_0)\mathbf{R}\mathbf{s}(\theta_0)} \tag{5}$$

which acts merely as a scaling factor ensuring that the gain of the beamformer along the direction θ_0 is equal to g. Differentiating the cost function J with respect to the weight vector $\mathbf{w}$:

$$\frac{\partial J}{\partial \mathbf{w}} = 2\mathbf{R}\mathbf{w} - 2\mathbf{s}(\theta_0)k^* \tag{5}$$

Substituting $\mathbf{R} = \mathbf{u}(n)\mathbf{u}^H(n)$ into equation (5) where $\mathbf{u}(n)$ is the input vector:

$$\frac{\partial J(n)}{\partial \mathbf{w}} = 2\mathbf{u}(n)\mathbf{u}^H(n)\hat{\mathbf{w}} - 2\mathbf{s}(\theta_0)k^* \tag{6}$$

Hence, the LMS algorithm for the MVDR beamformer is

$$\begin{aligned}\hat{\mathbf{w}}(n+1) &= \hat{\mathbf{w}} - \frac{1}{2}\mu\frac{\partial J}{\partial \mathbf{w}} \\ &= \left[\mathbf{I} - \mu\mathbf{u}(n)\mathbf{u}^H(n)\right]\hat{\mathbf{w}}(n) + \mathbf{s}(\theta_0)k^*\end{aligned} \tag{7}$$

where μ is the step-size parameter.

Comparing this algorithm with the LMS algorithm for temporal processing, we see that $\mathbf{s}(\theta_0)k^*$ plays the product role of desired response and input vector.

As a check on the algorithm described in Equation (7), suppose the step-size parameter μ is small enough to justify using the direct method. We may then replace $\mathbf{u}(n)\mathbf{u}^H(n)$ by $\mathbf{R}$ and write

$$\hat{\mathbf{w}}(n+1) = \left[\mathbf{I} - \mu\mathbf{R}\right]\hat{\mathbf{w}}(n) + \mu k^*\mathbf{s}(\theta_0) \tag{8}$$

For $n \to \infty$, $\hat{\mathbf{w}}(n) \approx \hat{\mathbf{w}}(n+1)$, in which case Equation (4) reduces to

$$\begin{aligned} \mathbf{R}\hat{\mathbf{w}}(n) &\approx k^*\mathbf{s}(\theta_0), && \text{when } n \text{ is large} \\ & \text{or} \\ \hat{\mathbf{w}}(n) &\approx k^*\mathbf{R}^{-1}\mathbf{s}(\theta_0), && \text{when } n \text{ is large} \end{aligned}$$

The corresponding beamformer output along θ_0 is as, desired

$$\begin{aligned} \hat{\mathbf{w}}^H(n)\mathbf{s}(\theta_0) =& k^*\mathbf{s}^H(\theta_0)\mathbf{R}^{-1}\mathbf{s}(\theta_0) \\ =& g^* \quad \text{for large } n \end{aligned}$$

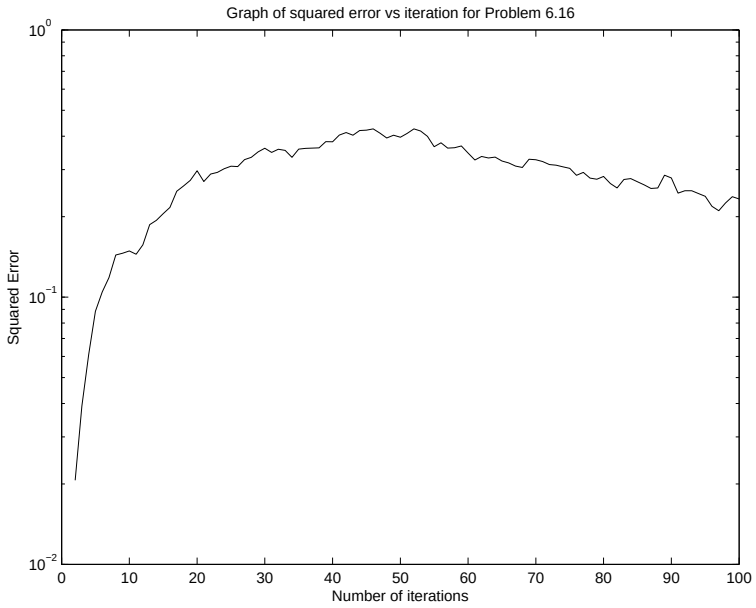
Problem 6.16

*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Adaptive Filter Theory 5e Solution Manual                                    %
%                                                                              %
% Chapter 6                                                                    %
% Question 19                                                                  %
%                                                                              %
% Program written to run on MATLAB 2010a (R)                                   %
%                                                                              %
% By Kelvin Hall                                                               %
% July 2, 2014                                                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear
clc
numberOfDatapoints=100; % try 300 to get a better picture of the process
numberOfRuns=100;       % increase this value to get smoothed out image

a=-0.99; %AR paramter
NoiseVariance=0.02; % the variance of the system noise given in the problem
mu=0.01; %LMS learning rate parameter it will interest the reader to also
                        %try also 0.5 and 0.75
```

*Correction: The problem is over defined by providing both a noise variance and an AR process variance. The noise variance as such can be ignored. However if the solution is found with the prescribed noise variance the resulting graphs will be nearly identical to those included in the solution manual.

```
stream = RandStream('mt19937ar','Seed',6);  % seed the random number
RandStream.setDefaultStream(stream);        % generator for reproducable
                                            % results
NoiseStandardDeviation=sqrt(NoiseVariance);

u=zeros(numberOfDatapoints,1);  % Allocate memory for input data stream
f=zeros(numberOfDatapoints,1);  % Allocate memory for error between
                                % prediction and resutls
g=zeros(numberOfDatapoints,1);  % allocate memory of squared averaged error

for k=1:numberOfRuns % Loop for performing the appropriate number of
                     % Monte Carlo simulations
    u(1)=0;          % Initialize the weights back to zero as well as the
    w=0;             % initial input to zero

    for n=2:numberOfDatapoints % The loop that does the data runs and
                               %filter updates

        u(n)=-a*u(n-1)+randn(1)*NoiseStandardDeviation;% update input data
                           % AR process described in the question
        f(n)=u(n)-w*u(n-1); % calculate the error in estimation
        w=w+mu*u(n-1)*f(n); % update the weights in the manner associated
                           % LMS based on the recient error in prediction
                           % of the last input
    end
    g=g+f.^2;              % accumulate squared error of estimation
end
g=g/numberOfRuns;          % normalize the accumulated error to reflect
                           % the average error based on the number of
                           % monte Carlo simulations completed
semilogy([1:numberOfDatapoints],g,'k') % plot the MSE vs iterations
title('Graph of squared error vs iteration for Problem 6.16')
xlabel('Number of iterations') % x-axis label
ylabel('Squared Error') % y-axis label
```



Problem 6.17

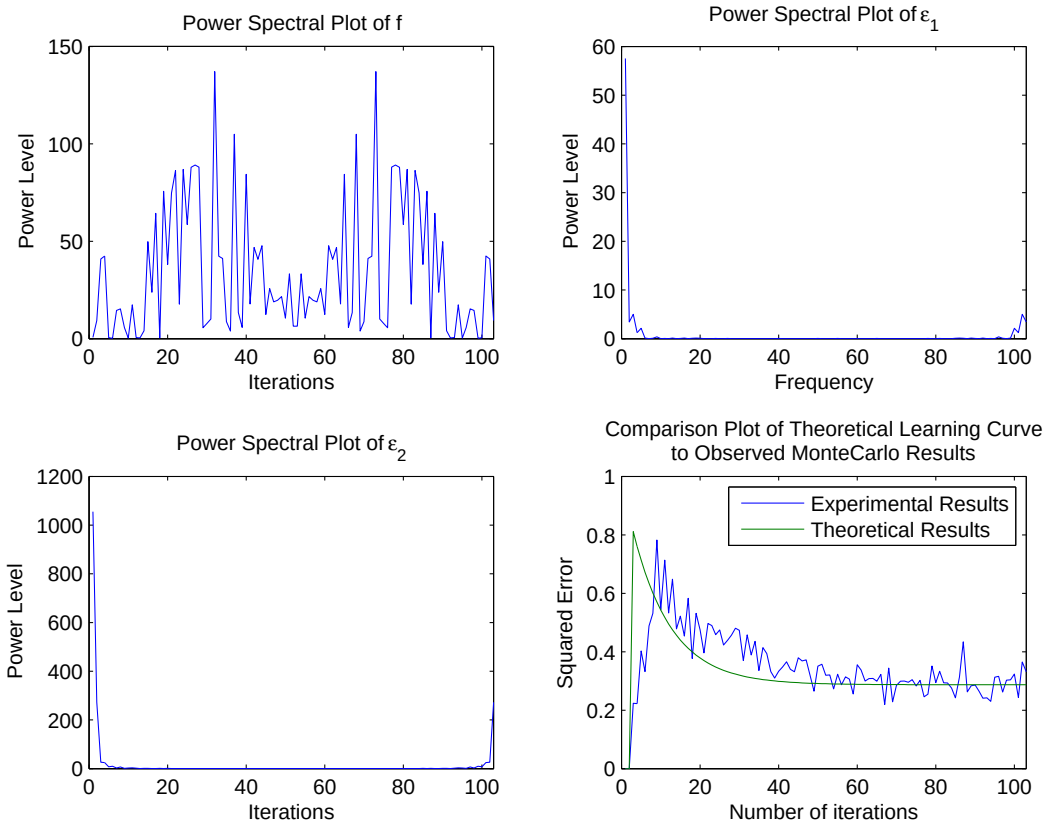
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Adaptive Filter Theory 5e Solution Manual                                      %
%                                                                               %
% Chapter 6                                                                     %
% Question 17                                                                   %
% Part a)                                                                       %
%                                                                               %
% Program written to run on MATLAB 2010a (R)                                    %
%                                                                               %
% By Kelvin Hall                                                                %
% June 30, 2014                                                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc
clear
LargeNumber=10000;  % as the sample variance approaches the actual variance
                    % of the system as the number of samples grow, the
                    % large nuber is the number of samples taken to
                    % calculate the sample variance.
NoiseVariance=10;    % An initial guess at the nois
NoiseStandardDeviation=sqrt(NoiseVariance);
```

```
u=zeros(LargeNumber+3,1); % Allocate memory for input data stream

aOne=0.1;       % AR paramters as given in the textbook
aTwo=-0.8;

Tolerance=0.001;   % the magnitude of the maximal error that is acceptable
MaxNumberOfImpovementCycles=100;  % The maximum number of updating cycles
error=Tolerance+1;  % an initial error larger then the tolerance of so at
                    % least one cycle is completed

mu=0.02;        % update learning parameter for LMS like learning

k=1;        % number of cycles completed itialized to the value one
while(abs(error)>Tolerance && k<MaxNumberOfImpovementCycles)
    NoiseStandardDeviation=sqrt(NoiseVariance);

    for n=3:(LargeNumber+3)
        u(n)=-aOne*u(n-1)-aTwo*u(n-2)+randn(1)*NoiseStandardDeviation;
    end

    error=(1-var(u(3:LargeNumber+3))); % Calculate the difference between
                    % process sample variance and desired process variance

    NoiseVariance=NoiseVariance+mu*error; % adjust the noise variance to
                                          % reduce error in process variance
    k=k+1;  % increment loop variable
end

if abs(error)>Tolerance % provide message explaining why solution wasn't found
    fprintf('sorry you may need to provide a closer starting value or')
    fprintf('run the system for more cycles')
    systemVariance=var(u(3:LargeNumber+3))
    NoiseVariance
else
    SampleProcessVariance=var(u(3:LargeNumber+3))
    NoiseVariance
    k
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Adaptive Filter Theory 5e Solution Manual                                %
%                                                                          %
% Chapter 6                                                                %
% Question 17                                                              %
% Parts c), d) and e)                                                      %
```

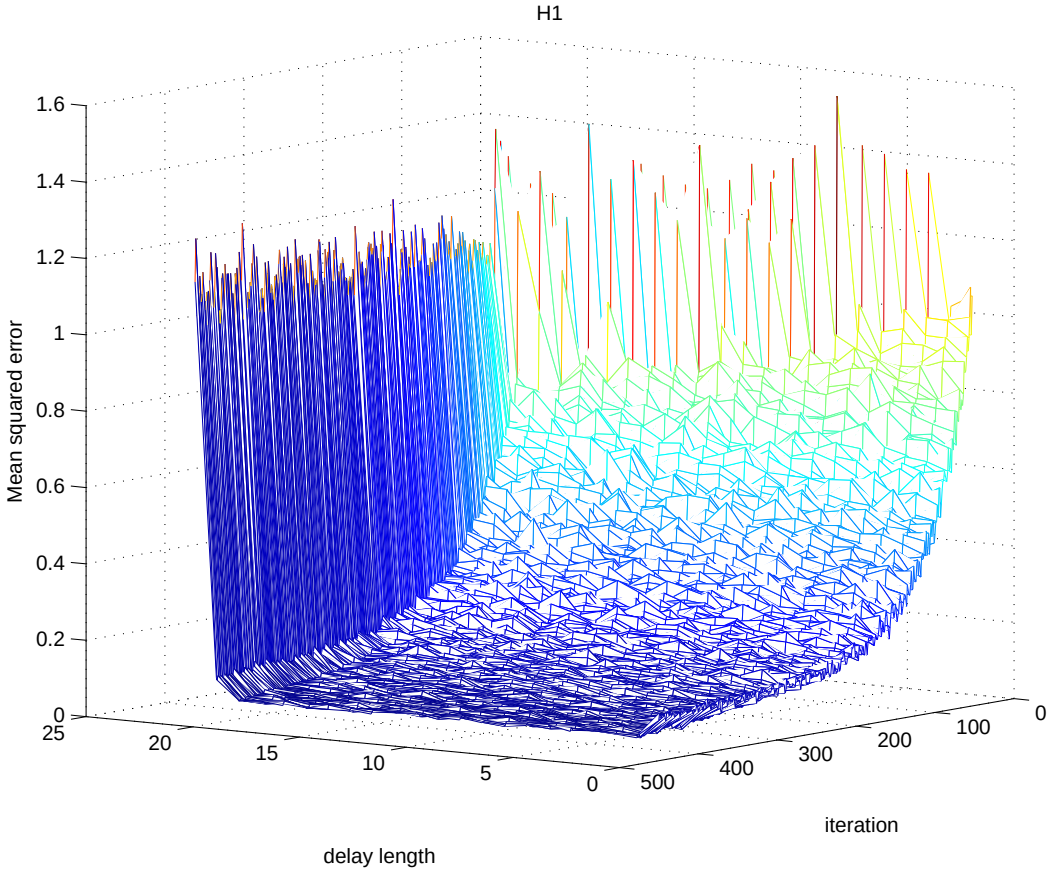
```
%                                                                       %
% Program written to run on MATLAB 2010a (R)                           %
%                                                                       %
% By Kelvin Hall                                                        %
% June 30, 2014                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear
clc
numberOfDatapoints=100; % try 300 to get a better picture of the process
numberOfRuns=100;   % The number of monteCarlo runs necissary for part d)
aOne=0.1;          %AR paramter
aTwo=-0.8;         %AR paramter as per textbook description

NoiseVariance=0.28;% The Noise variance found through part A of the problem
NoiseStandardDeviation=sqrt(NoiseVariance);
mu=0.05;

stream = RandStream('mt19937ar','Seed',30);  % seed the random number
RandStream.setDefaultStream(stream);        % generator for reproducable
                                            % results

u=zeros(numberOfDatapoints+3,1);    % Allocate memory for input data stream
f=zeros(numberOfDatapoints+3,1);    % Allocate memory for error between
                                    % prediction and resutls
epsilonOne=zeros(numberOfDatapoints+3,1);% Allocate memory for error
                                   % between found weight and AR paramter 1
epsilonTwo=zeros(numberOfDatapoints+3,1);% Allocate memory for error
                                   % between found weight and AR paramter 2
g=zeros(numberOfDatapoints+3,1);% allocate memory for squared-averaged error
J=zeros(numberOfDatapoints+3,1);% allocate memory for theoretical error

weights=zeros(2,numberOfDatapoints+3); % allocate memory for weights

for k=1:numberOfRuns % increment through the monte carlo runs of the problem
   W=zeros(2,numberOfDatapoints+3);    % reset the weights to zero between
                                       % each montecarlo run
   for n=3:numberOfDatapoints+3
        u(n)=aOne*u(n-1)+aTwo*u(n-2)+randn(1)*NoiseStandardDeviation;
             % create a new piece of input data to the filter

        f(n)=u(n)-W(1,n-1)*u(n-1)-W(2,n-1)*u(n-2);
             % Caluclate error between desired result and predicted results

        epsilonOne(n)=aOne-W(1,n-1);
             % calculate error between weight one and AR paramter one
```

```
        epsilonTwo(n)=aTwo-W(2,n-1);
             % calculate error between weight two and AR paramter two

        W(:,n)=W(:,n-1)+mu*f(n)*[u(n-1);u(n-2)];
                    %update the weights as per LMS
    end
    g=g+f.^2;       % accumulate squared error of estimation
end
g=g/numberOfRuns;  % accumulate squared error of estimation

for n=3:numberOfDatapoints+3
    J(n)=(1-NoiseVariance*(1+mu/2))*(1-mu)^(2*n)+NoiseVariance*(1+mu/2);
            %create the data for the graph of theoretical values for part
            %e) of the the experiment.
end

index=1:numberOfDatapoints+3; %the x axis of the plot


subplot(2,2,1) % first subplot showing the PSD of the error f, where it can
               % be seen that the error is similar to that of white noise
               % with approximatly even values across the frequency range
plot(index,(abs((fft(f)))).^2)
xlim([0 numberOfDatapoints+3])
title('Power Spectral Plot of f')
xlabel('Iterations') % x-axis label
ylabel('Power Level') % y-axis label

subplot(2,2,2) % second subplot showing the power spectral density of Error
               % epsilon 1, as there is a significant offset error the
               % frequency representation is highly noneven, with more
               % being towards the lower frequency
plot(index,(abs((fft(epsilonOne)))).^2)
xlim([0 numberOfDatapoints+3])
title('Power Spectral Plot of \epsilon_1')
xlabel('Frequency') % x-axis label
ylabel('Power Level') % y-axis label

subplot(2,2,3) % third subplot showing the power spectral density of Error
               % epsilon 2, as there is a significant offset error the
               % frequency representation is highly noneven, with more
               % being towards the lower frequency
plot(index,(abs((fft(epsilonTwo)))).^2)
xlim([0 numberOfDatapoints+3])
title('Power Spectral Plot of \epsilon_2')
```

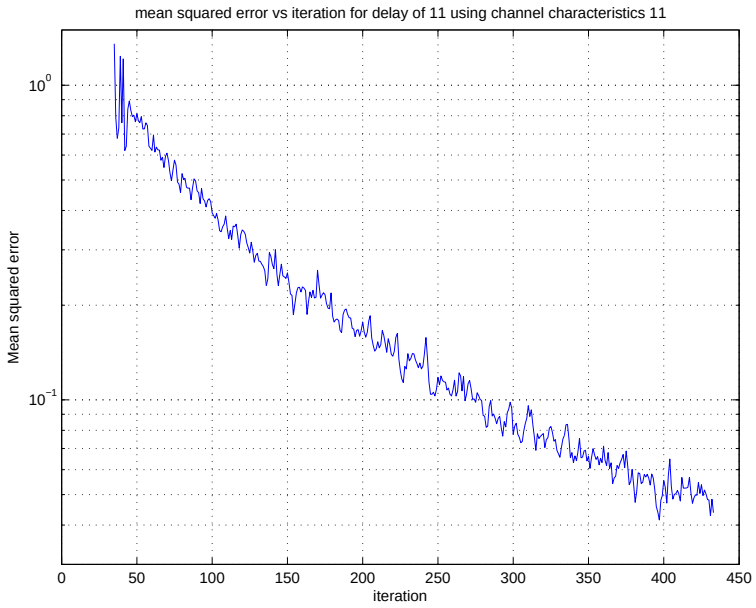
```
xlabel('Iterations') % x-axis label
ylabel('Power Level') % y-axis label

subplot(2,2,4) % Solution to part d) and e) on the same graph for
               % comparison purposes.
plot(index,g(1:numberOfDatapoints+3),index,J)
xlim([0 numberOfDatapoints+3])
ylim([0 1])
legend('Experimental Results','Theoretical Results')
xlabel('Number of iterations') % x-axis label
ylabel('Squared Error') % y-axis label
title({'Comparison Plot of Theoretical Learning Curve';...
                      'to Observed MonteCarlo Results'})
```



Problem 6.18



As it can be seen from the error surface for varying delay above the exact delay amount does not have a significant final value of the mean squared error. Therefore a delay of 11 is used in the subsequent part as it is approximately the lowest point.

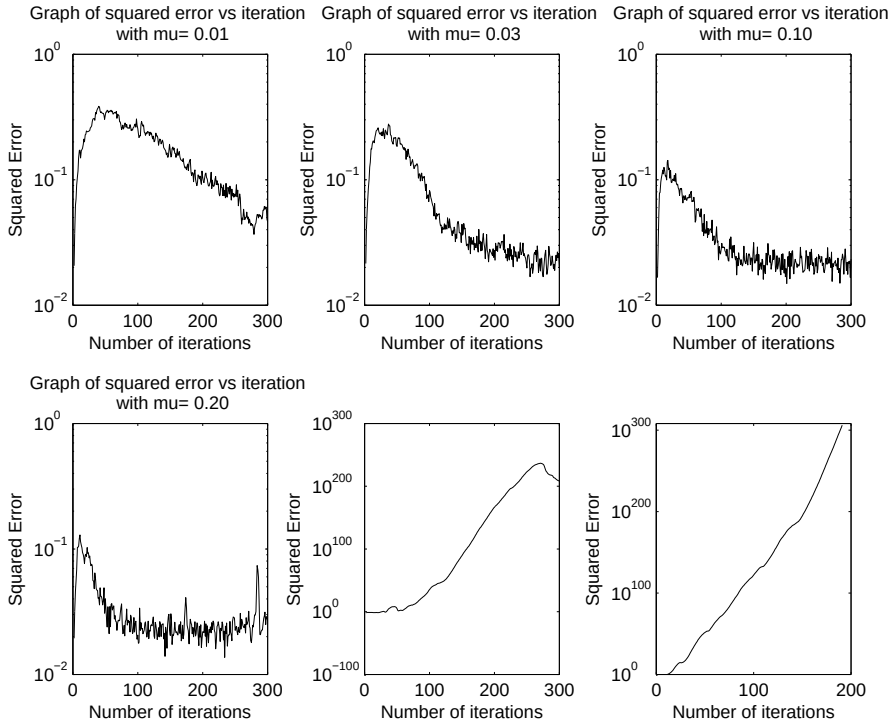


The results are similar for the other channel characteristics and can be shown by changing H1 to H2 or H3 in the code.

Problem 6.19

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Adaptive Filter Theory 5e Solution Manual                                    %
%                                                                              %
% Chapter 6                                                                    %
% Question 19                                                                  %
%                                                                              %
% Program written to run on MATLAB 2010a (R)                                   %
%                                                                              %
% By Kelvin Hall                                                               %
% July 2, 2014                                                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear
clc
numberOfDatapoints=300; % try 300 to get a better picture of the process
numberOfRuns=100;       % increase this value to get smoothed out image

a=-0.99; %AR paramter
```

```
NoiseVariance=0.02; % the variance of the system noise given in the problem
mu=[0.01, 0.03, 0.1, 0.2, 1 ,3]; %LMS learning rate parameter

stream = RandStream('mt19937ar','Seed',6);  % seed the random number
RandStream.setDefaultStream(stream);        % generator for reproducable
                                            % results
NoiseStandardDeviation=sqrt(NoiseVariance);

u=zeros(numberOfDatapoints,1);  % Allocate memory for input data stream
f=zeros(numberOfDatapoints,1);  % Allocate memory for error between
                                % prediction and resutls
g=zeros(numberOfDatapoints,1);  % allocate memory of squared averaged error

for m=1:size(mu,2)
    for k=1:numberOfRuns % Loop for performing the appropriate number of
                     % Monte Carlo simulations
    u(1)=0;          % Initialize the weights back to zero as well as the
    w=0;             % initial input to zero

    for n=2:numberOfDatapoints % The loop that does the data runs and
                               %filter updates

        u(n)=-a*u(n-1)+randn(1)*NoiseStandardDeviation;% update input data
                           % AR process described in the question
        f(n)=u(n)-w*u(n-1); % calculate the error in estimation
        w=w+mu(m)*u(n-1)*f(n); % update the weights in the manner associated
                           % LMS based on the recient error in prediction
                           % of the last input
    end
    g=g+f.^2;              % accumulate squared error of estimation
    end
    g=g/numberOfRuns;          % normalize the accumulated error to reflect
                           % the average error based on the number of
                           % monte Carlo simulations completed
    subplot(2,3,m)
    semilogy([1:numberOfDatapoints],g,'k') % plot the MSE vs iterations
    str=sprintf('Graph of squared error vs iteration \n with mu= %.2f',mu(m));
    title(str)
    xlabel('Number of iterations') % x-axis label
    ylabel('Squared Error') % y-axis label
end
```



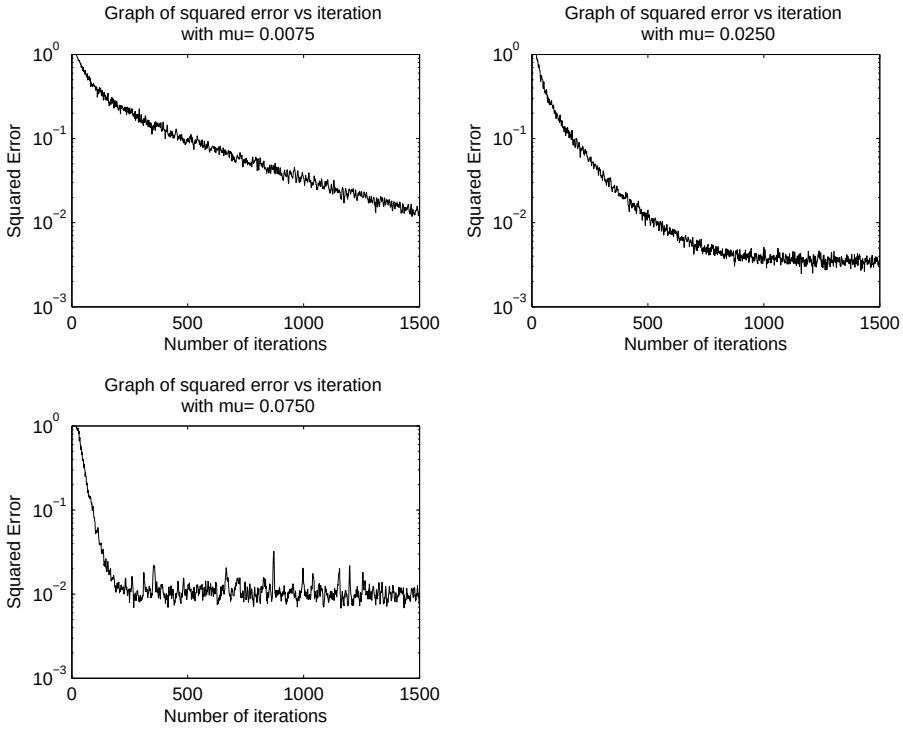
As it can be seen, experimentally, when $\mu \geq 1$ the system becomes unstable this result could have been predicted through adequate use of the LMS theory.

Problem 6.20

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Adaptive Filter Theory 5e Solution Manual                                     %
%                                                                               %
% Chapter 6                                                                     %
% Question 20                                                                   %
%                                                                               %
% Program written to run on MATLAB 2010a (R)                                    %
%                                                                               %
% By Kelvin Hall                                                                %
% July 2, 2014                                                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear
```

```
clc
numberOfDatapoints=1500; % try 300 to get a better picture of the process
numberOfRuns=200;       % increase this value to get smoothed out image

W=3.3; %eigen value spread parameter
NoiseVariance=0.001; % the variance of the system noise given in the problem
mu=[0.0075, 0.025, 0.075]; %LMS learning rate parameter

stream = RandStream('mt19937ar','Seed',3);  % seed the random number
RandStream.setDefaultStream(stream);        % generator for reproducable
                                            % results
NoiseStandardDeviation=sqrt(NoiseVariance);

u=zeros(numberOfDatapoints+14,1);  % Allocate memory for input data stream
r=zeros(numberOfDatapoints+14,1);  % Allocate memory for recieved data
f=zeros(numberOfDatapoints+14,1);  % Allocate memory for error between
                                % prediction and resutls
g=zeros(numberOfDatapoints+14,1);  % allocate memory of squared averaged error
h=[0.5*(1+cos(2*pi/W*(-1))),0.5*(1+cos(2*pi/W*(0))),0.5*(1+cos(2*pi/W*(1)))];

for m=1:size(mu,2)
    for k=1:numberOfRuns % Loop for performing the appropriate number of
                      % Monte Carlo simulations
    u=zeros(numberOfDatapoints,1);          % Initialize the weights back to zero as we
    W=zeros(11,1);             % initial weights to zero

    for n=3+11:numberOfDatapoints+3+11 % The loop that does the data runs and
                                %filter updates

        u(n)=binornd(1,0.5)*2-1;
        r(n)=u(n)*h(3)+u(n-1)*h(2)+u(n-2)*h(1)+randn(1)*NoiseStandardDeviation;
        f(n)=u(n-5)-W'*r(n:-1:n-10); % calculate the error in estimation
        W=W+mu(m)*r(n:-1:n-10)*f(n); % update the weights in the manner associated
                            % LMS based on the recient error in prediction
                            % of the last input
    end
    g=g+f.^2;               % accumulate squared error of estimation
    end
    g=g/numberOfRuns;           % normalize the accumulated error to reflect
                            % the average error based on the number of
                            % monte Carlo simulations completed
    subplot(2,2,m)
    semilogy([1:numberOfDatapoints+14],g,'k') % plot the MSE vs iterations
    str=sprintf('Graph of squared error vs iteration \n with mu= %.4f',mu(m));
    title(str)
    xlabel('Number of iterations') % x-axis label
```

```
    xlim([0 1500])
    ylim([1e-3 1])
    ylabel('Squared Error') % y-axis label
end
```

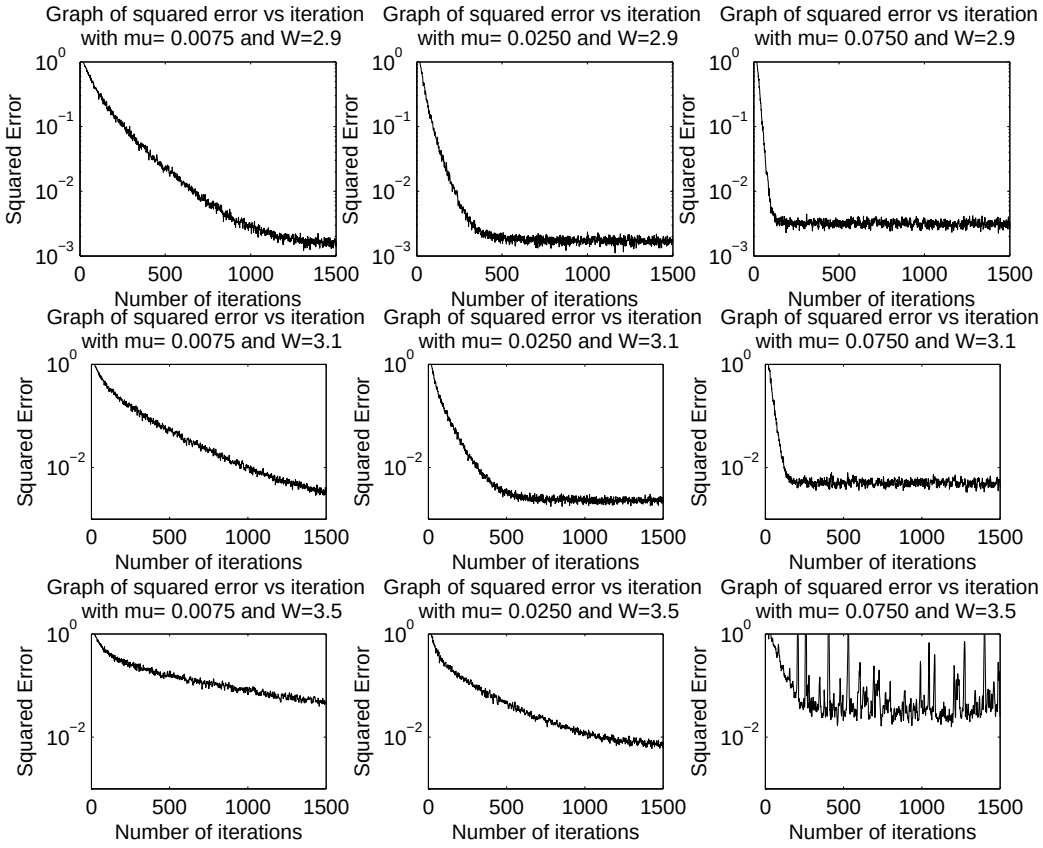


Problem 6.21

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Adaptive Filter Theory 5e Solution Manual                                      %
%                                                                               %
% Chapter 6                                                                     %
% Question 21                                                                   %
%                                                                               %
% Program written to run on MATLAB 2010a (R)                                    %
%                                                                               %
% By Kelvin Hall                                                                %
% July 2, 2014                                                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
clear
clc
numberOfDatapoints=1500; % try 300 to get a better picture of the process
numberOfRuns=200;       % increase this value to get smoothed out image

W=[2.9,3.1,3.5]; %eigen value spread parameter
NoiseVariance=0.001; % the variance of the system noise given in the problem
mu=[0.0075, 0.025, 0.075]; %LMS learning rate parameters

stream = RandStream('mt19937ar','Seed',3);  % seed the random number
RandStream.setDefaultStream(stream);        % generator for reproducable
                                            % results
NoiseStandardDeviation=sqrt(NoiseVariance);

u=zeros(numberOfDatapoints+14,1);  % Allocate memory for input data stream
r=zeros(numberOfDatapoints+14,1);  % Allocate memory for recieved data
f=zeros(numberOfDatapoints+14,1);  % Allocate memory for error between
                                % prediction and resutls
g=zeros(numberOfDatapoints+14,1);  % allocate memory of squared averaged error
for ell=1:3
    h=[0.5*(1+cos(2*pi/W(ell)*(-1))),0.5*(1+cos(2*pi/W(ell)*(0))),...
                                        0.5*(1+cos(2*pi/W(ell)*(1)))];
    for m=1:size(mu,2)
    for k=1:numberOfRuns % Loop for performing the appropriate number of
                    % Monte Carlo simulations
    u=zeros(numberOfDatapoints,1);          % Initialize the weights to zero

    for n=3+11:numberOfDatapoints+3+11 % The loop that does the data runs and
                                %filter updates

        u(n)=binornd(1,0.5)*2-1;
        r(n)=u(n)*h(3)+u(n-1)*h(2)+u(n-2)*h(1)+randn(1)*NoiseStandardDeviation;
        f(n)=u(n-5)-Weight'*r(n:-1:n-10); % calculate the error in estimation
        Weight=Weight+mu(m)*r(n:-1:n-10)*f(n); % update the weights in the manner assoc
                            % LMS based on the recient error in prediction
                            % of the last input
    end
    g=g+f.^2;               % accumulate squared error of estimation
    end
    g=g/numberOfRuns;           % normalize the accumulated error to reflect
                            % the average error based on the number of
                            % monte Carlo simulations completed
    subplot(3,3,(ell-1)*3+m)
    semilogy([1:numberOfDatapoints+14],g,'k') % plot the MSE vs iterations
    str=sprintf('Graph of squared error vs iteration \n with mu= %.4f and W=%.1f'...
                ,mu(m),W(ell));
```

```
    title(str)
    xlabel('Number of iterations') % x-axis label
    xlim([0 1500])
    ylim([1e-3 1])
    ylabel('Squared Error') % y-axis label
    end
end
```

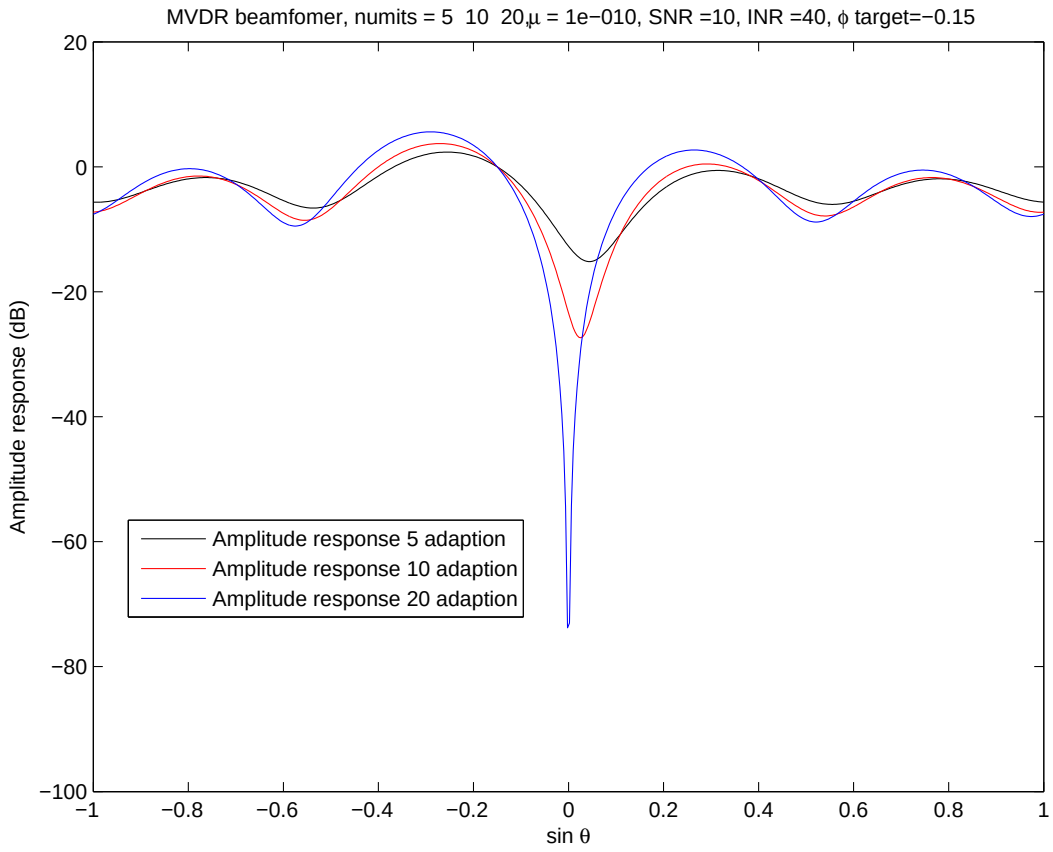


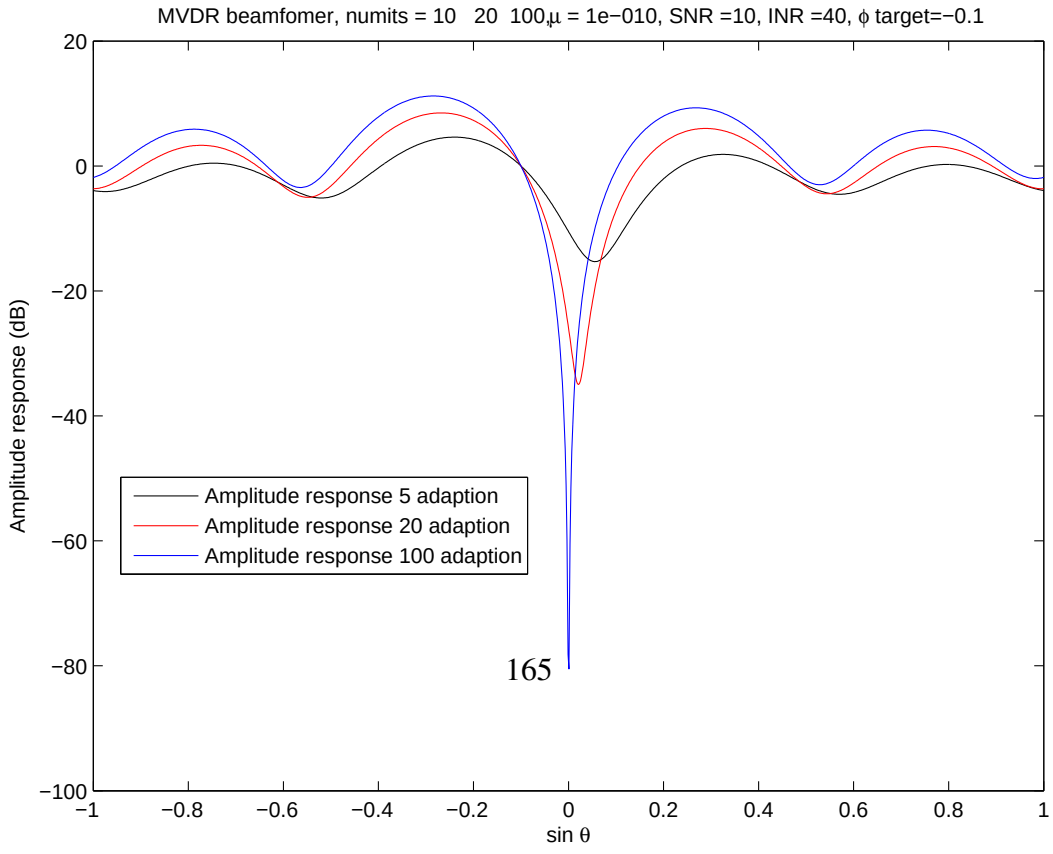
Problem 6.22

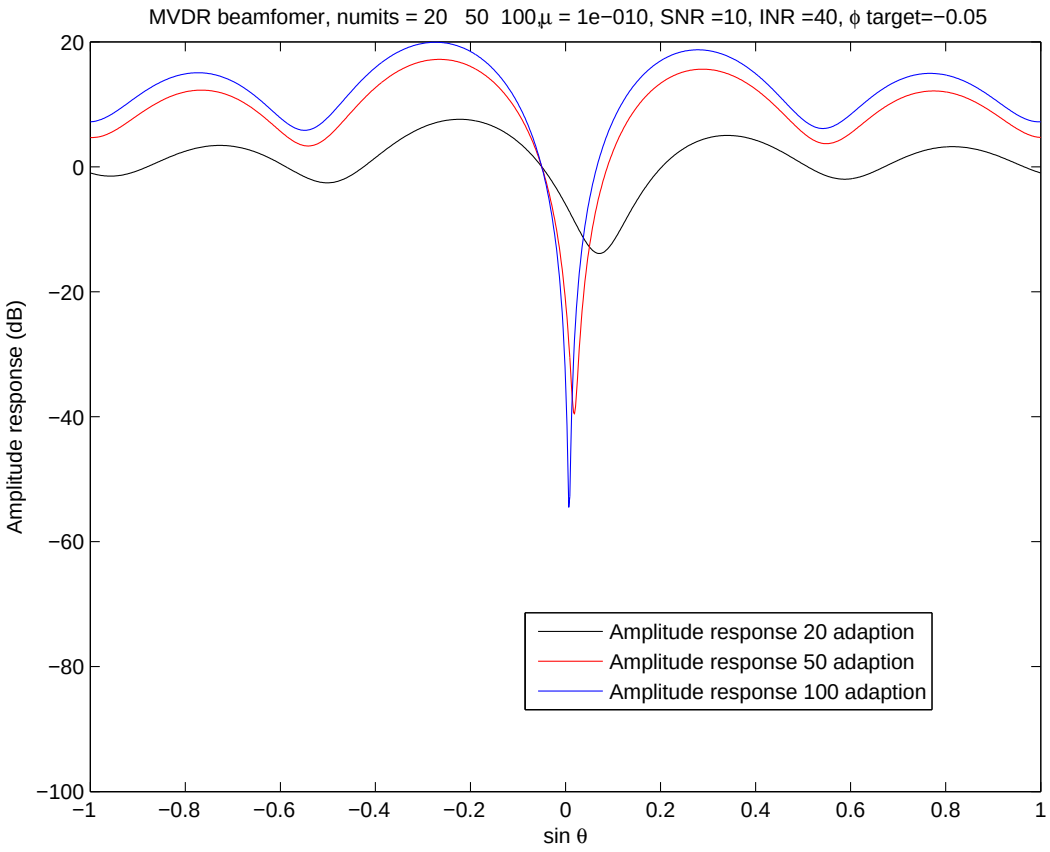
```
clear all;
close all;

mu = 1e-10; % step-size parameter
g = 1; % unity gain
```

```
p = 5; % number of sensors
Ninit = p; % nuber of smaples needed for initialization
Nsnaps = [20,50,100]; % number of snapshots or iteration

mean_v = 0; % white noise mean
var_v = 1; % white noise variance

SNRdB = 10; % target signal to noise ratio
INRdB = 40; % interference signal to noise ratio

sin_theta = [-0.05 0]; % location of signal and interference

numst = 1000; % resolution in spatial response
phi = pi.*sin_theta; % equivalent electrical angle
A = sqrt(var_v)*10.^([SNRdB INRdB]./20); % parameter for target/interference signal amp
% steering vector along electrical angle of look direction of interest
e = exp(-1j*[1:(p-1)]'*phi(1));
% setup input and output sequences

for n=1:3
Ndata = Ninit + Nsnaps(n); % total number of data
sig_x = A(1)*exp(1j*[1:p]*phi(1));
for i = 1:Ndata,
% random disturbances
    v_tmp = sqrt(var_v/2)*randn(2,p)+mean_v;
    v = v_tmp(1, :)+ 1j*v_tmp(2, :); % additive white noise of desired mean/variance
    Psi = 2*pi*rand; % uniform random phase on interference
    Xi(i, :) = sig_x + A(2)*exp(1j*[1:p]*phi(2) + Psi) + v;
end;
d = g*Xi(:,1);
u = diag(Xi(:,1))*(ones(Ndata,1)* e.')-Xi(:,2:p);

[W,xp] = lms(u,d,mu);
Wo = g- W * conj(e);
W = [Wo W];

% now, generate test vectors to compute spatially sampled response
W_H = conj(W(Ndata, :)); % Hemitian transpose of last one
st = linspace(-1,1,numst); % sine(theta) space
est = exp(-1j*pi*[0:(p-1)]'*st); % steering matrix
% amplitude response
P(:,n) = 20*log10(abs(W_H*est).^2);
end

p1 = plot(st,P(:,1),'k',st,P(:,2),'r',st,P(:,3),'b');
```

```
%axis([-1 1 min(P) max(P)]);
axis([-1 1 -100 20]);
xlabel('sin \theta');
ylabel('Amplitude response (dB)');
legend('Amplitude response 5 adaption','Amplitude response 10 adaption'...
       ,'Amplitude response 20 adaption');
title(['MVDR beamfomer, numits = ',num2str(Nsnaps),',\mu = ',num2str(mu),...
      ', SNR =',num2str(SNRdB),', INR =',num2str(INRdB),', \phi target=',...
      num2str(sin_theta(1))]);
```


MVDR beamfomer, numits = 5 10 20μ = 1e−010, SNR =10, INR =40, ϕ target=−0.15
Amplitude response (dB)
sin θ
Amplitude response 5 adaption
Amplitude response 10 adaption
Amplitude response 20 adaption


MVDR beamfomer, numits = 10 20 100μ = 1e−010, SNR =10, INR =40, ϕ target=−0.1
Amplitude response (dB)
sin θ
Amplitude response 5 adaption
Amplitude response 20 adaption
Amplitude response 100 adaption


MVDR beamfomer, numits = 20 50 100μ = 1e−010, SNR =10, INR =40, ϕ target=−0.05
Amplitude response (dB)
sin θ
Amplitude response 20 adaption
Amplitude response 50 adaption
Amplitude response 100 adaption
20
0
−20
−40
−60
−80
−100
−1
−0.8
−0.6
−0.4
−0.2
0
0.2
0.4
0.6
0.8
1

Chapter 7

Problem 7.1

a)

We note that

$$J(n+1) = \mathbb{E}\left[|e(n+1)|^2\right]$$

where

$$e(n+1) = d(n+1) - \mathbf{w}^H(n+1)\mathbf{u}(n+1)$$

Hence,

$$\begin{aligned} J(n+1) &= \mathbb{E}\left[\left|d(n+1) - \mathbf{w}^H(n+1)\mathbf{u}(n+1)\right|^2\right] \\ &= \sigma_d^2 - \left[\mathbf{w}(n) - \frac{1}{2}\mu(n)\nabla(n)\right]^H \mathbf{p} - \mathbf{p}^H\left[\mathbf{w}(n) - \frac{1}{2}\mu(n)\nabla(n)\right] \\ &\quad + \left[\mathbf{w}(n) - \frac{1}{2}\mu(n)\nabla(n)\right]^H \mathbf{R}\left[\mathbf{w}(n) - \frac{1}{2}\mu(n)\nabla(n)\right] \end{aligned}$$

Differentiating $J(n+1)$ with respect to $\mu(n)$, we get,

$$\begin{aligned} \frac{\partial J(n+1)}{\partial \mu(n)} =& \frac{1}{2}\left(\nabla^H(n)\mathbf{p}^H\nabla(n)\right) - \frac{1}{2}\left\{\nabla^H(n)\mathbf{R}\mathbf{w}(n) + \mathbf{w}^H(n)\mathbf{R}\nabla(n)\right\} \\ &+ \mu(n)\nabla^H(n)\mathbf{R}\nabla(n) \end{aligned}$$

Setting this differential equal to zero and solving for $\mu(n)$, we get

$$\mu_0(n) = \frac{\frac{1}{2}\left\{\nabla^H(n)\mathbf{R}\mathbf{w}(n) + \mathbf{w}^H(n)\mathbf{R}\nabla(n) - \nabla^H(n)\mathbf{p} - \mathbf{p}^H\nabla(n)\right\}}{\nabla^H(n)\mathbf{R}\nabla(n)} \tag{1}$$

b)

We are given that

$$\nabla(n) = 2\left(\mathbf{R}\mathbf{w}(n) - \mathbf{p}\right)$$

Hence, Equation (1) simplifies to

$$\mu_0(n) = \frac{\nabla^H(n)\nabla(n)}{\nabla^H(n)\mathbf{R}\nabla(n)}$$

Using the instantaneous estimates of $\mathbf{R}$ and $\nabla(n)$:

$$\hat{\mathbf{R}} = \mathbf{u}(n)\mathbf{u}^H(n)$$

$$\begin{aligned}\nabla(n) &= 2\left[\mathbf{u}(n)\mathbf{u}^H(n)\hat{\mathbf{w}}(n) - \mathbf{u}(n)d^*(n)\right] \\ &= -2\mathbf{u}(n)e^*(n)\end{aligned}$$

we find that the corresponding value of $\mu_0(n)$ is

$$\begin{aligned}\mu_0(n) &= \frac{\mathbf{u}^H(n)\mathbf{u}(n)\left|e(n)\right|^2}{\left(\mathbf{u}^H(n)\mathbf{u}(n)\right)^2\left|e(n)\right|^2} \\ &= \frac{1}{\mathbf{u}^H(n)\mathbf{u}(n)} \\ &= \frac{1}{\left\|\mathbf{u}(n)\right\|^2}\end{aligned}$$

Correspondingly, we have

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{\tilde{\mu}}{\left\|\mathbf{u}(n)\right\|^2}\mathbf{u}(n)e^*(n)$$

which is recognized as the normalized LMS algorithm.

Problem 7.2

$$\begin{aligned}\delta\hat{\mathbf{w}}(n+1) &= \hat{\mathbf{w}}(n+1) - \hat{\mathbf{w}}(n) \\ &= \frac{1}{\left\|\mathbf{u}(n)\right\|^2}\mathbf{u}^H(n)\mathbf{u}(n)\left[\hat{\mathbf{w}}(n+1) - \hat{\mathbf{w}}(n)\right]\end{aligned}$$

$$\begin{aligned}\delta\hat{\mathbf{w}}(n+1) &= \frac{1}{\|\mathbf{u}(n)\|^2}\mathbf{u}(n)\left[\mathbf{u}^H(n)\hat{\mathbf{w}}(n+1) - \mathbf{u}^H(n)\hat{\mathbf{w}}(n)\right] \\ &= \frac{1}{\|\mathbf{u}(n)\|^2}\mathbf{u}(n)\left[d^*(n) - \mathbf{u}^H(n)\hat{\mathbf{w}}(n)\right] \\ &= \frac{1}{\|\mathbf{u}(n)\|^2}\mathbf{u}(n)e^*(n)\end{aligned}$$

Problem 7.3

The second statement is the correct one. The justification is obvious from the solution to Problem 7.2; see also Equation (7.10) of the textbook.
Since

$$\mathbf{u}(n) = [u(n), u(n-1), \ldots, u(n-M+1)]^T$$

$$\hat{\mathbf{w}}(n) = [w_0(n), w_1(n), \ldots, w_{M-1}(n)]^T$$

$$\hat{\mathbf{w}}(n+1) = [w_0(n+1), w_1(n+1), \ldots, w_{M-1}(n+1)]^T$$

Replace all of these terms into the NLMS formula

$$\hat{\mathbf{w}}(n+1) - \hat{\mathbf{w}}(n) = \frac{\tilde{\mu}}{\|\mathbf{u}(n)\|^2}\mathbf{u}(n)e^*(n)$$

and look at each element in the vector. We then find that the correct answer is

$$\hat{w}_k(n+1) - \hat{w}_k(n) = \frac{\tilde{\mu}}{\|\mathbf{u}(n)\|^2}u(n-k)e^*(n) \qquad k = 0, 1, \ldots, M-1$$

Problem 7.4

$$\begin{aligned}\delta\hat{\mathbf{w}}(n+1) &= \hat{\mathbf{w}}(n+1) - \hat{\mathbf{w}}(n) \\ &= \mathbf{A}^{-1}(n)\mathbf{A}(n)\left[\hat{\mathbf{w}}(n+1) - \hat{\mathbf{w}}(n)\right] \\ &= \mathbf{A}^{-1}(n)\left[\mathbf{A}(n)\hat{\mathbf{w}}(n+1) - \mathbf{A}(n)\hat{\mathbf{w}}(n)\right] \\ &= \mathbf{A}^{-1}(n)\left[\mathbf{d}(n) - \mathbf{A}(n)\hat{\mathbf{w}}(n)\right]\end{aligned}$$

which from Equation (7.44) in the textbook is equal to,

$$\delta\hat{\mathbf{w}}(n+1) = \mathbf{A}^{-1}(n)\left[\frac{1}{2}\mathbf{A}(n)\mathbf{A}^H(n)\lambda\right]$$

$$\begin{aligned}\delta\hat{\mathbf{w}}(n+1) &= \frac{1}{2}\mathbf{A}^{-1}(n)\mathbf{A}(n)\mathbf{A}^{H}(n)\lambda \\ &= \frac{1}{2}\mathbf{A}^{H}(n)2\left(\mathbf{A}(n)\mathbf{A}^{H}(n)\right)^{-1}\mathbf{e}(n) \\ &= \mathbf{A}^{H}(n)\left(\mathbf{A}(n)\mathbf{A}^{H}(n)\right)^{-1}\mathbf{e}(n)\end{aligned}$$

where in the second line, we used Equation (7.46) from the textbook.

Problem 7.5

	Virtues	Limitations
LMS	Simple and stable H^{∞} robust (see Chapter 11) Model-independent	Slow convergence. Learning rate must have the dimension of inverse power.
NLMS	Convergent in the mean sense. Stable in mean-square sense. H^{∞} robust.(see Chapter 11) Invariant to scaling factor of input. Dimensionless step-size. (special case of APAF).	Little increase in computational complexity. (compared to LMS)
APAF	More accurate due to the use of more information. Semi-batch learning. Faster convergence.	Increased computational complexity.

Problem 7.6

Provided we show, under different scaling situations, that the learning rules (weight-update formula) are the same, then we can say the final solutions are the same under the same initial conditions.

1. NLMS, unscaled:

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{\tilde{\mu}}{\|\mathbf{u}(n)\|^{2}}\mathbf{u}(n)e^{*}(n)$$

NLMS, scaled:

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{\tilde{\mu}}{\|a\mathbf{u}(n)\|^2} a\mathbf{u}(n)\left[\mathbf{w}^H(n+1)a\mathbf{u}(n) - a\hat{\mathbf{w}}^H(n)\mathbf{u}(n)\right]$$

$$\begin{aligned}\hat{\mathbf{w}}(n+1) &= \hat{\mathbf{w}}(n) + \frac{\tilde{\mu}}{a^2\|\mathbf{u}(n)\|^2} a^2\mathbf{u}(n)\left[\mathbf{w}^H(n+1)\mathbf{u}(n) - \hat{\mathbf{w}}^H(n)\mathbf{u}(n)\right] \\ &= \hat{\mathbf{w}}(n) + \frac{\tilde{\mu}}{\|\mathbf{u}(n)\|^2}\mathbf{u}(n)\left[d(n) - \hat{\mathbf{w}}^H(n)\mathbf{u}(n)\right] \\ &= \hat{\mathbf{w}}(n) + \frac{\tilde{\mu}}{\|\mathbf{u}(n)\|^2}\mathbf{u}(n)e^*(n)\end{aligned}$$

which is the same as unscaled NLMS

2. APAF
Denoting

$$\mathbf{A}_{\text{scaled}} = [a\mathbf{u}(n), a\mathbf{u}(n-1), \ldots, a\mathbf{u}(n-N+1)]$$

$$\mathbf{d}_{\text{scaled}} = \mathbf{A}_{\text{scaled}}\hat{\mathbf{w}}(n+1)$$

Unscaled case:

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mathbf{A}^H(n)\left(\mathbf{A}(n)\mathbf{A}^H(n)\right)^{-1}\mathbf{e}(n)$$

Scaled case:

$$\begin{aligned}\hat{\mathbf{w}}(n+1) - \hat{\mathbf{w}}(n) &= \mathbf{A}^{-1}_{\text{scaled}}(n)\mathbf{A}_{\text{scaled}}(n)\left[\hat{\mathbf{w}}(n+1) - \hat{\mathbf{w}}(n)\right] \\ &= \mathbf{A}^{-1}_{\text{scaled}}(n)\left[\mathbf{A}_{\text{scaled}}(n)\hat{\mathbf{w}}(n+1) - \mathbf{A}_{\text{scaled}}(n)\hat{\mathbf{w}}(n)\right] \\ &= \mathbf{A}^{-1}_{\text{scaled}}(n)\left[\mathbf{d}_{\text{scaled}}(n) - \mathbf{A}_{\text{scaled}}(n)\hat{\mathbf{w}}(n)\right] \\ &= \mathbf{A}^{-1}_{\text{scaled}}(n)\left[\frac{1}{2}\mathbf{A}_{\text{scaled}}(n)\mathbf{A}^H_{\text{scaled}}(n)\lambda\right] \\ &= \frac{1}{2}\mathbf{A}^H_{\text{scaled}}(n)2\left[\mathbf{A}_{\text{scaled}}(n)\mathbf{A}^H_{\text{scaled}}(n)\right]^{-1}\mathbf{e}_{\text{scaled}}(n) \\ &= a\mathbf{A}^H(n)\left[a^2\mathbf{A}(n)\mathbf{A}^H(n)\right]^{-1}a\mathbf{e}(n) \\ &= \mathbf{A}^H(n)\left[\mathbf{A}(n)\mathbf{A}^H(n)\right]^{-1}\mathbf{e}(n)\end{aligned}$$

which is the same as the unscaled APAF

Problem 7.7

$$u(n) = \sum_{k=1}^{N-1} w_k^* u(n-k) + \nu(n)$$

a)

The algorithms can be APAF formulated as follows:
Step (1): Initialize AR coefficients $w_1, w_2, \ldots, w_{N-1}$ as random values
Step (2): Suppose:

$$\mathbf{w}(n) = \begin{bmatrix} w_1(n) & \ldots & w_{N-1}(n) \end{bmatrix}^H$$

$$\mathbf{u}(n) = \begin{bmatrix} u(n-1) & \ldots & u(n_N+1) \end{bmatrix}^H$$

Hence, we have

$$u(n) = \hat{\mathbf{w}}^H(n)\mathbf{u}(n-1) + \nu(n)$$

Suppose

$$\mathbf{A}^H(n) = \begin{bmatrix} \mathbf{u}(n-1) & \ldots & \mathbf{u}(n-N+1) \end{bmatrix}^H$$

Calculate

$$\mathbf{e}(n) = \mathbf{u}(n) - \mathbf{A}^H(n)\hat{\mathbf{w}}(n)$$

if $\|\mathbf{e}(n)\|^2 \leq \alpha$ (α is a predefined small positive value)
go to end
else go to step (3)
Step(3): update $\hat{\mathbf{w}}$

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \tilde{\mu}\mathbf{A}^H(n)\left[\mathbf{A}(n)\mathbf{A}^H(n)\right]^{-1}\mathbf{e}(n)$$

go to step (2)

b)

$$\begin{aligned}\phi(n) &= \left[\mathbf{I} - \mathbf{A}^H(n)\left(\mathbf{A}(n)\mathbf{A}^H(n)\right)^{-1}\mathbf{A}^H(n)\right]\mathbf{u}(n) \\ &= \mathbf{u}(n) - \mathbf{A}^H(n)\left(\mathbf{A}(n)\mathbf{A}^H(n)\right)^{-1}\mathbf{A}^H(n)\mathbf{u}(n) \\ &= \mathbf{u}(n) - \mathbf{P}\mathbf{u}(n)\end{aligned}$$

Where $\mathbf{P}$ is the projection operator

Geometrically, the distance (difference) between $\mathbf{u}(n)$ and its projected vector $\mathbf{Pu}(n)$ is the error (noise vector). Hence, with $\nu(n)$ assumed to be white Gaussian of zero mean, it follows that the elements of the vector $\phi(n)$ are themselves zero-mean white Gaussian process.

Problem 7.8

From Equation (7.19) of the textbook,

$$\tilde{\mu}_{\text{opt}} = \frac{\text{Re}\left\{\mathbb{E}\left[\xi_u(n)e^*(n)/\left\|\mathbf{u}(n)\right\|^2\right]\right\}}{\mathbb{E}\left[\left|e(n)\right|^2/\left\|\mathbf{u}(n)\right\|^2\right]}$$

Assuming that the undisturbed estimation error $\xi(n)$ is equal to the disturbed estimation error (i.e., normal error signal) $e(n)$, then we may put

$$\tilde{\mu}_{\text{opt}} \approx 1$$

in which case Equation (7.18) from the textbook provides the bounds on $\tilde{\mu}$ as

$$0 < \tilde{\mu} < 2$$

Also, from Equation (7.56) we know

$$0 < \tilde{\mu} < \frac{2\mathbb{E}\left[\text{Re}\left\{\xi_u^H(n)\left(\mathbf{A}(n)\mathbf{A}^H(n)\right)^{-1}\mathbf{e}(n)\right\}\right]}{\mathbb{E}\left[\mathbf{e}^H(n)\left(\mathbf{A}(n)\mathbf{A}^H(n)\right)^{-1}\mathbf{e}(n)\right]}$$

where $\xi(n) = \mathbf{A}(n)\mathbf{w} - \mathbf{a}(n)\tilde{\mathbf{w}}(n)$ is the undisturbed error vector, and $\mathbf{e}(n) = \mathbf{A}(n)\tilde{\mathbf{w}}(n+1) - \mathbf{a}(n)\tilde{\mathbf{w}}(n)$ is the disturbed error vector. Here again, if these two error vectors are assumed to be equal, then the bounds of $\tilde{\mu}$ given in Equation (7.56) reduce to $0 < \tilde{\mu} < 2$.

Problem 7.9

a)

For the NLMS filter, suppose M is the length of the filter. Examine the weight-update formula

$$\begin{cases} \hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \tilde{\mu}\dfrac{\mathbf{u}(n)\mathbf{e}^*(n)}{\|\mathbf{u}(n)\|^2} \\ \\ e(n) = d(n) - \hat{\mathbf{w}}^H(n)\mathbf{u}(n) \end{cases}$$

The computation involving these calculations involve $5M$ multiplications (or divisions) and $2M$ additions. Hence, the computational complexity is $O(M)$

b)

For the APAF, suppose N is the order of the filter. Examine the weight-update formula

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \tilde{\mu}\mathbf{A}^H(n)\left(\mathbf{A}(n)\mathbf{A}^H(n)\right)^{-1}\mathbf{e}(n)$$

$$\mathbf{A}^H(n) = \begin{bmatrix}\mathbf{u}(n) & \mathbf{u}(n-1) & \dots & \mathbf{u}(n-N+1)\end{bmatrix}$$

$$\mathbf{e}(n) = \mathbf{d}(n) - \mathbf{A}(n)\hat{\mathbf{w}}(n)$$

$$\mathbf{d}(n) = \begin{bmatrix}d(n) & d(n-1) & \dots & d(n-N+1)\end{bmatrix}$$

Here we see that the computation involved in APAF is about N times as that of NLMS. Hence, the computational complexity of APAF is $O(MN)$

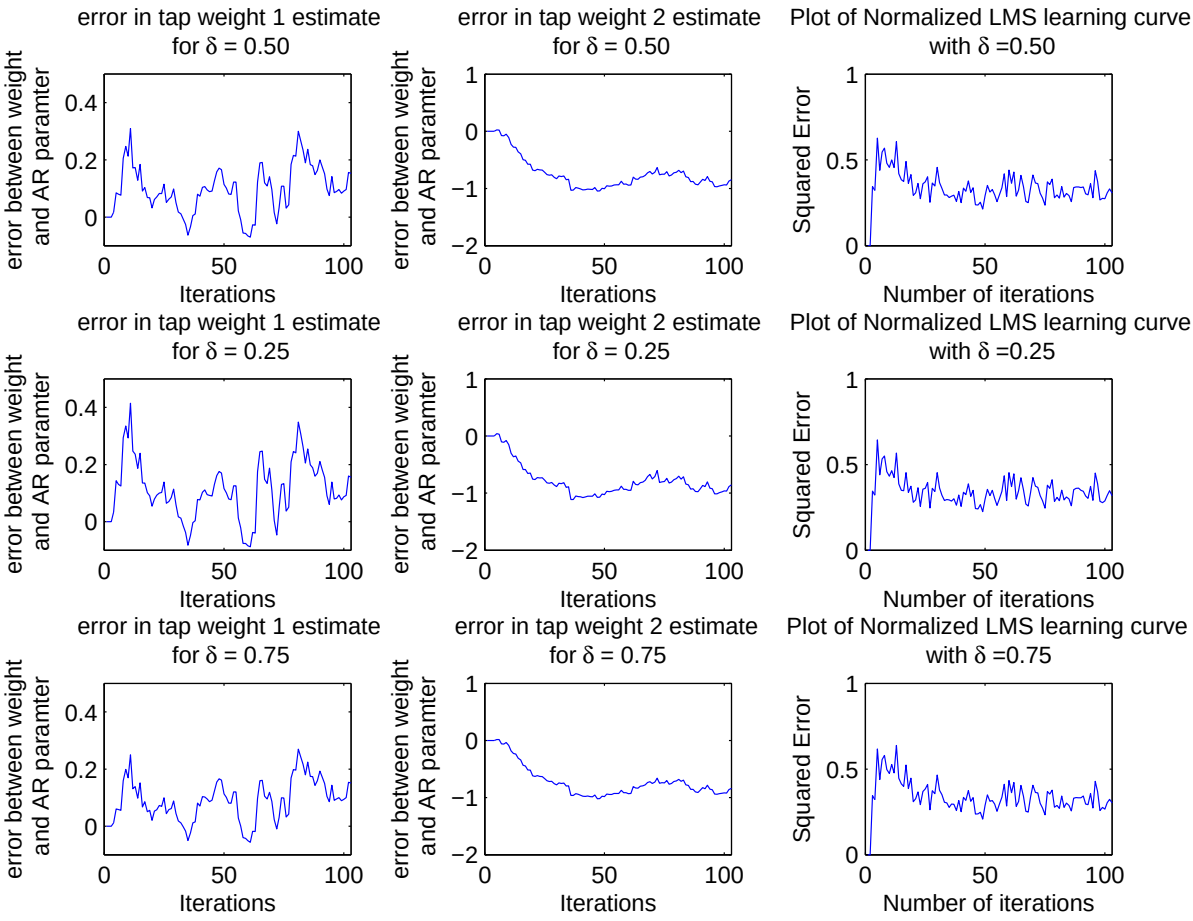
Problem 7.10

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Adaptive Filter Theory 5e Solution Manual
%
%
%
% Chapter 7
%
```

In the textbook the question statement refers to a system described in Chapter 6, question 18. This is a misprint, the question is actually suppose to refer back to the question 17 in Chapter 6.

```
% Question 10
%
% Parts a), b), and c)
%
%
%
% Program written to run on MATLAB 2010a (R)
%
%
%
% By Kelvin Hall
%
% July 2, 2014
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear
clc
numberOfDatapoints=100; % try 300 to get a better picture of the process
numberOfRuns=100;   % The number of monteCarlo runs necissary for part d)
aOne=0.1;          %AR paramter
aTwo=-0.8;         %AR paramter as per textbook description

NoiseVariance=0.28;% The Noise variance found through part A of the problem
NoiseStandardDeviation=sqrt(NoiseVariance);
mu=0.2;
delta=[0.5,0.25,0.75];

u=zeros(numberOfDatapoints+3,1);    % Allocate memory for input data stream
f=zeros(numberOfDatapoints+3,1);    % Allocate memory for error between
                                    % prediction and resutls
epsilonOne=zeros(numberOfDatapoints+3,1);% Allocate memory for error
                                   % between found weight and AR paramter 1
epsilonTwo=zeros(numberOfDatapoints+3,1);% Allocate memory for error
                                   % between found weight and AR paramter 2
J=zeros(numberOfDatapoints+3,1);% allocate memory for theoretical error

weights=zeros(2,numberOfDatapoints+3); % allocate memory for weights

for ell=1:3

stream = RandStream('mt19937ar','Seed',2);  % seed the random number
RandStream.setDefaultStream(stream);        % generator for reproducable
                                            % results
g=zeros(numberOfDatapoints+3,1);% allocate memory for squared-averaged error
for k=1:numberOfRuns % increment through the monte carlo runs of the problem
```

```
   W=zeros(2,numberOfDatapoints+3);     % reset the weights to zero between
                                        % each montecarlo run
   for n=3:numberOfDatapoints+3
        u(n)=aOne*u(n-1)+aTwo*u(n-2)+randn(1)*NoiseStandardDeviation;
             % create a new piece of input data to the filter

        f(n)=u(n)-W(1,n-1)*u(n-1)-W(2,n-1)*u(n-2);
             % Caluclate error between desired result and predicted results

        W(:,n)=W(:,n-1)+(mu/(delta(ell)+u(n-1)^2+u(n-2)^2)*([u(n-1);u(n-2)]*f(n)));
                  %update the weights as per LMS
        e(:,n)=W(:,n)-[u(n-1);u(n-2)];
    end
    g=g+f.^2;       % accumulate squared error of estimation
end
g=g/numberOfRuns;  % accumulate squared error of estimation
index=1:numberOfDatapoints+3; %the x axis of the plot

subplot(3,3,1+(ell-1)*3)
plot(index,W(1,:))
xlim([0 numberOfDatapoints+3])
ylim([-0.1, 0.5])
str=sprintf('error in tap weight 1 estimate for delta = %.2f',delta(ell));
title(str)
xlabel('Iterations')
ylabel('error between weight and AR paramter')

subplot(3,3,2+(ell-1)*3)
plot(index,W(2,:))
xlim([0 numberOfDatapoints+3])
str=sprintf('error in tap weight 2 estimate for delta = %.2f',delta(ell));
title(str)
xlabel('Iterations')
ylabel('error between weight and AR paramter')

subplot(3,3,3+(ell-1)*3)
plot(index,g(1:numberOfDatapoints+3))
xlim([0 numberOfDatapoints+3])
ylim([0 1])
xlabel('Number of iterations')
ylabel('Squared Error')
str=sprintf('Plot of Normalized LMS learning curve with delta =%.2f',delta(ell));
title(str);
end
```


error in tap weight 1 estimate
for δ = 0.50
error between weight and AR paramter
Iterations
error in tap weight 2 estimate
for δ = 0.50
error between weight and AR paramter
Iterations
Plot of Normalized LMS learning curve
with δ =0.50
Squared Error
Number of iterations
error in tap weight 1 estimate
for δ = 0.25
error between weight and AR paramter
Iterations
error in tap weight 2 estimate
for δ = 0.25
error between weight and AR paramter
Iterations
Plot of Normalized LMS learning curve
with δ =0.25
Squared Error
Number of iterations
error in tap weight 1 estimate
for δ = 0.75
error between weight and AR paramter
Iterations
error in tap weight 2 estimate
for δ = 0.75
error between weight and AR paramter
Iterations
Plot of Normalized LMS learning curve
with δ =0.75
Squared Error
Number of iterations

Chapter 8

Problem 8.1

When $M = 6$, $L = 4$, we have the k^{th} block

$$\begin{bmatrix} u(4k) & u(4k-1) & u(4k-2) & u(4k-3) & u(4k-4) & u(4k-5) \\ u(4k+1) & u(4k) & u(4k-1) & u(4k-2) & u(4k-3) & u(4k-4) \\ u(4k+2) & u(4k+1) & u(4k) & u(4k-1) & u(4k-2) & u(4k-3) \\ u(4k+3) & u(4k+2) & u(4k+1) & u(4k) & u(4k-1) & u(4k-2) \end{bmatrix} \begin{bmatrix} w_0(k) \\ w_1(k) \\ w_2(k) \\ w_3(k) \\ w_4(k) \\ w_5(k) \end{bmatrix} = \begin{bmatrix} y(4k) \\ y(4k+1) \\ y(4k+2) \\ y(4k+3) \end{bmatrix}$$

For the $(k+j)^{th}$ block, with $j = \pm 1, \pm 2, \ldots$, we have

$$\begin{bmatrix} u(4k+4j) & u(4k+4j-1) & u(4k+4j-2) & u(4k+4j-3) & u(4k+4j-4) & u(4k+4j-5) \\ u(4k+4j+1) & u(4k+4j) & u(4k+4j-1) & u(4k+4j-2) & u(4k+4j-3) & u(4k+4j-4) \\ u(4k+4j+2) & u(4k+4j+1) & u(4k+4j) & u(4k+4j-1) & u(4k+4j-2) & u(4k+4j-3) \\ u(4k+4j+3) & u(4k+4j+2) & u(4k+4j+1) & u(4k+4j) & u(4k+4j-1) & u(4k+4j-2) \end{bmatrix} \begin{bmatrix} w_0(k) \\ w_1(k) \\ w_2(k) \\ w_3(k) \\ w_4(k) \\ w_5(k) \end{bmatrix} = \begin{bmatrix} y(4k+4j) \\ y(4k+4j+1) \\ y(4k+4j+2) \\ y(4k+4j+3) \end{bmatrix}$$

Problem 8.2

a)

Using the $2M$-by-$2M$ constraint matrix $\mathbf{G}$, we may rewrite the cross-correlation vector $\phi(k)$ as follows:

$$\begin{bmatrix} \phi^T & 0 & \ldots & 0 \end{bmatrix}^T = \mathbf{G}_1 \mathbf{F}^{-1} \mathbf{U}^H(k) \mathbf{E}(k)$$

Accordingly, the weight-update is rewritten in the compact form:

$$\hat{\mathbf{W}}(k+1) = \hat{\mathbf{W}}(k) + \mu \mathbf{F}\mathbf{G}_1\mathbf{F}^{-1}\mathbf{U}^H(k)\mathbf{E}(k)$$

$$\hat{\mathbf{W}}(k+1) = \hat{\mathbf{W}}(k) + \mu \mathbf{G}\mathbf{U}^H(k)\mathbf{E}(k)$$

where

$$\mathbf{G} = \mathbf{F}\mathbf{G}_1\mathbf{F}^{-1}$$

The matrix operator $\mathbf{F}$ denotes discrete Fourier transformation, and $\mathbf{F}^{-1}$ denotes inverse discrete Fourier transformation.

b)

With

$$\mathbf{G}_2 = \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix}$$

the error vector $\mathbf{E}(k)$ is readily rewriten as follows:

$$\mathbf{E}(k) = \mathbf{F} \begin{bmatrix} 0 & \dots & 0 & \mathbf{e}^T(k) \end{bmatrix}^T$$

$$\mathbf{E}(k) = \mathbf{F}\mathbf{G}_2^T\mathbf{e}^T(k)$$

c)

Using these matrix notations, the fast LMS algorithm may be reformulated as follows:

Initialization

$$\hat{\mathbf{W}}(0) = \begin{bmatrix} 0 & 0 & \dots & 0 \end{bmatrix}^T$$

$$P_i(0) = \delta_i, \quad i = 0, 1, \dots, 2M-1$$

where M is the number of taps, and δ_i is a small positive number.

Computation: For each new block of M input samples, compute the following:

$$\mathbf{U}(k) = \text{diag}\left(\mathbf{F}\begin{bmatrix} u(kM-M) & \dots & u(kM-1) & u(kM) & \dots & u(kM+M-1) \end{bmatrix}^T\right)$$

$$\mathbf{Y}(k) = \mathbf{U}(k)\mathbf{W}(k)$$

$$\mathbf{y}(k) = \mathbf{G}_2\mathbf{F}^{-1}\mathbf{Y}(k)$$

$$\mathbf{e}(k) = \mathbf{d}(k) - \mathbf{y}(k)$$
$$\mathbf{E}(k) = \mathbf{F}\mathbf{G}_2^T\mathbf{e}(k)$$
$$P_i(k) = \gamma P_i(k-1) + (1-\gamma)\left|U_i(k)\right|^2, \quad i = 0, 1, \ldots, 2M-1$$
$$\mu(k) = \text{diag}\left(P_0^{-1}(k), P_1^{-1}(k), \ldots, P_{2M-1}^{-1}(k)\right)$$
$$\hat{\mathbf{W}}(k+1) = \hat{\mathbf{W}}(k) + \mathbf{F}\mathbf{G}_1\mathbf{F}^{-1}\mu(k)\mathbf{U}(k)\mathbf{E}(k)$$

d)
When the constraint gradient $\mathbf{G}$ is equal to the identity matrix, the fast LMS algorithm reduces to the unconstrained frequency-domain form.

Problem 8.3

Removing the gradient constraint shown in Figure 8.3 permits the adaptive filter to perform circular convolution instead of linear convolution. (Circular convolution is the form of convolution performed by the discrete Fourier transform.) This modification may be acceptable in those applications where there is no particular concern as to how the input signal is used to minimize the mean-square value of the error signal (estimation error). One such application is that of noise (interference) cancellation.

Problem 8.4

a)
From Figure P8.1, we note that the z-transform of $x_k(n)$ is related to the z-transform of the input $u(n)$ as follows:

$$z\left[x_k(n)\right] = \frac{1 - z^{-M}}{1 - \exp\left(-\dfrac{\mathrm{j}\,2\pi}{M}\right) z^{-1}} z\left[u(n)\right], \qquad k = 0, 1, \ldots, M-1$$

Cross-multiplying and rearranging terms:

$$z\left[x_k(n)\right] = \exp\left(-\frac{\mathrm{j}\,2\pi}{M}\right) z^{-1} \times z\left[x_k(n)\right] + z\left[u(n)\right] - z^{-M} z\left[u(n)\right]$$

Taking the inverse z-transforms:

$$x_k(n) = \exp\left(-\frac{\mathrm{j}\,2\pi}{M}k\right) x_k(n-1) + u(n) - u(n-M), \qquad k = 0, 1, \ldots, M-1$$

b)

We are given that

$$\mathbf{h}(n+1) = \mathbf{h}(n) + \mu \mathbf{D}^{-1}\mathbf{x}(n)e^*(n) \tag{1}$$

The error signal is

$$\begin{aligned} e(n) &= d(n) - \mathbf{h}^H(n)\mathbf{x}(n) \\ &= d(n) - \mathbf{x}^T(n)\mathbf{h}^*(n) \end{aligned} \tag{2}$$

Hence, substituting Equation (2) into Equation (1) and rearranging:

$$\mathbf{h}(n+1) = \left[\mathbf{I} - \mu \mathbf{D}^{-1}\mathbf{x}(n)\mathbf{x}^H(n)\right]\mathbf{h}(n) + \mu \mathbf{D}^{-1}\mathbf{x}(n)d^*(n)$$

Taking the expecation of both sides of this equation, and invoking the properties of LMS filters with a small step size:

$$\mathbb{E}\left[\mathbf{h}(n+1)\right] = \left[\mathbf{I} - \mu \mathbf{D}^{-1}\mathbf{R}_x\right]\mathbb{E}\left[\mathbf{h}(n)\right] + \mu \mathbf{D}^{-1}\mathbf{p}_x \tag{3}$$

where $\mathbf{R}_x$ is the correlation matrix of the input vector $\mathbf{x}(n)$, and $\mathbf{p}_x$ is the cross-correlation vector between $\mathbf{x}(n)$ and $d(n)$. Assuming convergence in the mean,

$$\mathbb{E}\left[\mathbf{h}(n+1)\right] = \mathbb{E}\left[\mathbf{h}(n)\right] = \mathbf{h}_0 \quad \text{as } n \to \infty$$

Hence, from Equation (3)we deduce that

$$\mathbf{R}_x\mathbf{h}_0 = \mathbf{p}_x \tag{4}$$

But $x_k(n)$ is related to the input vector $\mathbf{u}(n)$ by the DFT formula:

$$\begin{aligned} x_k(n) &= \sum_{i=0}^{M-1} u(n-i)\exp\left(-\frac{\mathrm{j}\,2\pi}{M}ik\right) \\ &= \mathbf{u}^T(n)\varepsilon_k^* = \varepsilon_k^H\mathbf{u}(n) \end{aligned}$$

where

$$\varepsilon_k = \left[1 \quad \exp\left(\frac{\mathrm{j}\,2\pi}{M}k\right) \quad \dots \quad \exp\left(\frac{\mathrm{j}\,2\pi}{M}(M-1)\right)\right]^T$$

Hence, the vector $\mathbf{x}(n)$ is related to $\mathbf{u}(n)$ by

$$\mathbf{x}(n) = \begin{bmatrix} \varepsilon_0^H \\ \varepsilon_1^H \\ \vdots \\ \varepsilon_{M-1}^H \end{bmatrix} \mathbf{u}(n)$$

$$\mathbf{x}(n) = \sqrt{M}\mathbf{Q}\mathbf{u}(n)$$

where $\mathbf{Q}$ is the M-by$M$ unitary matrix

$$\mathbf{Q} = \frac{1}{\sqrt{M}} \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & \exp\left(-\frac{\mathrm{j}\,2\pi}{M}\right) & \dots & \exp\left(-\frac{\mathrm{j}\,2\pi}{M}(M-1)\right) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \exp\left(-\frac{\mathrm{j}\,2\pi}{M}(M-1)\right) & \dots & \exp\left(-\frac{\mathrm{j}\,2\pi}{M}(M-1)^2\right) \end{bmatrix} \tag{5}$$

Accordingly, we have

$$\begin{aligned} \mathbf{R}_x &= \mathbb{E}\left[\mathbf{x}(n)\mathbf{x}^H(n)\right] \\ &= \mathbb{E}\left[\mathbf{Q}\mathbf{u}(n)\mathbf{u}^H(n)\mathbf{Q}^H(n)\right] \\ &= \mathbf{Q}\mathbf{R}\mathbf{Q}^H \end{aligned} \tag{6}$$

where $\mathbf{R}$ is the correlation matrix of the input vector $\mathbf{u}(n)$. Similarly, we may write

$$\begin{aligned} \mathbf{p}_x &= \mathbb{E}\left[\mathbf{x}(n)d^*(n)\right] \\ &= \mathbb{E}\left[\mathbf{Q}\mathbf{u}(n)d^*(n)\right] \\ &= \mathbf{Q}\mathbf{p} \end{aligned} \tag{7}$$

Where $\mathbf{p}$ is the cross-correlation vector between $\mathbf{u}(n)$ and $d(n)$. Next, substituting Equation (6) and Equation (7) into Equation (4):

$$\mathbf{Q}\mathbf{R}\mathbf{Q}^H\mathbf{h}_0 = \mathbf{Q}\mathbf{p}$$

The matrix $\mathbf{Q}$ is nonsingular, and its inverse matrix therefore exists. Hence,

$$\mathbf{R}\mathbf{Q}^H\mathbf{h}_0 = \mathbf{p}$$

We next recognize the Wiener solution:

$$\mathbf{w}_0 = \mathbf{R}^{-1}\mathbf{p}$$

It follows therefore that

$$\mathbf{Q}^H\mathbf{h}_0 = \mathbf{w}_0$$

The matrix $\mathbf{Q}$ is in actual fact a unitary matrix; we may thus also write

$$\mathbf{h}_0 = \mathbf{Q}\mathbf{w}_0$$

The unitary matrix $\mathbf{Q}$ is related to the DTF by Equation (5)

c.i)

From part **b)**, we have

$$\begin{aligned}\mathbf{h}_0 &= \mathbf{Q}\mathbf{w}_0 \\ &= \mathbf{Q}\mathbf{R}^{-1}\mathbf{p} \\ &= \mathbf{Q}\mathbf{R}^{-1}\mathbf{Q}^H\mathbf{Q}\mathbf{p}\end{aligned}$$

We recognize $\mathbf{Q}\mathbf{R}^{-1}\mathbf{Q}^H$ as the correlation matrix of the tap-input vector $\mathbf{x}(n)$, and $\mathbf{Q}\mathbf{p}$ as the cross correlation vector between $\mathbf{x}(n)$ and the desired response $d(n)$. From the unitary similarity transformation, $\mathbf{Q}\mathbf{R}^{-1}\mathbf{Q}^H$ equals a diagonal matrix. This is functionally equivalent to prewhitening the original input vector $\mathbf{u}(n)$ characterized by the correlation matrix $\mathbf{R}$.

c.ii)

The eigenvalue spread $\chi(\mathbf{R})$ of the original input vector $\mathbf{u}(n)$ is, in general, greater than unity. On the other hand, the eigenvalue spread of $\chi(\mathbf{R})$ of the DFT output vector $\mathbf{x}(n)$, which is prewhitened by the DFT, is closer to unity. In general, we therefore find the eigenvalue spread of $\mathbf{u}(n)$ is larger than that of $\mathbf{x}(n)$.

c.iii)

The frequency-domain LMS algorithm has a faster rate of convergence than the conventional LMS algorithm, because of the results reported in sub-part **ii** of **c**.

Problem 8.5

a)

We start with

$$\begin{aligned} C_m(n) &= \frac{1}{2}k_m(-1)^m W_{2M}^{m/2} A_m(n) \\ &= \frac{1}{2}k_m(-1)^m W_{2M}^{m/2}\left(A_m^{(1)}(n) + A_m^{(2)}(n)\right) \\ &= \frac{1}{2}k_m\left(C_m^{(1)}(n) + C_m^{(2)}(n)\right), \qquad m = 0, 1, \ldots, M-1 \end{aligned}$$

Where

$$\begin{aligned} C_m^{(1)}(n) &= (-1)^m W_{2M}^{m/2} A_m^{(1)}(n) \\ &= (-1)^m W_{2M}^{m/2}\left[W_{2M}^{m} A_m^{(1)}(n-1) + u(n) - (-1)^m u(n-M)\right] \\ &= W_{2M}^{m/2}\left[W_{2M}^{m/2} C_m^{(1)}(n-1) + (-1)^m u(n) - u(n-M)\right] \end{aligned}$$

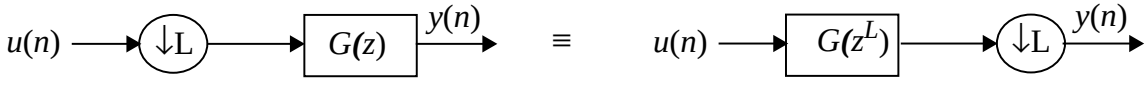
$$\begin{aligned} C_m^{(2)}(n) &= (-1)^m W_{2M}^{m/2} A_m^{(2)}(n) \\ &= (-1)^m W_{2M}^{m/2}\left[W_{2M}^{-m} A_m^{(2)}(n-1) + W_{2M}^{-m}\left(u(n) - (-1)^m u(n-M)\right)\right] \\ &= W_{2M}^{-m/2}\left[W_{2M}^{-m/2} C_m^{(2)}(n-1) + (-1)^m u(n) - u(n-M)\right] \end{aligned}$$

b)

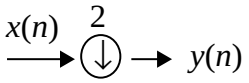
Basically, the term $W_{2m}^{m/2}$, $C_m^{(1)}(n-1)$ and $u(n-m)$ involved in the computation of $C_m^{(1)}(n)$, and likewise the terms $W_{2m}^{-m/2}$, $C_m^{(2)}(n-1)$ and $u(n-m)$ involved in the computation of $C_m^{(2)}(n)$ are all multiplied by the constant β, to ensure stability of the algorithm.

Problem 8.6

Using nobel identity 1



Putting $L = 2$ and noting that in the z-domain, the 2-band down sampling operator



can be expressed by

$$Y(E) = \frac{1}{2}\left[X\left(z^{1/2}\right) + X\left(-z^{1/2}\right)\right] \tag{1}$$

Observing the bottom part of Figure 8.13(b) in the text, we may formulate the synthesis filter output as

$$\text{upper} \quad \frac{1}{2}\left[U\left(z^{1/2}\right)\hat{W}\left(z^{1/2}\right)H_1\left(z^{1/2}\right) + U\left(-z^{1/2}\right)\hat{W}\left(-z^{1/2}\right)H_1\left(-z^{1/2}\right)\right] \tag{2}$$

$$\text{lower} \quad \frac{1}{2}\left[U\left(z^{1/2}\right)\hat{W}\left(z^{1/2}\right)H_0\left(z^{1/2}\right) + U\left(-z^{1/2}\right)\hat{W}\left(-z^{1/2}\right)H_1\left(-z^{1/2}\right)\right] \tag{3}$$

By polyphase decomposition of $\hat{W}(z)$ [see Equation (8.79) of the text]

$$\hat{W}(z) = \hat{W}_1(z^2) + z^{-1}\hat{W}_2(z^2)$$

if follows that

$$\hat{W}(z^{1/2}) = \hat{W}_1(z) + z^{-1/2}\hat{W}_2(z) \tag{4}$$

Substituting Equation (4) into Equation (2) and rearranging the terms, we have

$$\begin{aligned}&\frac{1}{2}\left[U\left(z^{1/2}\right)H_1\left(z^{1/2}\right) + U\left(-z^{1/2}\right)H_1\left(-z^{1/2}\right)\right]\hat{W}_1(z)\\ &\qquad + \frac{1}{2}\left[U\left(z^{1/2}\right)H_1\left(z^{1/2}\right)z^{1/2} + U\left(-z^{1/2}\right)H_1\left(-z^{1/2}\right)z^{1/2}\right]\hat{W}_2(z)\end{aligned} \tag{5}$$

Similarly Equation(4) into Equation (3), we have

$$\begin{aligned}&\frac{1}{2}\left[U\left(z^{1/2}\right)H_0\left(z^{1/2}\right) + U\left(-z^{1/2}\right)H_0\left(-z^{1/2}\right)\right]\hat{W}_1(z)\\ &\qquad + \frac{1}{2}\left[U\left(z^{1/2}\right)H_0\left(z^{1/2}\right)z^{-1/2} + U\left(-z^{1/2}\right)H_0\left(-z^{1/2}\right)z^{1/2}\right]\hat{W}_2(z)\end{aligned} \tag{6}$$

Equation (5) corresponds to the description of the upper-bottom part of Figure (8.15). Equation (6) corresponds to the description of the lower-bottom part of Figure (8.15). The equivalence between the two Figures 8.13 and 8.15 is therefore established.

Problem 8.7

Consider the case when the input $u(n)$ is complex-valued. Suppose

$$u(n) = u_I(n) + \mathrm{j}\, u_Q(n)$$

$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}_I(n) + \mathrm{j}\, \hat{\mathbf{w}}_Q(n)$$

$$\begin{aligned}\mathbf{e}(n) &= \left[e_{1,I}(n) + \mathrm{j}\, e_{1,Q}(n), \quad e_{2,I}(n) + \mathrm{j}\, e_{2,Q}(n)\right] \\ &= \left(\left[e_{1,I}(n), e_{2,I}(n)\right] + \mathrm{j}\, \left[e_{1,Q}(n), e_{2,Q}(n)\right]\right)\end{aligned}$$

$$\mathbf{P} = \begin{bmatrix}\alpha_1 & 0 \\ 0 & \alpha_2\end{bmatrix}$$

Define the cost function

$$\begin{aligned}J(n) &= \frac{1}{2}\mathbf{e}^H(n)\mathbf{P}\mathbf{e}^*(n) \\ &= \frac{1}{2}\left[\alpha_1\left(e_{1,I}^2 + e_{1,Q}^2\right) + \alpha_2\left(e_{2,I}^2 + e_{2,Q}^2\right)\right] \qquad (1)\end{aligned}$$

Suppose

$$\hat{W}_{1,k} = \hat{W}_{1,k}^{(I)} + \mathrm{j}\, \hat{W}_{1,k}^{(Q)}$$

$$\hat{W}_{2,k} = \hat{W}_{2,k}^{(I)} + \mathrm{j}\, \hat{W}_{2,k}^{(Q)}$$

Similar to the real-valued case, we may derive

$$\hat{W}_{1,k}^{(I)}(n+1) = \hat{W}_{1,k}^{(I)}(n+1) + \mu\frac{\partial J(n)}{\partial \hat{W}_{1,k}^{(I)}}$$

$$\hat{W}_{2,k}^{(I)}(n+1) = \hat{W}_{2,k}^{(I)}(n+1) + \mu\frac{\partial J(n)}{\partial \hat{W}_{2,k}^{(I)}}$$

$$\hat{W}_{1,k}^{(Q)}(n+1) = \hat{W}_{1,k}^{(Q)}(n+1) + \mu\frac{\partial J(n)}{\partial \hat{W}_{1,k}^{(Q)}}$$

$$\hat{W}_{2,k}^{(Q)}(n+1) = \hat{W}_{2,k}^{(Q)}(n+1) + \mu\frac{\partial J(n)}{\partial \hat{W}_{2,k}^{(Q)}}$$

From Equation (1) we also have

$$\frac{\partial J(n)}{\partial \hat{W}_{1,k}^{(I)}} = \alpha_1 \left[e_{1,I}(n) \frac{\partial e_{1,I}(n)}{\partial \hat{W}_{1,k}^{(I)}} + e_{1,Q}(n) \frac{\partial e_{1,Q}(n)}{\partial \hat{W}_{1,k}^{(I)}} \right] + \alpha_2 \left[e_{2,I}(n) \frac{\partial e_{2,I}(n)}{\partial \hat{W}_{1,k}^{(I)}} + e_{1,Q}(n) \frac{\partial e_{1,Q}(n)}{\partial \hat{W}_{1,k}^{(I)}} \right] \quad (2)$$

$$\frac{\partial J(n)}{\partial \hat{W}_{2,k}^{(I)}} = \alpha_1 \left[e_{1,I}(n) \frac{\partial e_{1,I}(n)}{\partial \hat{W}_{2,k}^{(I)}} + e_{1,Q}(n) \frac{\partial e_{1,Q}(n)}{\partial \hat{W}_{2,k}^{(I)}} \right] + \alpha_2 \left[e_{2,I}(n) \frac{\partial e_{2,I}(n)}{\partial \hat{W}_{2,k}^{(I)}} + e_{1,Q}(n) \frac{\partial e_{1,Q}(n)}{\partial \hat{W}_{2,k}^{(I)}} \right] \quad (3)$$

$$\frac{\partial J(n)}{\partial \hat{W}_{1,k}^{(Q)}} = \alpha_1 \left[e_{1,I}(n) \frac{\partial e_{1,I}(n)}{\partial \hat{W}_{1,k}^{(Q)}} + e_{1,Q}(n) \frac{\partial e_{1,Q}(n)}{\partial \hat{W}_{1,k}^{(Q)}} \right] + \alpha_2 \left[e_{2,I}(n) \frac{\partial e_{2,I}(n)}{\partial \hat{W}_{1,k}^{(Q)}} + e_{1,Q}(n) \frac{\partial e_{1,Q}(n)}{\partial \hat{W}_{1,k}^{(Q)}} \right] \quad (4)$$

$$\frac{\partial J(n)}{\partial \hat{W}_{2,k}^{(Q)}} = \alpha_1 \left[e_{1,I}(n) \frac{\partial e_{1,I}(n)}{\partial \hat{W}_{2,k}^{(Q)}} + e_{1,Q}(n) \frac{\partial e_{1,Q}(n)}{\partial \hat{W}_{2,k}^{(Q)}} \right] + \alpha_2 \left[e_{2,I}(n) \frac{\partial e_{2,I}(n)}{\partial \hat{W}_{2,k}^{(Q)}} + e_{1,Q}(n) \frac{\partial e_{1,Q}(n)}{\partial \hat{W}_{2,k}^{(Q)}} \right] \quad (5)$$

Similarly, we obtain for $i = 1, 2$ and $j = 1, 2$

$$\frac{\partial e_{i,I}(n)}{\partial \hat{W}_{j,k}^{(I)}} = -X_{ij}^{(I)}(n-k), \qquad \frac{\partial e_{i,Q}(n)}{\partial \hat{W}_{j,k}^{(I)}} = -X_{ij}^{(Q)}(n-k) \quad (6)$$

$$\frac{\partial e_{i,Q}(n)}{\partial \hat{W}_{j,k}^{(Q)}} = -X_{ij}^{(Q)}(n-k), \qquad \frac{\partial e_{i,I}(n)}{\partial \hat{W}_{j,k}^{(Q)}} = -X_{ij}^{(I)}(n-k) \quad (7)$$

Substituting Equation (6) and Equation (7) into Equations (2)-(5), we finally get

$$\hat{W}_{1,k}^{(I)}(n+1) = \hat{W}_{1,k}^{(I)}(n) + \mu \Big[\alpha_1 e_{1,I}(n) X_{11}^{(I)}(n-k) - \alpha_1 e_{1,Q}(n) X_{11}^{(Q)}(n-k) + \alpha_2 e_{2,I}(n) X_{21}^{(I)}(n-k) - \alpha_2 e_{2,Q}(n) X_{21}^{(Q)}(n-k)\Big]$$

$$\hat{W}_{1,k}^{(Q)}(n+1) = \hat{W}_{1,k}^{(Q)}(n) + \mu \Big[\alpha_1 e_{1,I}(n) X_{11}^{(Q)}(n-k) - \alpha_1 e_{1,Q}(n) X_{11}^{(I)}(n-k) + \alpha_2 e_{2,I}(n) X_{21}^{(Q)}(n-k) - \alpha_2 e_{2,Q}(n) X_{21}^{(I)}(n-k)\Big]$$

$$\hat{W}_{2,k}^{(I)}(n+1) = \hat{W}_{2,k}^{(1)}(n) + \mu \Big[\alpha_1 e_{1,I}(n) X_{12}^{(I)}(n-k) - \alpha_1 e_{1,Q}(n) X_{12}^{(Q)}(n-k) + \alpha_2 e_{2,I}(n) X_{22}^{(I)}(n-k) - \alpha_2 e_{2,Q}(n) X_{22}^{(Q)}(n-k)\Big]$$

$$\hat{W}_{2,k}^{(Q)}(n+1) = \hat{W}_{2,k}^{(Q)}(n) + \mu \Big[\alpha_1 e_{1,I}(n) X_{12}^{(Q)}(n-k) - \alpha_1 e_{1,Q}(n) X_{12}^{(I)}(n-k) + \alpha_2 e_{2,I}(n) X_{22}^{(Q)}(n-k) - \alpha_2 e_{2,Q}(n) X_{22}^{(I)}(n-k)\Big]$$

Problem 8.8

Using NLMS algorithm, suppose

$$\mathbf{x}_{1,1} = \left[x_{11}(n) \quad x_{11}(n-1) \quad \dots \quad x_{11}\left(n - \frac{M}{2} + 1\right)\right]^T$$

$$\mathbf{x}_{2,1} = \left[x_{21}(n) \quad x_{21}(n-1) \quad \dots \quad x_{21}\left(n - \frac{M}{2} + 1\right)\right]^T$$

Applying the NLMS version of Equation (8.91) of the text, we may write

$$\hat{W}_{1,k}(n+1) = \hat{W}_{1,k}(n) + \mu \left[\frac{\alpha_1 e_1(n) x_{11}(n-k)}{\|\mathbf{x}_{1,1}\|^2} + \frac{\alpha_2 e_2(n) x_{21}(n-k)}{\|\mathbf{x}_{2,1}\|^2}\right]$$

Similarly, the NLMS version of Equation (8.92) is

$$\hat{W}_{2,k}(n+1) = \hat{W}_{2,k}(n) + \mu \left[\frac{\alpha_1 e_1(n) x_{12}(n-k)}{\|\mathbf{x}_{1,2}\|^2} + \frac{\alpha_2 e_2(n) x_{22}(n-k)}{\|\mathbf{x}_{2,2}\|^2}\right]$$

Where

$$\mathbf{x}_{1,2} = \left[x_{12}(n) \quad x_{12}(n-1) \quad \dots \quad x_{12}\left(n - \frac{M}{2} + 1\right) \right]^T$$

$$\mathbf{x}_{2,2} = \left[x_{22}(n) \quad x_{22}(n-1) \quad \dots \quad x_{22}\left(n - \frac{M}{2} + 1\right) \right]^T$$

Chapter 9

Problem 9.1

The Hermitian transpose of data matrix $\mathbf{A}$ is defined by

$$\mathbf{A}^H = \begin{bmatrix} u(1,1) & u(1,2) & \dots & u(1,n) \\ u(2,1) & u(2,2) & \dots & u(2,n) \\ \vdots & \vdots & & \vdots \\ u(M,1) & u(M,2) & \dots & u(M,n) \end{bmatrix}$$

where $u(k,i)$ is the output of sensor k in the linear array at time i, where $k = 1, 2, \dots, M$, and $i = 1, 2, \dots, n$.

a)

The matrix product $\mathbf{A}^H\mathbf{A}$ is given by:

$$\mathbf{A}^H\mathbf{A} = \begin{bmatrix} \sum_{i=1}^{n} u(1,i)u^*(1,i) & \sum_{i=1}^{n} u(1,i)u^*(2,i) & \dots & \sum_{i=1}^{n} u(1,i)u^*(M,i) \\ \sum_{i=1}^{n} u(2,i)u^*(1,i) & \sum_{i=1}^{n} u(2,i)u^*(2,i) & \dots & \sum_{i=1}^{n} u(2,i)u^*(M,i) \\ \vdots & \vdots & & \vdots \\ \sum_{i=1}^{n} u(M,i)u^*(1,i) & \sum_{i=1}^{n} u(M,i)u^*(2,i) & \dots & \sum_{i=1}^{n} u(M,i)u^*(M,i) \end{bmatrix}$$

This matrix product represents the M-by-M spatial (deterministic) correlation matrix of the array with temporal averaging applied to each element of the matrix. This form of averaging assumes that the environment in which the array operates is temporally stationary.

b)
The alternative matrix product $\mathbf{A}\mathbf{A}^H$ is defined by:

$$\mathbf{A}\mathbf{A}^H = \begin{bmatrix} \sum_{k=1}^{n} u^*(k,1)u(k,1) & \sum_{k=1}^{n} u^*(k,1)u(k,2) & \dots & \sum_{k=1}^{n} u^*(k,1)u(k,n) \\ \sum_{k=1}^{n} u^*(k,2)u(k,1) & \sum_{k=1}^{n} u^*(k,2)u(k,2) & \dots & \sum_{k=1}^{n} u^*(k,2)u(k,n) \\ \vdots & \vdots & & \vdots \\ \sum_{k=1}^{n} u^*(k,n)u(k,1) & \sum_{k=1}^{n} u^*(k,n)u(k,2) & \dots & \sum_{k=1}^{n} u^*(k,n)u(k,n) \end{bmatrix}$$

This second matrix represent the n-by-n temporal (deterministic) correlation matrix of the array with spatial averaging applied to each element of the matrix. This form of averaging assumes that the environment is spatially stationary.

Problem 9.2

We say that the least-squares estimate of the 1-by-N $\hat{\mathbf{w}}$ is *consistent* if

$$\lim_{N\to\infty} \mathbb{E}\left[\|\hat{\mathbf{w}} - \mathbf{w}_0\|^2\right] = 0 \tag{1}$$

We note that

$$\begin{aligned} \mathbb{E}\left[\|\hat{\mathbf{w}} - \mathbf{w}_0\|^2\right] =& \mathbb{E}\left[(\hat{\mathbf{w}} - \mathbf{w}_0)^H (\hat{\mathbf{w}} - \mathbf{w}_0)\right] \\ =& \text{tr}\left\{\mathbb{E}\left[(\hat{\mathbf{w}} - \mathbf{w}_0)^H (\hat{\mathbf{w}} - \mathbf{w}_0)\right]\right\} \\ =& \mathbb{E}\left[\text{tr}\left\{(\hat{\mathbf{w}} - \mathbf{w}_0)^H (\hat{\mathbf{w}} - \mathbf{w}_0)\right\}\right] \\ =& \mathbb{E}\left[\text{tr}\left\{(\hat{\mathbf{w}} - \mathbf{w}_0) (\hat{\mathbf{w}} - \mathbf{w}_0)^H\right\}\right] \\ =& \text{tr}\left\{\mathbb{E}\left[(\hat{\mathbf{w}} - \mathbf{w}_0) (\hat{\mathbf{w}} - \mathbf{w}_0)^H\right]\right\} \\ =& \text{tr}\left[\mathbf{K}\right] \\ =& \sigma^2 \text{tr}\left[\phi^{-1}\right] \end{aligned} \tag{2}$$

where the N-by-N correlation matrix ϕ is dependent on N. Substituting Equation (2) in (1):

$$\lim_{N\to\infty} \mathbb{E}\left[\|\hat{\mathbf{w}} - \mathbf{w}_0\|^2\right] = \lim_{N\to\infty} \sigma^2 \text{tr}\left[\phi^{-1}\right]$$

which shows that $\hat{\mathbf{w}}$ is consistent if

$$\lim_{N\to\infty} \text{tr}\left[\phi^{-1}\right] = 0$$

Problem 9.3

The data matrix is

$$\mathbf{A} = \begin{bmatrix} 2 & 3 \\ 1 & 2 \\ -1 & 1 \end{bmatrix}$$

The desired response vector is

$$\mathbf{d} = \begin{bmatrix} 2 \\ 1 \\ 1/34 \end{bmatrix}$$

The tap-weights vector of the linear least-squares filter is

$$\hat{\mathbf{w}} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{d} \tag{1}$$

We first note that

$$\begin{aligned}\mathbf{A}^T\mathbf{A} &= \begin{bmatrix} 2 & 1 & -1 \\ 3 & 2 & 1 \end{bmatrix}\begin{bmatrix} 2 & 3 \\ 1 & 2 \\ -1 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 6 & 7 \\ 7 & 14 \end{bmatrix}\end{aligned}$$

$$\det\left(\mathbf{A}^T\mathbf{A}\right) = 35$$

$$\left(\mathbf{A}^T\mathbf{A}\right)^{-1} = \frac{1}{35}\begin{bmatrix} 14 & -7 \\ -7 & 6 \end{bmatrix} = \begin{bmatrix} \frac{2}{5} & -\frac{1}{5} \\ -\frac{1}{5} & \frac{6}{35} \end{bmatrix}$$

Hence, using Equation (1) we get

$$\begin{aligned}\hat{\mathbf{w}} &= \begin{bmatrix} \frac{2}{5} & -\frac{1}{5} \\ -\frac{1}{5} & \frac{6}{35} \end{bmatrix} \begin{bmatrix} 2 & 1 & -1 \\ 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 1/34 \end{bmatrix} \\ &= \begin{bmatrix} \frac{2}{5} & -\frac{1}{5} \\ -\frac{1}{5} & \frac{6}{35} \end{bmatrix} \begin{bmatrix} \frac{169}{34} \\ \frac{273}{34} \end{bmatrix} \\ &= \begin{bmatrix} 0.382 \\ 0.382 \end{bmatrix}\end{aligned}$$

Problem 9.4

Express the transfer function of the (forward) prediction-error filter as follows (see Fig. 1):

$$H_f(z) = \left(1 - z_i z^{-1}\right) H_f'(z)$$

where

$$H_f(z) = \sum_{k=0}^{M} a_{M,k}^{*} z^{-1}, \quad a_{M,0} = 1$$

and

$$z_i = \rho_i \exp\left(\mathrm{j}\,\theta_i\right)$$

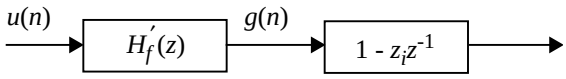


Figure 1

From this figure we note that

$$Z\left[f_M(n)\right] = \left(1 - z_i z^{-1}\right) Z\left[g(n)\right]$$

Hence

$$f_M(n) = g(n) - z_i g(n-1)$$

The prediction-error energy equals (according to the autocorrelation method)

$$\begin{aligned}\epsilon_f &= \sum_{n=1}^{\infty} |f_M(n)|^2 \\ &= \sum_{n=1}^{\infty} [g(n) - z_i g(n-1)] [g^*(n) - z_i^* g^*(n-1)]\end{aligned}$$

With $z_i = \rho_i \exp(\mathrm{j}\,\theta_i)$, we may expand the expression for the prediction-error energy as

$$\epsilon_f = \sum_{n=1}^{\infty} \left\{ |g(n)|^2 + \rho_i^2 |g(n-1)|^2 - 2\mathrm{Re}\left[\rho_i \exp(\mathrm{j}\,\theta_i)\, g(n-1) g^*(n)\right] \right\}$$

Differentiating ϵ_f with respect to ρ_i:

$$\frac{\partial \epsilon_f}{\partial \rho_i} = 2\rho_i \sum_{n=1}^{\infty} \left\{ |g(n-1)|^2 - 2\mathrm{Re}\left[\exp(\mathrm{j}\,\theta_i)\, g(n-1) g^*(n)\right] \right\} \tag{1}$$

From the Cauchy-Schwartz inequality, we have

$$\mathrm{Re}\left[\sum_{n=1}^{\infty} \exp(\mathrm{j}\,\theta_i)\, g(n-1) g^*(n)\right] \leq \left[\sum_{n=1}^{\infty} |g^*(n)|^2\right]^{1/2} \left[\sum_{n=1}^{\infty} |\exp(\mathrm{j}\,\theta_i)\, g(n-1)|^2\right]^{1/2}$$

Hence, we may make the statement:

$$\begin{aligned}\sum_{n=1}^{\infty} |\exp(\mathrm{j}\,\theta_i)\, g(n-1)|^2 &= \sum_{n=1}^{\infty} |g(n-1)|^2 \\ &= \sum_{n=0}^{\infty} |g(n)|^2 \end{aligned} \tag{2}$$

where it is recognized that in the autocorrelation method $g(n)$ is zero for $n \leq 0$. Accordingly, we may rewrite the Cauchy-Schwartz inequality as

$$\mathrm{Re}\left[\sum_{n=1}^{\infty} \exp(\mathrm{j}\,\theta_i)\, g(n-1) g^*(n)\right] \leq \sum_{n=0}^{\infty} |g(n)|^2 \tag{3}$$

Using Equations. (1)-(3), we thus have

$$\frac{\partial \epsilon_f}{\partial \rho_i} \geq 2(\rho_i - 1) \sum_{n=0}^{\infty} |g(n)|^2 \tag{4}$$

For $\rho_i \geq 1$, the right-hand side of Equation (4) is always greater than or equal to zero. The derivative of the prediction error energy with respect to ρ_i is zero if, and only if, $\rho_i = 1$ and $g^*(n) = g(n-1)\exp(\mathrm{j}\,\theta_i)$ for $n \geq 0$. In such a case $u(n)$ and $g(n)$ are identically zero. Thus, $\partial\epsilon/(\partial\rho_i) \geq 1$ for $\rho_i \geq 1$, and so we conclude that all the zeros of the transfer function $H_f(z)$ must lie inside the unit circle. That is, $H_f(z)$ is minimum phase.

Problem 9.5

For forward linear prediction, we have

a)

The data matrix is

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

The correlation matrix is $\Phi = \mathbf{A}^H\mathbf{A}$.

b)

The desired response vector is

$$\mathbf{d}^H = \begin{bmatrix} u(M+1) & u(M+2) & \dots & u(N) \end{bmatrix}$$

The cross-correlation vector is $\mathbf{A}^H\mathbf{d}$.

c)

The minimum value of E_f is

$$E_{f,\text{min}} = \mathbf{d}^H\mathbf{d} - \mathbf{d}^H\left(\mathbf{A}^H\mathbf{A}\right)^{-1}\mathbf{A}^H\mathbf{d}$$

Problem 9.6

For backward linear prediction, we have

a)

The data matrix is

$$\mathbf{A}^H = \begin{bmatrix} u^*(2) & u^*(3) & \dots & u^*(N-M+1) \\ u^*(3) & u^*(4) & \dots & u^*(N-M+2) \\ \vdots & \vdots & & \vdots \\ u^*(M+1) & u^*(M+2) & \dots & u^*(N) \end{bmatrix}$$

Note the difference between this data matrix and that for forward linear prediction in Problem 9.15.

The correlation matrix is

$$\Phi = \mathbf{A}^H \mathbf{A}$$

b)

The data vector is

$$\mathbf{d}^H = \begin{bmatrix} u^*(1) & u^*(2) & \dots & u^*(N-M) \end{bmatrix}$$

The cross correlation vector is $\mathbf{A}^H \mathbf{d}$.

c)

The minimum value of E_b is

$$E_{b,\text{min}} = \mathbf{d}^H \mathbf{d} - \mathbf{d}^H \left(\mathbf{A}^H \mathbf{A} \right)^{-1} \mathbf{d}$$

Problem 9.7

The data matrix is

$$\mathbf{A}^H = \begin{bmatrix} \mathbf{u}(M) & \mathbf{u}(M+1) & \dots & \mathbf{u}(N) \end{bmatrix}$$

The desired response vector is

$$\mathbf{d}^H = \begin{bmatrix} d(M) & d(M+1) & \dots & d(N) \end{bmatrix}$$

The cost function is

$$E(\mathbf{w}) = \mathbf{d}^H\mathbf{d} - \mathbf{w}^H\mathbf{z} - \mathbf{z}^H\mathbf{w} + \mathbf{w}^H\Phi\mathbf{w}$$

where $\mathbf{z}$ is the cross-correlation vector:

$$\mathbf{z} = \mathbf{A}^H\mathbf{d}$$

and Φ is the correlation matrix

$$\Phi = \mathbf{A}^H\mathbf{A}$$

Differentiating the cost function with respect ot the weight vector:

$$\frac{\partial E(\mathbf{w})}{\partial(\mathbf{w})} = -2\mathbf{z} + 2\Phi\mathbf{w}$$

Setting this result equal to zero and solving for the optimum $\mathbf{w}$, we get

$$\begin{aligned}\hat{\mathbf{w}} =& \Phi^{-1}\mathbf{z} \\ =& \left(\mathbf{A}^H\mathbf{A}\right)^{-1}\mathbf{A}^H\mathbf{d}\end{aligned}$$

Problem 9.8

$$E_{\text{reg}} = \sum_{n=1}^{K}\left|\mathbf{w}^H\mathbf{u}(n)\right|^2 + \lambda\left(\mathbf{w}^H\mathbf{s}(\theta) - 1\right) + \delta\left\|\mathbf{w}\right\|^2$$

Taking the derivative of E_{reg} with respect to the weight vector $\mathbf{w}$ and setting the result equal to zero:

$$\frac{\partial E_{\text{reg}}}{\partial\mathbf{w}} = \sum_{n=1}^{K} 2\mathbf{w}^H\mathbf{u}(n)\mathbf{u}^H(n) + \lambda\mathbf{s}(\theta) + 2\delta\mathbf{w} = \mathbf{0}$$

Hence,

$$\hat{\mathbf{w}} = \frac{-\lambda\mathbf{s}\left(\theta\right)}{2\left(\sum_{n=1}^{K}\mathbf{u}(n)\mathbf{u}^H(n) + \delta\mathbf{I}\right)}$$

Since

$$\hat{\mathbf{w}}^H \mathbf{s}(\theta) = 1 \tag{1}$$

$$-\lambda = 2\|\hat{\mathbf{w}}\|^2 \left(\sum_{n=1}^{K} \mathbf{u}(n)\mathbf{u}^H(n) + \delta \mathbf{I} \right) \tag{2}$$

By the virtue of Equation (9.80) in the textbook, we may have

$$\Phi \hat{\mathbf{w}} = -\frac{1}{2}\lambda \mathbf{s}(\theta) \tag{3}$$

Using Equations (1) and (2) in (3) and rearranging terms, we obtain

$$\Phi = \sum_{n=1}^{K} \mathbf{u}(n)\mathbf{u}^H(n) + \delta \mathbf{I}$$

Problem 9.9

Starting with the cost function

$$E = \sum_{i=M+1}^{N} \left(|f_M(i)|^2 + |b_M(i)|^2 \right)$$

$$f_M(i) = \mathbf{a}_M^H \mathbf{u}_{M+1}(i)$$
$$\mathbf{a}_M^H = \begin{bmatrix} 1 & -w_1 & -w_2 & \dots & -w_M \end{bmatrix} = \begin{bmatrix} 1 & -\mathbf{w}^T \end{bmatrix}$$
$$\mathbf{u}_{M+1}^H(i) = \begin{bmatrix} u(i) & u(i-1) & \dots & u(i-M) \end{bmatrix}$$
$$b_M(i) = \mathbf{a}_M^H \mathbf{u}_{M+1}^B(i)$$
$$\mathbf{u}_{M+1}^{BH} = \begin{bmatrix} u(i-M) & u(i-M+1) & \dots & u(i) \end{bmatrix}$$

a)

Using these definitions, we may rewrite the cost function as

$$E = \sum_{i=M+1}^{N} \left(\mathbf{a}_M^H \mathbf{u}_{M+1}(i)\mathbf{u}_{M+1}^H(i)\mathbf{a}_M + \mathbf{a}_M^H \mathbf{u}_{M+1}^B(i)\mathbf{u}_{M+1}^{BH}(i)\mathbf{a}_M \right)$$

By definition,

$$\mathbf{a}_M = \begin{bmatrix} 1 \\ -\mathbf{w} \end{bmatrix} \quad \text{and} \quad \mathbf{u}_{M+1}(i) = \begin{bmatrix} u(i) \\ \mathbf{u}_M(i-1) \end{bmatrix}$$

Hence,

$$E = \sum_{i=M+1}^{N} \left(|u(i)|^2 + \mathbf{w}^H \Phi_f \mathbf{w} - \theta_f^H \mathbf{w} - \mathbf{w}^H \theta_f + |u(i-M)|^2 + \mathbf{w}^H \Phi_b \mathbf{w} - \theta_b^H \mathbf{w} - \mathbf{w}^H \theta_b \right)$$

where

$$\Phi_f = \sum_{i=M+1}^{N} \mathbf{u}_M(i-1)\mathbf{u}_M^H(i-1)$$

$$\theta_f = \sum_{i=M+1}^{N} \mathbf{u}_M(i-1)u^*(i)$$

$$\Phi_b = \sum_{i=M+1}^{N} \mathbf{u}_M^{B*}(i)\mathbf{u}_M^{BH}(i)$$

$$\theta_b = \sum_{i=M+1}^{N} \mathbf{u}_M^{B}(i)u^*(i-M)$$

Setting

$$\frac{\partial E}{\partial \mathbf{w}} = 0$$

and solving for the optimum value of $\mathbf{w}$, we obtain

$$\hat{\mathbf{w}} = \phi^{-1}\theta$$

where

$$\Phi = \Phi_f + \Phi_b = \sum_{i=M+1}^{N} \left[\mathbf{u}_M (i-1)\, \mathbf{u}_M^H (i-1) + \mathbf{u}_M^{B*} (i)\, \mathbf{u}_M^{BH} (i) \right]$$

$$\theta = \theta_f + \theta_b = \sum_{i=M+1}^{N} \left[\mathbf{u}_M (i-1)\, u (i) + \mathbf{u}_M^{B*} (i)\, u (i-M) \right]$$

b)
Finally

$$E_{\text{min}} = E_{f,\text{min}} + E_{b,\text{min}} = \sum_{i=M+1}^{N} \left[|u(i)|^2 + |u(i-M)|^2\right] - \theta^H \hat{\mathbf{w}}$$

c)

$$\hat{\mathbf{a}} = \begin{bmatrix} 1 \\ -\hat{\mathbf{w}} \end{bmatrix}, \quad \phi\hat{\mathbf{a}} = \begin{bmatrix} E_{\text{min}} \\ \mathbf{0} \end{bmatrix}$$

$$\Phi = \Phi_f + \Phi_b = \sum_{i=M+1}^{N} \left[\mathbf{u}_M\left(i-1\right)\mathbf{u}_M^H\left(i-1\right) + \mathbf{u}_M^{B*}\left(i\right)\mathbf{u}_M^{BH}\left(i\right)\right] \tag{1}$$

Examining Equation (1) and noting that

$$\mathbf{u}_M^H(i) = \begin{bmatrix} u(i) & u(i-1) & \dots & u(i-M) \end{bmatrix}$$

$$\mathbf{u}_M^{BH}(i) = \begin{bmatrix} u(i-M) & u(i-M+1) & \dots & u(i) \end{bmatrix}$$

we readily find that

$$\begin{aligned} &\Phi(M-k, M-t) = \Phi^*(t,k), && 0 \le (k,t) \le M \\ &\Phi(k,t) = \Phi^*(t,k), && 0 \le (k,t) \le M \end{aligned}$$

Problem 9.10

The signal-to-interference ratio (SIR) maximization problem may be stated as follows:

$$\max_{w} \left(\frac{\mathbf{w}^H \mathbf{s}\mathbf{s}^H \mathbf{w}}{\mathbf{w}^H \mathbf{R}\mathbf{w}}\right) \quad \text{subject to } \mathbf{C}_{N-1}^H \mathbf{w} = \mathbf{f}_{N-1}$$

a)
Using Lagrange's method, the SIR maximization problem can be written as a minimization problem

$$\min J(\mathbf{w}) = \min\left\{\frac{1}{2}\mathbf{w}^H\mathbf{R}\mathbf{w} + \lambda^H\left(\mathbf{C}_{N-1}^H\mathbf{w} - \mathbf{f}_{N-1}\right)\right\}$$

where $\lambda^H = \begin{bmatrix} \lambda_1 & \dots & \lambda_{N-1} \end{bmatrix}$ is the Lagrange vector $(0 < \lambda_i < 1)$. Set:

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{0}$$

Hence, we obtain

$$\begin{aligned} \mathbf{R}\mathbf{w} + \mathbf{C}_{N-1}^{H}\lambda &= \mathbf{0} \\ \mathbf{w}_{\text{opt}} &= -\mathbf{R}^{-1}\mathbf{C}_{N-1}\lambda \end{aligned} \tag{1}$$

Since $\mathbf{C}_{N-1}^{H}\mathbf{w}_{\text{opt}} = \mathbf{f}_{N-1} = -\mathbf{C}_{N-1}^{H}\mathbf{R}^{-1}\mathbf{C}_{N-1}\lambda$

$$\lambda = -\left[\mathbf{C}_{N-1}^{H}\mathbf{R}^{-1}\mathbf{C}_{N-1}\right]^{-1}\mathbf{f}_{N-1} \tag{2}$$

Using Equation (2) in (1):

$$\mathbf{w}_{\text{opt}} = \mathbf{R}^{-1}\mathbf{C}\left[\mathbf{C}^{H}\mathbf{R}^{-1}\mathbf{C}\right]^{-1}\mathbf{f}$$

b)

When $\mathbf{w}_{\text{opt}} = \mathbf{R}^{-1}\mathbf{C}\left[\mathbf{C}^{H}\mathbf{R}^{-1}\mathbf{C}\right]^{-1}\mathbf{f}$, the SIR becomes

$$\begin{aligned} \frac{\mathbf{w}^{H}\mathbf{s}\mathbf{s}^{H}\mathbf{w}}{\mathbf{w}^{H}\mathbf{R}\mathbf{w}} &= \frac{\mathbf{w}^{H}\mathbf{s}\mathbf{s}^{H}\mathbf{w}}{\mathbf{f}^{H}\left[\mathbf{C}^{H}\mathbf{R}^{-1}\mathbf{C}\right]^{-H}\mathbf{C}^{H}\mathbf{R}^{-H}\mathbf{C}\left[\mathbf{C}^{H}\mathbf{R}^{-1}\mathbf{C}\right]^{-1}\mathbf{f}} \\ &= \frac{\mathbf{f}^{H}\left[\mathbf{C}^{H}\mathbf{R}^{-1}\mathbf{C}\right]^{-H}\mathbf{C}^{H}\mathbf{s}\mathbf{s}^{H}\mathbf{R}^{-1}\mathbf{C}\left[\mathbf{C}^{H}\mathbf{R}^{-1}\mathbf{C}\right]^{-1}\mathbf{f}}{\mathbf{f}^{H}\left[\mathbf{C}^{H}\mathbf{R}^{-1}\mathbf{C}\right]^{-H}\mathbf{C}^{H}\mathbf{R}^{-H}\mathbf{C}\left[\mathbf{C}^{H}\mathbf{R}^{-1}\mathbf{C}\right]^{-1}\mathbf{f}} \end{aligned}$$

c)

When there are no auxiliary constants $\mathbf{C}_{n-1} = 0$, the fixed value of $\mathbf{f}_0$ will only determine the normalization of the solution yielding an unconditional maximum for SIR:

$$\mathbf{w}_{\text{opt}} = \alpha\mathbf{R}^{-1}\mathbf{s} \quad (\alpha \text{ is a constnat})$$

$$\text{SIR}_{\text{max}} = \frac{\mathbf{s}^{H}\mathbf{R}^{-H}\mathbf{s}\mathbf{s}^{H}\mathbf{R}^{-1}\mathbf{s}}{\mathbf{s}^{H}\mathbf{R}^{-H}\mathbf{s}} = \mathbf{s}^{H}\mathbf{R}^{-1}\mathbf{s}$$

d)

$$\hat{\mathbf{R}}(n) = \lambda\hat{\mathbf{R}}(n-1) + \mathbf{u}(n)\mathbf{u}^{H}(n)$$

where $0 < \lambda < 1$; the parameter λ is a weighting factor, not to be confused with the Lagrange multiplier.

Problem 9.11

a) We are given

$$\mathbf{A} = \begin{bmatrix} 1 & -1 \\ 0.5 & 2 \end{bmatrix}$$

Therefore,

$$\begin{aligned} \mathbf{A}^T\mathbf{A} &= \begin{bmatrix} 1 & 0.5 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 0.5 & 2 \end{bmatrix} \\ &= \begin{bmatrix} 1.25 & 0 \\ 0 & 5 \end{bmatrix} \end{aligned}$$

This is a diagonal matrix. Hence, its eigenvalues are

$$\lambda_1 = 1.25, \quad \lambda_2 = 5$$

The singular values of the matrix $\mathbf{A}$ are therefore

$$\sigma_1 = \sqrt{\lambda_1} = \sqrt{1.25} = \frac{\sqrt{5}}{2}$$

$$\sigma_2 = \sqrt{\lambda_2} = \sqrt{5}$$

The eigenvectors of $\mathbf{A}^T\mathbf{A}$ are the right singular vectors of $\mathbf{A}$. For the problem at hand, the eigenvectors of $\mathbf{A}^T\mathbf{A}$ constitute a unit matrix. We therefore have

$$\mathbf{V} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

b)
The matrix product $\mathbf{A}\mathbf{A}^T$ is

$$\begin{aligned} \mathbf{A}\mathbf{A}^T &= \begin{bmatrix} 1 & -1 \\ 0.5 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0.5 \\ -1 & 2 \end{bmatrix} \\ &= \begin{bmatrix} 2 & -1.5 \\ -1.5 & 4.25 \end{bmatrix} \end{aligned}$$

The eigenvalues of $\mathbf{A}\mathbf{A}^T$ are the roots of the characteristic equation

$$(2 - \lambda)(4.25 - \lambda) - (1.5)^2 = 0$$

Expanding this equation:

$$\lambda^2 - 6.25\lambda + 6.25 = 0$$

Solving for the roots of this quadratic equation:

$$\lambda_1 = 1.25, \quad \lambda_2 = 5$$

The singular values of the data matrix $\mathbf{A}$ are therefore

$$\sigma_1 = \sqrt{\lambda_1} = \frac{\sqrt{5}}{2}$$

$$\sigma_2 = \sqrt{\lambda_2} = \sqrt{5}$$

which are identical to the values calculated in part **a)**. To find the eigenvectors of $\mathbf{A}\mathbf{A}^T$, we note that

$$\begin{bmatrix} 2 & -1.5 \\ -1.5 & 4.25 \end{bmatrix} \begin{bmatrix} q_{11} \\ q_{12} \end{bmatrix} = 1.25 \begin{bmatrix} q_{11} \\ q_{12} \end{bmatrix}$$

Hence,

$$0.75q_{11} - 1.5q_{12} = 0$$

Equivalently,

$$q_{11} = 2q_{12}$$

Setting $q_{11}^2 + q_{12}^2 = 1$, we find that

$$q_{11} = \pm\frac{1}{\sqrt{5}} \text{ and } q_{12} = \pm\frac{2}{\sqrt{5}}$$

We may thus write

$$\mathbf{q}_1 = \frac{1}{\sqrt{5}} \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

Similarly, we may show that

$$\mathbf{q}_2 = \frac{1}{\sqrt{5}} \begin{bmatrix} 2 \\ -1 \end{bmatrix}$$

The eigenvectors of $\mathbf{A}\mathbf{A}^T$ are the left singular vectors of $\mathbf{A}$. We may thus express the left singular vectors of matrix $\mathbf{A}$ as

$$\mathbf{U} = \frac{1}{\sqrt{5}} \begin{bmatrix} 1 & 2 \\ 2 & -1 \end{bmatrix}$$

The pseudoinverse of the matrix $\mathbf{A}$ is given by

$$\begin{aligned} \mathbf{A}^+ =& \mathbf{V} \begin{bmatrix} \Sigma^{-1} & 0 \\ 0 & 0 \end{bmatrix} \mathbf{U}^T \\ =& \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{2}{\sqrt{5}} & 0 \\ 0 & \frac{1}{\sqrt{5}} \end{bmatrix} \frac{1}{\sqrt{5}} \begin{bmatrix} 1 & 2 \\ 2 & -1 \end{bmatrix} \\ =& \frac{1}{5} \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 2 & -1 \end{bmatrix} \\ =& \frac{1}{5} \begin{bmatrix} 2 & 4 \\ 2 & -1 \end{bmatrix} \end{aligned}$$

Problem 9.12

Given the 2×2 complex matrix

$$\mathbf{A} = \begin{bmatrix} 1+\mathrm{j} & 1+0.5\,\mathrm{j} \\ 0.5-\mathrm{j} & 1-\mathrm{j} \end{bmatrix}$$

a)

We have

$$\mathbf{A}^H\mathbf{A} = \begin{bmatrix} 3.25 & 3 \\ 3 & 3.25 \end{bmatrix}$$

The eigenvalues and eigenvector os the matrix product $\mathbf{A}^H\mathbf{A}$ are found in the usual way be first solving

$$|\mathbf{A} - \lambda\mathbf{I}| = 0$$

for the eigenvalues λ_i for $i = 1, 2$, and then using these value in

$$\mathbf{A}^H\mathbf{A} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \lambda_i \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

to determine the eigenvectors $\mathbf{x}$.

The two eigenvalues in descending order for $\mathbf{A}^H\mathbf{A}$ are $\lambda_1 = 6.25$ and $\lambda_2 = 0.25$. Regarding the eigenvectors, we have

$$\mathbf{V} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

b)

Next, we have

$$\mathbf{A}\mathbf{A}^H = \begin{bmatrix} 3.25 & 3\,\mathrm{j} \\ -3\,\mathrm{j} & 3.25 \end{bmatrix}$$

The eigenvalues and eigenvector os the matrix product $\mathbf{A}^H\mathbf{A}$ are found in the usual way be first solving

$$|\mathbf{A} - \lambda\mathbf{I}| = 0$$

for the eigenvalues λ_i for $i = 1, 2$, and then using these value in

$$\mathbf{A}\mathbf{A}^H \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \lambda_i \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

to determine the eigenvectors $\mathbf{x}$.

The two eigenvalues in descending order for $\mathbf{A}^H\mathbf{A}$ are $\lambda_1 = 6.25$ and $\lambda_2 = 0.25$. Regarding the eigenvectors, we have

$$\mathbf{V} = \frac{1}{\sqrt{2}}\begin{bmatrix} \mathrm{j} & 1 \\ \mathrm{j} & -1 \end{bmatrix}$$

c)

The SVD of $\mathbf{A}$ is given by

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^H$$

where the right singular matrix $\mathbf{V}$ is the eigen-matrix of $\mathbf{A}^H\mathbf{A}$ and the left singular matrix $\mathbf{U}$ is the eigen-matrix of $\mathbf{A}\mathbf{A}^H$. The singular values should be the square roots of the common eigenvalues: $\sigma_1 = \sqrt{\lambda_1} = 2.5$ and $\sigma_2 = 0.5$ with

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}$$

Interestingly enough, we find that

$$\mathbf{U}\mathbf{\Sigma}\mathbf{V}^H = \mathbf{B} = \begin{bmatrix} 1.25 + \mathrm{j}\,0.25 & -1.25 + \mathrm{j}\,0.25 \\ 0.25 + \mathrm{j}\,1.25 & 0.25 - \mathrm{j}\,1.25 \end{bmatrix}$$

where, obviously $\mathbf{B} \neq \mathbf{A}$, but, $\mathbf{B}^H\mathbf{B} = \mathbf{A}^H\mathbf{A}$ and $\mathbf{B}\mathbf{B}^H = \mathbf{A}\mathbf{A}^H$. In order to have $\mathbf{B} = \mathbf{A}$, the arbitrariness in choosing the eigenvectors of $\mathbf{A}^H\mathbf{A}$ and $\mathbf{A}\mathbf{A}^H$ above, should be removed. The natural constraint relation is that $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^H$ hold true.

Starting from that premise, we take $\mathbf{V}$ as is and define $\mathbf{U}$ using

$$\mathbf{U} = \mathbf{A}\mathbf{V}\mathbf{\Sigma}^{-1} = \begin{bmatrix} 0.5657 + \mathrm{j}\,0.4243 & \mathrm{j}\,0.7071 \\ 0.4243 - \mathrm{j}\,0.5657 & -0.7071 \end{bmatrix}$$

We can easily verify that the new matrix $\mathbf{U}$ is indeed composed of eigenvectors of $\mathbf{A}\mathbf{A}^H$ determined above. Clearly, the new $\mathbf{U}$ is related to the old one by some similarity transformation

$$\mathbf{U}_{\text{new}} = \mathbf{W}^H\mathbf{U}_{\text{old}}\mathbf{W}$$

where $\mathbf{W}$ is a unitary matrix.

This problem illustrates the need for a less arbitrary procedure for determining the SVD of a matrix, even in the simplest case, with a 2×2 matrix.

Problem 9.13

We are given

$$\mathbf{A} = \begin{bmatrix} 2 & 3 \\ 1 & 2 \\ -1 & 1 \end{bmatrix}$$

We note that

$$\mathbf{A}^T = \begin{bmatrix} 2 & 1 & -1 \\ 3 & 2 & 1 \end{bmatrix}$$

Next, we set up

$$\mathbf{A}^T\mathbf{A} = \begin{bmatrix} 2 & 1 & -1 \\ 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} 2 & 3 \\ 1 & 2 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 6 & 7 \\ 7 & 14 \end{bmatrix}$$

The eigenvalues of $\mathbf{A}^T\mathbf{A}$ are roots of the characteristic equation:

$$(6-\lambda)(14-\lambda)-(7)^2=0$$

or,

$$\lambda^2-20\lambda+35=0$$

Solving the roots of this equation, we thus get

$$\lambda=10\pm\sqrt{65}$$

That is,

$$\begin{aligned}\lambda_1&=10-\sqrt{65}\approx 1.94, &\qquad \sigma_1&=\sqrt{\lambda_1}=1.393\\ \lambda_2&=10+\sqrt{65}\approx 18.06, &\qquad \sigma_2&=\sqrt{\lambda_2}=4.25\end{aligned}$$

To find the eigenvectors of $\mathbf{A}^T\mathbf{A}$, we note that

$$\begin{bmatrix}6 & 7\\ 7 & 14\end{bmatrix}\begin{bmatrix}q_{11}\\ q_{12}\end{bmatrix}=1.94\begin{bmatrix}q_{11}\\ q_{12}\end{bmatrix}$$

Hence,

$$4.06q_{11}+7q_{12}=0$$

or

$$q_{12}=-0.58q_{11}$$

We also note that

$$q_{11}^2+q_{12}^2=1$$

Therefore,

$$q_{11}^2+0.58^2q_{11}^2=1$$

or

$$q_{11}=\pm\frac{1}{\sqrt{1.336}}=\pm 0.87$$

Hence,

$$q_{12} = \mp 0.505$$

Similarly, we may show that the eigenvector associated with $\lambda_2 = 19.06$ is defined by

$$q_{21} = -0.505$$
$$q_{22} = -0.87$$

Therefore, the right singular vectors of the data matrix $\mathbf{A}$ constitute the matrix

$$\mathbf{V} = \begin{bmatrix} 0.87 & -0.505 \\ -0.505 & -0.87 \end{bmatrix} \tag{1}$$

Next, we set up

$$\mathbf{A}\mathbf{A}^T = \begin{bmatrix} 2 & 3 \\ 1 & 2 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & -1 \\ 3 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 13 & 8 & 1 \\ 8 & 5 & 1 \\ 1 & 1 & 2 \end{bmatrix}$$

The eigenvalues of this matrix can be found as

$$\det \begin{bmatrix} 13-\lambda & 8 & 1 \\ 8 & 5-\lambda & 1 \\ 1 & 1 & 2-\lambda \end{bmatrix} = 0$$

or

$$\lambda \left(\lambda^2 - 20\lambda + 25\right) = 0$$

The eigenvalues of $\mathbf{A}\mathbf{A}^T$ are therefore as follows

$$\lambda_0 = 0$$
$$\lambda_1 \approx 1.94$$
$$\lambda_2 \approx 18.06$$

The two nonzero eigenvalues of $\mathbf{A}\mathbf{A}^T$ are the same as those found for $\mathbf{A}^T\mathbf{A}$. To find the eigenvectors of $\mathbf{A}\mathbf{A}^T$, we note that for $\lambda_1 = 1.94$:

$$\begin{bmatrix} 13 & 8 & 1 \\ 8 & 5 & 1 \\ 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} q_{11} \\ q_{12} \\ q_{13} \end{bmatrix} = 1.94 \begin{bmatrix} q_{11} \\ q_{12} \\ q_{13} \end{bmatrix}$$

$$\begin{array}{rcrcrcl} 11.06q_{11} & + & 8q_{12} & + & q_{13} & = & 0 \\ 8q_{11} & + & 3.06q_{12} & + & q_{13} & = & 0 \\ q_{11} & + & q_{12} & + & 0.06q_{13} & = & 0 \end{array}$$

Using the first two equations to eliminate q_{13}:

$$3.06q_{11} + 4.94q_{12} = 0$$

Hence,

$$q_{12} \approx -0.619q_{11}$$

Using the first and third equations to eliminate q_{12}:

$$3.06q_{11} + 0.52q_{13} = 0$$

Hence,

$$q_{13} \approx -5.88q_{11}$$

Next, we note that

$$q_{11}^2 + q_{12}^2 + q_{13}^2 = 1$$

Therefore,

$$\begin{array}{l} q_{11}^2 + 0.383q_{11}^2 + 34.574q_{11}^2 = 1 \\ 36q_{11}^2 \approx 1 \\ q_{11} \approx \pm 0.167 \end{array}$$

Correspondingly,

$$\begin{array}{l} q_{12} \approx \mp 0.109 \\ q_{13} \approx \mp 0.167 \end{array}$$

We may thus set

$$\mathbf{q}_1 \approx \begin{bmatrix} 0.167 \\ -0.109 \\ -0.981 \end{bmatrix} \tag{2}$$

Using a similar method for the eigenvalue $\lambda_2 = 19.06$, we may find the eigenvector

$$\mathbf{q}_2 = \begin{bmatrix} -0.845 \\ -0.524 \\ -0.0845 \end{bmatrix} \tag{3}$$

Note that $\mathbf{q}_2$ is orthogonal to $\mathbf{q}_1$, as it should be; that is,

$$\mathbf{q}_1^T \mathbf{q}_2 = 0.$$

To complete the eigencomputation, we need to determine the eigenvector associated with the zero eigenvalue of matrix $\mathbf{A}\mathbf{A}^T$. Here we note

$$\begin{bmatrix} 13 & 8 & 1 \\ 8 & 5 & 1 \\ 1 & 1 & 2 \end{bmatrix} \mathbf{q}_0 = \mathbf{0}$$

where $\|\mathbf{q}_0\|^2 = 1$. Solving for $\mathbf{q}_0$, we get

$$\mathbf{q}_0 = \begin{bmatrix} 0.5071 \\ -0.8452 \\ 0.169 \end{bmatrix} \tag{4}$$

Putting together Equations (2) through (4) for the eigenvectors $\mathbf{q}_1$, $\mathbf{q}_2$, and $\mathbf{q}_0$, we may express the left singular vectors of the data matrix $\mathbf{A}$ as the matrix

$$\mathbf{U} = \begin{bmatrix} 0.167 & 0.845 & 0.5071 \\ -0.109 & 0.524 & -0.8452 \\ -0.981 & 0.088 & 0.169 \end{bmatrix}$$

where the third column corresponds to the zero eigenvalue of $\mathbf{A}\mathbf{A}^T$. The singular value decomposition of data matrix $\mathbf{A}$ may therefore be expressed as

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} 0.167 & 0.845 & 0.5071 \\ -0.109 & 0.524 & -0.8452 \\ -0.981 & 0.088 & 0.169 \end{bmatrix} \begin{bmatrix} 1.393 & 0 \\ 0 & 4.25 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0.87 & -0.505 \\ -0.505 & -0.87 \end{bmatrix} \\ &= \begin{bmatrix} 0.167 & 0.845 \\ -0.109 & 0.524 \\ -0.981 & 0.088 \end{bmatrix} \begin{bmatrix} 1.393 & 0 \\ 0 & 4.25 \end{bmatrix} \begin{bmatrix} 0.87 & -0.505 \\ -0.505 & -0.87 \end{bmatrix} \end{aligned}$$

As a check, carrying out the matrix multiplication given here, we get

$$\mathbf{A} = \begin{bmatrix} 2.016 & 3.0069 \\ 0.9925 & 2.0142 \\ -1.0065 & 1.0044 \end{bmatrix}$$

which is very close to the original data matrix.

a)
The pseudoinverse of matrix $\mathbf{A}$ is

$$\mathbf{A}^{+} = \sum_{i=1}^{2} \frac{1}{\sigma_i}\mathbf{v}_i\mathbf{u}_i^T$$

b)
The least-squares weight vector is

$$\begin{aligned}
\hat{\mathbf{w}} =& \mathbf{A}^{+}\mathbf{d} \\
=& \sum_{i=1}^{2} \frac{\left(\mathbf{u}_i^T\mathbf{d}\right)}{\sigma_i}\mathbf{v}_i \\
=& \frac{\left(\mathbf{u}_1^T\mathbf{d}\right)}{\sigma_1}\mathbf{v}_1 + \frac{\left(\mathbf{u}_2^T\mathbf{d}\right)}{\sigma_2}\mathbf{v}_2 \\
=& \frac{1}{1.393}\begin{bmatrix}0.167 & 0.109 & -0.981\end{bmatrix}\begin{bmatrix}2\\1\\1/24\end{bmatrix}\begin{bmatrix}0.87\\-0.505\end{bmatrix} \\
& + \frac{1}{4.250}\begin{bmatrix}0.845 & 0.524 & 0.085\end{bmatrix}\begin{bmatrix}2\\1\\1/34\end{bmatrix}\begin{bmatrix}0.505\\0.87\end{bmatrix} \\
=& \begin{bmatrix}0.3859\\0.3859\end{bmatrix}
\end{aligned}$$

Problem 9.14

First, we set up the following identifications:

	Least-squares	Normalized LMS
Data matrix	$\mathbf{A}$	$\mathbf{x}^H(n)$
Desired data vector	$\mathbf{d}$	$e^*(n)$
Parameter vector	$\mathbf{w}$	$\mathbf{c}(n+1)$
Eigenvalue	σ_i^2	$\|\|\mathbf{x}(n)\|\|^2$
Eigenvector	$\mathbf{u}_i$	1

Hence, the application of the linear least-squares solution yields

$$\mathbf{c}(n+1) = \frac{1}{\|\mathbf{x}(n)\|^2}\mathbf{x}(n)e^*(n)$$

That is to say,

$$\delta\hat{\mathbf{w}}(n+1) = \frac{1}{\|\mathbf{u}(n)\|^2 + a}\mathbf{u}(n)e^*(n)$$

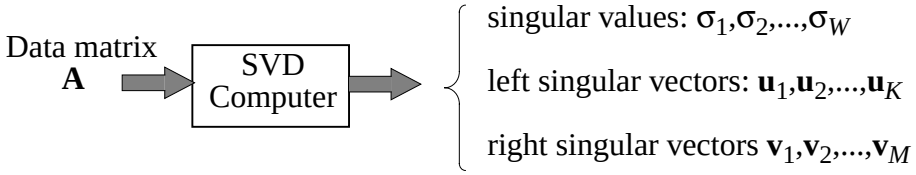
Correspondingly, we may write

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{\mu}{\|\mathbf{u}(n)\|^2 + a}\mathbf{u}(n)e^*(n)$$

where μ is the step-size parameter.

Problem 9.15

We are given an SVD computer that may be pictured as follows:



W: rank of data matrix **A**
K: number of rows of **A**
M: number of columns of **A**

The MVDR spectrum is defined by

$$S_{\text{MVDR}}(\omega) = \frac{1}{\mathbf{s}^H(\omega)\Phi^{-1}\mathbf{S}(\omega)} \tag{1}$$

where $\mathbf{s}(\omega)$ is the steering vector, and Φ^{-1} is the inverse of the correlation matrix Φ. Specifically, Φ is defined in terms of the data matrix $\mathbf{A}$ by

$$\Phi = \mathbf{A}^H\mathbf{A}$$

where

$$\mathbf{A} = \sum_{i=1}^{W} \sigma_i \mathbf{u}_i \mathbf{v}_i^H$$

$$\mathbf{A}^H = \sum_{i=1}^{W} \sigma_i \mathbf{v}_i \mathbf{u}_i^H$$

That is,

$$\Phi = \sum_{i=1}^{W} \sigma_i^2 \mathbf{v}_i \mathbf{v}_i^H, \quad \|\mathbf{u}_i\|^2 = 1 \text{ for all } i$$

Correspondingly, we may express the inverse of matrix Φ^{-1} as

$$\Phi^{-1} = \sum_{i=1}^{W} \frac{1}{\sigma_i^2} \mathbf{v}_i \mathbf{v}_i^H, \quad \|\mathbf{u}_i\|^2 = 1 \text{ for all } i \tag{2}$$

Hence, the denominator of the MVDR spectrum may be expressed as

$$\begin{aligned} \mathbf{s}^H(\omega)\Phi^{-1}\mathbf{s}(\omega) &= \sum_{i=1}^{W} \frac{1}{\sigma_i^2} \mathbf{s}^H(\omega)\mathbf{v}_i\mathbf{v}_i^H\mathbf{s}(\omega) \\ &= \sum_{i=1}^{W} |z_i(\omega)|^2 \end{aligned}$$

where $z_i(\omega)$ is a frequency-dependent scalar that is defined by the inner product:

$$z_i(\omega) = \frac{1}{\sigma_i} \mathbf{S}^H(\omega)\mathbf{v}_i, \quad i = 1, 2, \ldots, W \tag{4}$$

Accordingly, Equation (2) may be rewritten as

$$S_{\text{MVDR}}(\omega) = \frac{1}{\sum_{i=1}^{W} |z_i(\omega)|} \tag{5}$$

Thus, to evaluate the MVDR spectrum, we may proceed as follows:

- Compute the SVD of the data matrix $\mathbf{A}$.
- Use the right-singular vectors $\mathbf{v}_i$ ad the corresponding singular values σ_i in Equation (4) to evaluate $z_i(\omega)$ for $i = 1, 2, \ldots, W$, where W is the rank of $\mathbf{A}$.
- Use Equation (5) to evaluate the MVDR spectrum.

Chapter 10

Problem 10.1

Assume that

$$\beta(n,i) = \lambda(i)\beta(n,i-1), \quad i = 1,\ldots,n$$

Hence, for $i = n$:

$$\beta(n,n) = \lambda(n)\beta(n,n-1), \quad i = 1,\ldots,n$$

Since $\beta(n,n) = 1$, we have

$$\lambda^{-1}(n) = \beta(n,n-1)$$

Next, for $i = n-1$

$$\beta(n,n-1) = \lambda(n-1)\beta(n,n-2), \quad i = 1,\ldots,n$$

or equivalently,

$$\begin{aligned}\beta(n,n-2) =& \lambda^{-1}(n-1)\beta(n,n-1)\\ =& \lambda^{-1}(n-1)\lambda^{-1}(n)\end{aligned}$$

Proceeding in this way, we may thus write

$$\begin{aligned}\beta(n,i) =& \lambda^{-1}(i+1)\ldots\lambda^{-1}(n-1)\lambda^{-1}(n)\\ =& \prod_{k=i+1}^{n}\lambda^{-1}(k)\end{aligned}$$

For $\beta(n,i)$ to equal λ^{n-i}, we must have

$$\lambda^{-1}(i+1)\ldots\lambda^{-1}(n-1)\lambda^{-1}(n) = \lambda^{n-i}$$

This requirement is satisfied by choosing

$$\lambda(k) = \lambda^{-1} \quad \text{for all } k$$

We thus find that

$$\beta(n,i) = \lambda^{n-i}$$

Problem 10.2

The matrix inversion lemma states that if

$$\mathbf{A} = \mathbf{B}^{-1} + \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^H \tag{1}$$

then

$$\mathbf{A}^{-1} = \mathbf{B} - \mathbf{B}\mathbf{C}\left(\mathbf{D} + \mathbf{C}^H\mathbf{B}\mathbf{C}\right)^{-1}\mathbf{C}^H\mathbf{B} \tag{2}$$

To prove this lemma, we multiply Equation (1) by (2):

$$\begin{aligned}\mathbf{A}\mathbf{A}^{-1} =& \left(\mathbf{B}^{-1} + \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^H\right)\left[\mathbf{B} - \mathbf{B}\mathbf{C}\left(\mathbf{D} + \mathbf{C}^H\mathbf{B}\mathbf{C}\right)^{-1}\mathbf{C}^H\mathbf{B}\right] \\ =& \mathbf{B}^{-1}\mathbf{B} - \mathbf{B}^{-1}\mathbf{B}\mathbf{C}\left(\mathbf{D} + \mathbf{C}^H\mathbf{B}\mathbf{C}\right)^{-1}\mathbf{C}^H\mathbf{B} \\ & - \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^H\mathbf{B}\mathbf{C}\left(\mathbf{D} + \mathbf{C}^H\mathbf{B}\mathbf{C}\right)^{-1}\mathbf{C}^H\mathbf{B} + \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^H\mathbf{B}\end{aligned}$$

We have to show that $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$. Since $\left(\mathbf{D} + \mathbf{C}^H\mathbf{B}\mathbf{C}\right)^{-1}\left(\mathbf{D} + \mathbf{C}^H\mathbf{B}\mathbf{C}\right) = \mathbf{I}$, and $\mathbf{B}^{-1}\mathbf{B} = \mathbf{I}$, we may rewrite this result as

$$\begin{aligned}\mathbf{A}\mathbf{A}^{-1} =& \mathbf{I} - \mathbf{C}\left(\mathbf{D} + \mathbf{C}^H\mathbf{B}\mathbf{C}\right)^{-1}\mathbf{C}^H\mathbf{B} \\ & + \mathbf{C}\mathbf{D}^{-1}\left(\mathbf{D} + \mathbf{C}^H\mathbf{B}\mathbf{C}\right)\left(\mathbf{D} + \mathbf{C}^H\mathbf{B}\mathbf{C}\right)^{-1}\mathbf{C}^H\mathbf{B} \\ & - \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^H\mathbf{B}\mathbf{C}\left(\mathbf{D} + \mathbf{C}^H\mathbf{B}\mathbf{C}\right)^{-1}\mathbf{C}^H\mathbf{B} \\ =& \mathbf{I} - \left[\mathbf{C} - \mathbf{C}\mathbf{D}^{-1}\left(\mathbf{D} + \cancel{\mathbf{C}^H\mathbf{B}\mathbf{C}}\right) - \cancel{\mathbf{C}\mathbf{D}^{-1}\mathbf{C}^H\mathbf{B}\mathbf{C}}\right]\left(\mathbf{D} + \mathbf{C}^H\mathbf{B}\mathbf{C}\right)^{-1}\mathbf{C}^H\mathbf{B} \\ =& \mathbf{I} - \left(\mathbf{C} - \mathbf{C}\mathbf{D}^{-1}\mathbf{D}\right)\left(\mathbf{D} + \mathbf{C}^H\mathbf{B}\mathbf{C}\right)^{-1}\mathbf{C}^H\mathbf{B}\end{aligned}$$

Since $\mathbf{D}^{-1}\mathbf{D} = \mathbf{I}$, the second term in the last line is zero; hence,

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$$

which is the desired result.

Problem 10.3

We are given

$$\Phi(n) = \delta\mathbf{I} + \mathbf{u}(n)\mathbf{u}^H(n) \tag{1}$$

Let

$$\mathbf{A} = \mathbf{B}^{-1} + \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^H \tag{2}$$

Then, according to the matrix inversion lemma:

$$\mathbf{A}^{-1} = \mathbf{B} - \mathbf{B}\mathbf{C}\left(\mathbf{D} + \mathbf{C}^H\mathbf{B}\mathbf{C}\right)^{-1}\mathbf{C}^H\mathbf{B} \tag{3}$$

From Equations (1) and (2), we note:

$$\begin{aligned}\mathbf{A} &= \Phi(n)\\ \mathbf{B}^{-1} &= \delta\mathbf{I}(n)\\ \mathbf{C} &= \mathbf{u}(n)\\ \mathbf{D} &= \mathbf{I}\end{aligned}$$

Hence, using Equation (3):

$$\begin{aligned}\Phi^{-1}(n) =& \frac{1}{\delta}\mathbf{I} - \frac{1}{\delta^2}\mathbf{u}(n)\left(1 + \frac{1}{\delta}\mathbf{u}^H(n)\mathbf{u}(n)\right)^{-1}\mathbf{u}^H(n)\\ =& \frac{1}{\delta}\left(\frac{\mathbf{u}(n)\mathbf{u}^H(n)}{\delta + \mathbf{u}^H(n)\mathbf{u}(n)}\right)\\ =& \frac{1}{\delta}\left(\frac{\delta\mathbf{I}}{\delta + \mathbf{u}^H(n)\mathbf{u}(n)}\right)\\ =& \left(\frac{1}{\delta + \mathbf{u}^H(n)\mathbf{u}(n)}\right)\mathbf{I}\end{aligned}$$

Problem 10.4

From Equation (10.45) Section 10.6 of the textbook, we have

$$\hat{w}(n) = \hat{w}(n-1) + k(n)\xi^*(n) \tag{1}$$

Define

$$\varepsilon(n) = w_0 - \hat{w}(n)$$

We may thus write

$$\varepsilon(n) = \varepsilon(n-1) - k(n)\xi^*(n) \tag{2}$$

Since

$$\xi(n) = d(n) - \hat{w}^*(n-1)u(n)$$

$$e_0(n) = d(n) - w_0^* u(n)$$

We may expand Equation (2) as follows:

$$\begin{aligned}
\varepsilon(n) =& \varepsilon(n-1) - k(n)\left(d\left(n\right) - \hat{w}^*\left(n-1\right)u\left(n\right)\right)^* \\
=& \varepsilon(n-1) - k(n)(d\left(n\right) - w_0^* u(n) - \hat{w}^*(n-1)u(n))^* \\
=& \varepsilon(n-1) - k(n)\left(e_0(n) + \varepsilon^*(n-1)u(n)\right)^* \\
=& \varepsilon(n-1) - k(n)e_0^*(n) - k(n)u^*(n)\varepsilon(n-1) \\
=& 1 - k(n)u^*\left(n\right)\varepsilon(n-1) - k(n)e_0^*(n)
\end{aligned}$$

Hence,

$$a(n) = 1 - k(n)u^*(n) < 1, \text{ provided that } 0 \leq |k(n)u^*(n)| \leq 1$$

Then under this condition, $\varepsilon(n)$ is guaranteed to decrease with increasing n. The convergence process is perturbed by the white noise $e_0(n)$

Problem 10.5

From Equation (10.25) of the textbook:

$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1) + \mathbf{k}(n)\xi^*(n) \tag{1}$$

where we have

$$\begin{aligned}
\hat{\mathbf{w}}(1) &= \hat{\mathbf{w}}(0) + \mathbf{k}(1)\xi^*(1) \\
\hat{\mathbf{w}}(2) &= \hat{\mathbf{w}}(1) + \mathbf{k}(2)\xi^*(2) \\
&\vdots \\
\hat{\mathbf{w}}(n) &= \hat{\mathbf{w}}(n-1) + \mathbf{k}(n)\xi^*(n)
\end{aligned}$$

Summing the above n equations, we have

$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(0) + \sum_{i=1}^{n} \mathbf{k}(i)\xi^*(i)$$

Hence

$$\begin{aligned}\varepsilon(n) =& \mathbf{w}_0 - \hat{\mathbf{w}}(n) \\ =& \mathbf{w}_0 - \hat{\mathbf{w}}(0) - \sum_{i=1}^{n} \mathbf{k}(i)\xi^*(i) \\ =& \varepsilon(0) - \sum_{i=1}^{n} \mathbf{k}(i)\xi^*(i)\end{aligned}$$

Problem 10.6

The a posteriori estimation error $e(n)$ and the a priori estimation error $\xi(n)$ are related by

$$e(n) = \gamma(n)\xi(n)$$

where $\gamma(n)$ is defined by [see Equation (10.42) of the textbook]

$$\gamma(n) = \frac{1}{1 + \lambda^{-1}\mathbf{u}^H(n)\Phi^{-1}(n-1)\mathbf{u}(n)}$$

Hence, for $n = 1$ we have

$$e(1) = \gamma(1)\xi(1)$$

where

$$\begin{aligned}\gamma(1) =& \frac{1}{1 + \lambda^{-1}\mathbf{u}^H(1)\Phi^{-1}(0)\mathbf{u}(1)} \\ =& \frac{1}{1 + \lambda^{-1}\delta^{-1}(0)\mathbf{u}^H(1)\mathbf{u}(1)}, \qquad \Phi^{-1}(0) = \delta^{-1}\mathbf{I} \\ =& \frac{\lambda\delta}{\lambda\delta + |u(1)|^2}, \qquad \mathbf{u}(1) = \begin{bmatrix} u(1) \\ \mathbf{0} \end{bmatrix} \\ \approx& \frac{\lambda\delta}{|u(1)|^2} \qquad \delta << 1\end{aligned}$$

Hence, $\gamma(1)$ is small.

Next, we observe that

$$\xi(n) = d(n) - \hat{\mathbf{w}}^H(n-1)\mathbf{u}(n)$$

For $n = 1$:

$$\begin{aligned}\xi(1) =& d(1)\hat{\mathbf{w}}^H(0)\mathbf{u}(1)\\ =& d(1), \qquad\qquad \hat{\mathbf{w}}(0) = 0\end{aligned}$$

We thus conclude that $e(1)$ equals a very small quantity namely $\gamma(1)$, multipled by $d(1)$, which, in turn, makes $e(1)$ small.

As n increases, $\gamma(n)$ approaches towards the final value of 1. Correspondingly, the mean-square value of the estimation error $e(n)$ increases towards its steady-state value, J_{min}.

Problem 10.7

a)

$$\mathbb{E}\left[\hat{\mathbf{w}}(n)\right] = \mathbf{w}_0 - \frac{\delta}{n}\mathbf{p}$$

Therefore,

$$\mathbb{E}\left[\varepsilon(n)\right] = \frac{\delta}{n}\mathbf{p}$$

$$\mathbb{E}\left[\varepsilon(n-1)\right] = \frac{\delta}{n-1}\mathbf{p}$$

Hence,

$$\begin{aligned}\mathbb{E}\left[\varepsilon(n)\right] - \mathbb{E}\left[\varepsilon(n-1)\right] =& \frac{-\delta}{n(n-1)}\mathbf{p}\\ \approx& -\frac{1}{n}\mathbb{E}\left[\varepsilon(n-1)\right]\end{aligned}$$

Rearranging terms:

$$\mathbb{E}\left[\varepsilon(n)\right] \approx \left(1 - \frac{1}{n}\right)\mathbb{E}\left[\varepsilon(n-1)\right] \tag{1}$$

b)

Compared to the self-orthogonalizing adaptive filter discussed in Section 8.4 of the textbook, we can see from Equation (8.36) that

$$\mathbb{E}\left[\varepsilon(n)\right] \approx (1-\alpha)\,\mathbb{E}\left[\varepsilon(n-1)\right] \tag{2}$$

where the constant α lies in the range $0 < \alpha < 1$. For $\alpha = 1/n$, Equations (1) and (2) are the same. This is not surprising because in the RLS filter, the step-size μ is the inverse correlation matrix $\Phi^{-1}(n)$ whereas in self-orthogonalizing filter, step-size is $\alpha\mathbf{R}^{-1}$ where $\mathbf{R}$ is the correlation matrix.

Problem 10.8

a)

From Appendix H, using the property of complex Wishart distribution, we have: For any given nonzero α in $\Re^M$, we know that $\dfrac{\alpha^H\mathbf{R}^{-1}\alpha}{\alpha^H\Phi^{-1}(n)\alpha}$ is χ^2 (chi-squared) distributed with $n - M + 1$ degrees of freedom. Hence,

$$\begin{aligned}\mathbb{E}\left[\alpha^H\Phi^{-1}(n)\alpha\right] =& \alpha\mathbf{R}^{-1}\alpha\mathbb{E}\left[\frac{1}{\chi^2\left(n-M+1\right)}\right] \\ =& \frac{1}{n-M+1}\alpha^H\mathbf{R}^{-1}\alpha \qquad\qquad (n > M+1)\end{aligned}$$

which further implies that

$$\mathbb{E}\left[\Phi^{-1}(n)\right] = \frac{1}{n-M+1}\mathbf{R}^{-1}$$

Usually $n >> M - 1$. Hence,

$$\mathbb{E}\left[\Phi^{-1}(n)\right] \approx \frac{1}{n}\mathbf{R}^{-1}$$

b)

By definition $\mathscr{D}(n) = \mathbb{E}\left[\varepsilon^H(n)\varepsilon(n)\right]$, and $\mathbf{K}(n) = \mathbb{E}\left[\varepsilon(n)\varepsilon^H(n)\right]$; thus

$$\mathscr{D}(n) = \text{tr}\left[\mathbf{K}(n)\right]$$

From Equation (10.59) in the textbook, we may obtain

$$\begin{aligned}\mathbf{K}(n) =& \sigma_0^2\mathbb{E}\left[\Phi^{-1}(n)\right] \\ =& \frac{1}{n}\sigma_0^2\mathbf{R}^{-1}\end{aligned}$$

Hence,

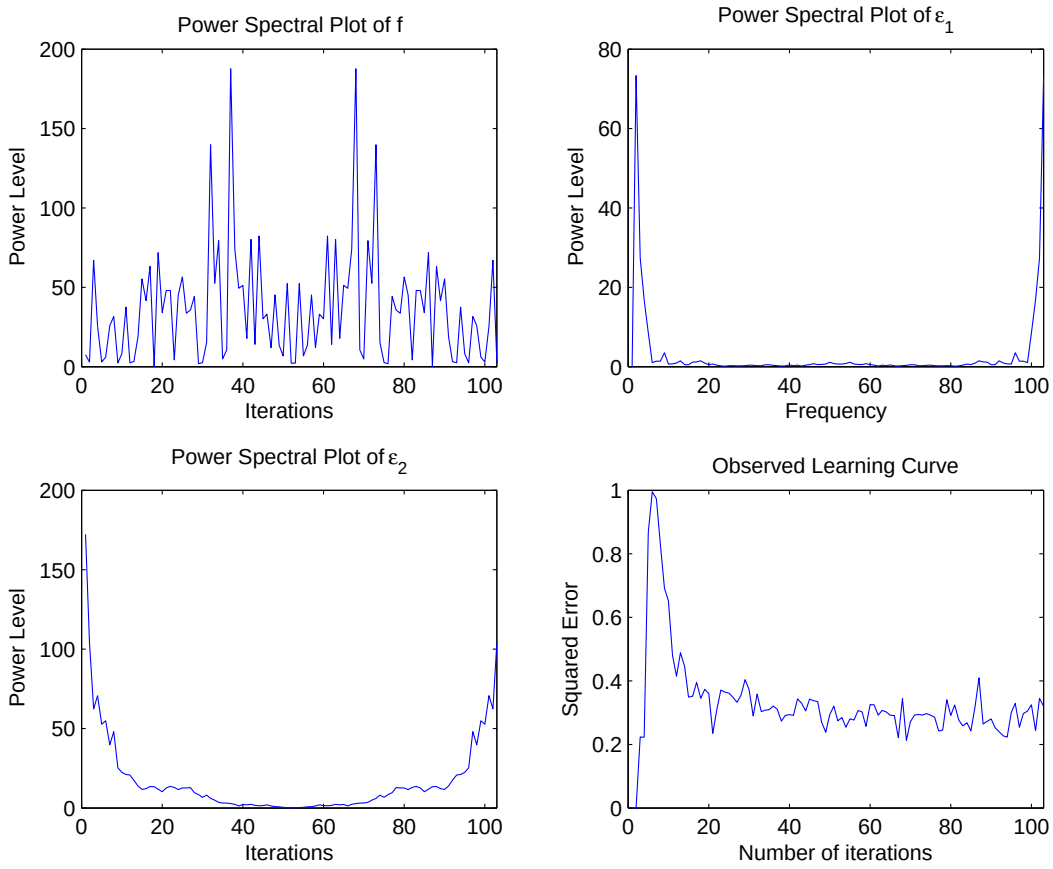
$$\begin{aligned}\mathscr{D}(n) =& \frac{1}{n}\sigma_0^2 \text{tr}\left[\mathbf{R}^{-1}\right] \\ =& \frac{1}{n}\sigma_0^2 \sum_{i=1}^{M} \frac{1}{\lambda_i}\end{aligned}$$

Problem 10.9

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Adaptive Filter Theory 5e Solution Manual
%
%
%
% Chapter 10
%
% Question 9
%
% Parts c), d) and e)
%
%
%
% Program written to run on MATLAB 2010a (R)
%
%
%
% By Kelvin Hall
%
% June 30, 2014
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear
clc
numberOfDatapoints=100; % try 300 to get a better picture of the process
numberOfRuns=100;   % The number of monteCarlo runs necissary for part d)
aOne=0.1;           %AR paramter
aTwo=-0.8;          %AR paramter as per textbook description

NoiseVariance=0.28;% The Noise variance found through part A of the problem
NoiseStandardDeviation=sqrt(NoiseVariance);
```

```
lambda=0.99;
delta=20;

stream = RandStream('mt19937ar','Seed',30);  % seed the random number
RandStream.setDefaultStream(stream);         % generator for reproducable
                                             % results

u=zeros(numberOfDatapoints+3,1);    % Allocate memory for input data stream
error=zeros(numberOfDatapoints+3,1);    % Allocate memory for error between
                                    % prediction and resutls
epsilonOne=zeros(numberOfDatapoints+3,1);% Allocate memory for error
                                   % between found weight and AR paramter 1
epsilonTwo=zeros(numberOfDatapoints+3,1);% Allocate memory for error
                                   % between found weight and AR paramter 2
g=zeros(numberOfDatapoints+3,1);% allocate memory for squared-averaged error
J=zeros(numberOfDatapoints+3,1);% allocate memory for theoretical error

weights=zeros(2,numberOfDatapoints+3); % allocate memory for weights

for k=1:numberOfRuns % increment through the monte carlo runs of the problem
    P=eye(2)*delta;
    W=zeros(2,numberOfDatapoints+3);    % reset the weights to zero between
                                        % each montecarlo run
   for n=3:numberOfDatapoints+3
        u(n)=aOne*u(n-1)+aTwo*u(n-2)+randn(1)*NoiseStandardDeviation;
             % create a new piece of input data to the filter
        U=[u(n-1);u(n-2)];

        epsilonOne(n)=aOne-W(1,n-1);
             % calculate error between weight one and AR paramter one

        epsilonTwo(n)=aTwo-W(2,n-1);
             % calculate error between weight two and AR paramter two

        kappa=lambda^(-1)*P*U/(1+lambda^(-1)*U'*P*U);%update kappa as per
                                   % RLS algorithm

        error(n)=u(n)-W(1,n-1)*u(n-1)-W(2,n-1)*u(n-2);% find error between
                                   % predicted value and measured value

        W(:,n)=W(:,n-1)+kappa*error(n); % updtate weights

        P=lambda^(-1)*P-lambda^(-1)*kappa*U'*P;
                    %update the weights as per RLS
    end
    g=g+error.^2;      % accumulate squared error of estimation
```

```
end
g=g/numberOfRuns;  % accumulate squared error of estimation


index=1:numberOfDatapoints+3; %the x axis of the plot


subplot(2,2,1) % first subplot showing the PSD of the error f, where it can
               % be seen that the error is similar to that of white noise
               % with approximatly even values across the frequency range
plot(index,(abs((fft(error)))).^2)
xlim([0 numberOfDatapoints+3])
title('Power Spectral Plot of f')
xlabel('Iterations') % x-axis label
ylabel('Power Level') % y-axis label

subplot(2,2,2) % second subplot showing the power spectral density of Error
               % epsilon 1, as there is a significant offset error the
               % frequency representation is highly noneven, with more
               % being towards the lower frequency
plot(index,(abs((fft(epsilonOne)))).^2)
xlim([0 numberOfDatapoints+3])
title('Power Spectral Plot of \epsilon_1')
xlabel('Frequency') % x-axis label
ylabel('Power Level') % y-axis label

subplot(2,2,3) % third subplot showing the power spectral density of Error
               % epsilon 2, as there is a significant offset error the
               % frequency representation is highly noneven, with more
               % being towards the lower frequency
plot(index,(abs((fft(epsilonTwo)))).^2)
xlim([0 numberOfDatapoints+3])
title('Power Spectral Plot of \epsilon_2')
xlabel('Iterations') % x-axis label
ylabel('Power Level') % y-axis label

subplot(2,2,4) % Solution to part d) and e) on the same graph for
               % comparison purposes.
plot(index,g(1:numberOfDatapoints+3))
xlim([0 numberOfDatapoints+3])
ylim([0 1])
xlabel('Number of iterations') % x-axis label
ylabel('Squared Error') % y-axis label
title({'Observed Learning Curve'})
```

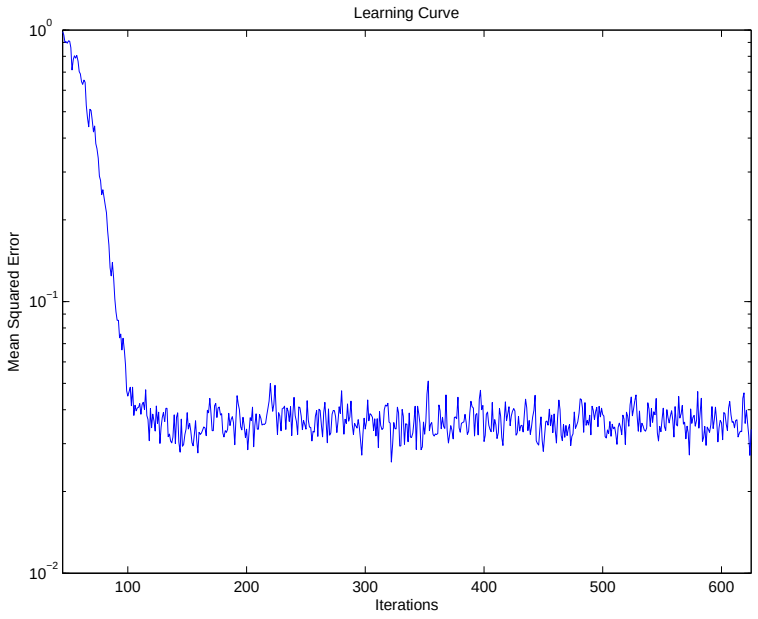


Problem 10.10

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Adaptive Filter Theory 5e Solution Manual                                      %
%                                                                               %
% Chapter 10                                                                    %
% Question 10                                                                   %
%                                                                               %
%                                                                               %
% Program written to run on MATLAB 2010a (R)                                    %
%                                                                               %
% By Kelvin Hall                                                                %
% June 30, 2014                                                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
clear
lambda=0.9;   % forgetting factor
delta=0.005;  % regularization paramter
H1=[0.25,1,0.25];   %channel response one
H2=[0.25,1,-0.25];  %channel response two
H3=[-0.25,1,0.25];  %channel response three
standardDeviationNoise=sqrt(0.01);  %addivite white noise standard deviation
delay=11;       % optimal delay found in earlier question (ch6 question 18)

numberOfDatapoints=600;     % number of data points per run
NumberOfRuns=200;       % number of monte carlo runs that the results
                        %are averaged over
u=zeros(numberOfDatapoints+delay+1+21,1);  % allocate memory for filter input
r=zeros(1,numberOfDatapoints+delay+1+21);  % allocate memory for filter output
error=zeros(numberOfDatapoints+delay+1+21,1);  %allocate memory for errors
E=zeros(numberOfDatapoints+delay+1+21,1); % allocate memory for MSE

for k=1:NumberOfRuns
    P=eye(21)*delta; %reset filter variable between monte carlo runs
    W=zeros(21,1);
    for n=delay+1+21:numberOfDatapoints+delay+1+21
        u(n)=rand(1); % create new input data
        if u(n)>0.5
            u(n)=1;
        else
            u(n)=-1;
        end
        d=u(n-delay);
        r(n)=H1*[u(n);u(n-1);u(n-2)]+randn*standardDeviationNoise;

        U=r(n:-1:n-20)'; % create a vector of the 21 most recent filter outputs

        error(n)=u(n-delay)-U'*W;

        kappa=lambda^(-1)*P*U/(1+lambda^(-1)*U'*P*U); % update kappa as perRLS

        W=W+kappa*error(n); % update weights

        P=lambda^(-1)*P-lambda^(-1)*kappa*U'*P; %update as per RLS
    end
    E=E+error.^2; % accumulate squared error in prediction
end
E=E/NumberOfRuns; % divide the squared error by the number of
                % monte carlo runs to get the Mean Squared Error

semilogy(E)
```

```
ylim([0.01 1])
xlim([45 625])
title('Learning Curve')
xlabel('Iterations')
ylabel('Mean Squared Error')
```

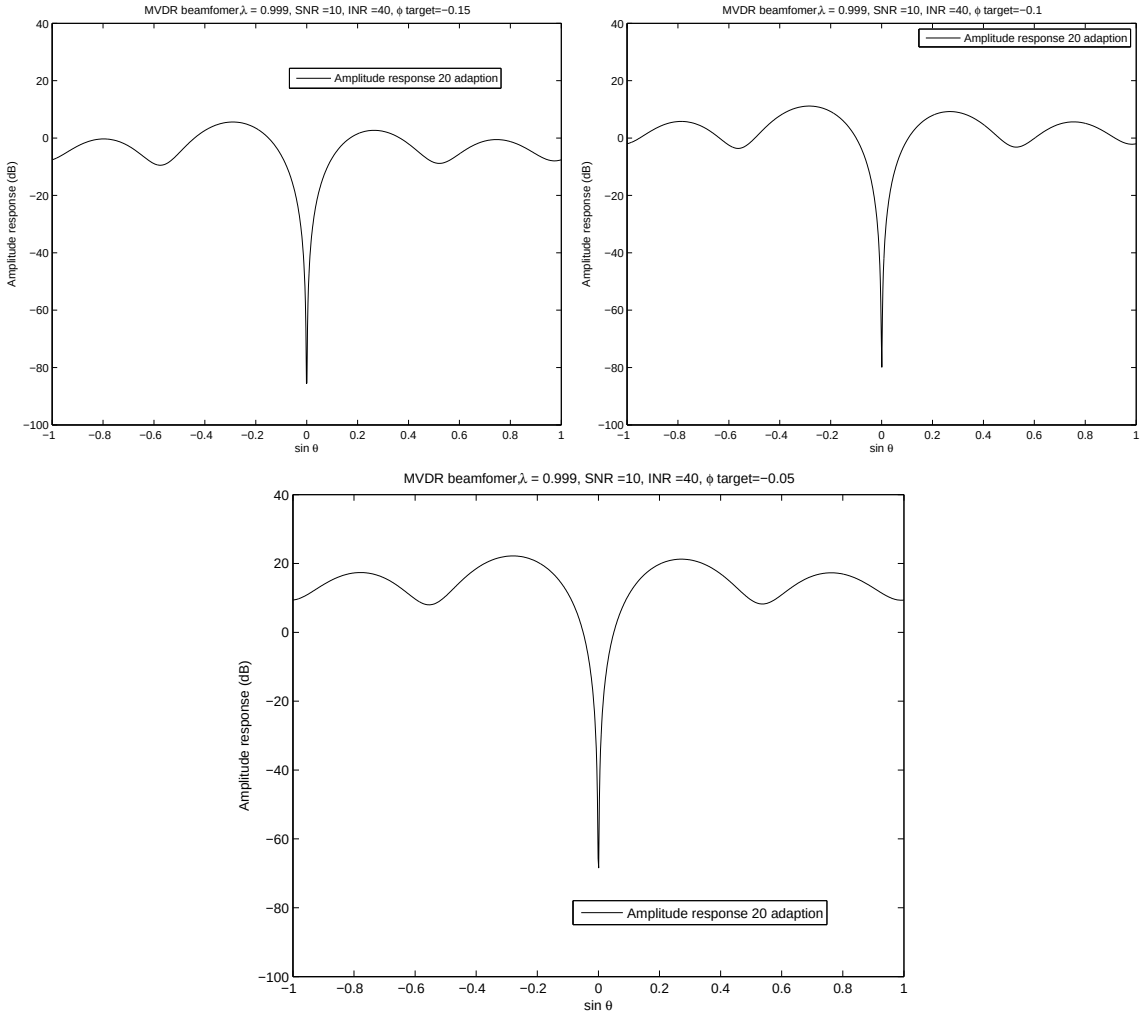


Problem 10.11

```
clear all;
close all;
clc;
p = 5; % number of sensors
Ninit = p; % nuber of smaples needed for initialization
Nsnaps = [10, 10, 20]; % number of snapshots or iteration
mean_v = 0; % white noise mean
var_v = 1; % white noise variance
lambda = 0.999; % forgetting factor
SNRdB = 10; % target signal to noise ratio
INRdB = 40; % interference signal to noise ratio
numst = 1000; % resolution in spatial response
sin_theta = [-0.05 0]; % location of signal and interference
phi = pi.*sin_theta; % equivalent electrical angle
```

```
A = sqrt(var_v)*10.^([SNRdB INRdB]./20); % parameter for target/interference signal amp
% steering vector along electrical angle of look direction of interest
e = exp(-1j*[1:(p-1)]'*phi(1));
% setup input and output sequences

for n=1:3
    Ndata = Ninit + Nsnaps(n); % total number of data
    sig_x = A(1)*exp(1j*[1:p]*phi(1));
    for i = 1:Ndata,
    % random disturbances
        v_tmp = sqrt(var_v/2)*randn(2,p)+mean_v;
        v = v_tmp(1, :)+ 1j*v_tmp(2, :); % additive white noise
        Psi = 2*pi*rand; % uniform random phase on interference
        Xi(i, :) = sig_x + A(2)*exp(1j*[1:p]*phi(2) + Psi) + v;
    end;
    g = 1; % unity gain
    d = g*Xi(:,1);
    u = diag(Xi(:,1))*(ones(Ndata,1)* e.')-Xi(:,2:p);

    [W,xp] = rls(u,d,lambda);
    Wo = g- W * conj(e);
    W = [Wo W];

    % now, generate test vectors to compute spatially sampled response
    W_H = conj(W(Ndata, :)); % Hemitian transpose of last one
    st = linspace(-1,1,numst); % sine(theta) space
    est = exp(-1j*pi*[0:(p-1)]'*st); % steering matrix
    % amplitude response
    P(:,n) = 20*log10(abs(W_H*est).^2);
end

plot(st,P(:,3),'k');
%axis([-1 1 min(P) max(P)]);
axis([-1 1 -100 40]);
xlabel('sin \theta');
ylabel('Amplitude response (dB)');
legend('Amplitude response 20 adaption');
title(['MVDR beamfomer,','\lambda = ',num2str(lambda),', SNR =',num2str(SNRdB),', INR =


function [ W,xp ] = rls( u,d,lambda )
N = min(size(u, 1),size(d, 1));
Nin = size(u,2);
Nout =size(d,2);
w=zeros(Nout,Nin);
W=[];
```

```
delta=0.1;
P=eye(Nin)*delta; %reset filter variable between monte carlo runs
for n=1:N
    W=[W;w];
    xp(n,:)=u(n,:)*w';
    e(n,:)=d(n,:)-xp(n,:);
    kappa=lambda^(-1)*P*u(n,:)'/(1+lambda^(-1)*u(n,:)*P*u(n,:)'); % update kappa as perR

    w=w+kappa'*e(n); % update weights

    P=lambda^(-1)*P-lambda^(-1)*u(n,:)*kappa*P; %update as per R

end
```



Chapter 11

Problem 11.1

Starting with the formula for the LMS algorithm (for real-data):

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu \mathbf{u} e(n)$$

where

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

$$d(n) = \mathbf{w}^T\mathbf{u}(n) + \nu(n)$$

and

$$\tilde{\mathbf{w}}(n) = \mathbf{w} - \hat{\mathbf{w}}(n)$$

a) *

Accordingly, we proceed by first writing:

$$\begin{aligned}
\mu^{-\frac{1}{2}}\tilde{\mathbf{w}}(n+1) =& \mu^{-\frac{1}{2}}\left(\mathbf{w} - \hat{\mathbf{w}}(n+1)\right) \\
=& \mu^{-\frac{1}{2}}\left(\mathbf{w} - (\hat{\mathbf{w}}(n) + \mu\mathbf{u}(n)e(n))\right) \\
=& \mu^{-\frac{1}{2}}\left(\tilde{\mathbf{w}}(n) - \mu\mathbf{u}(n)\left(d(n) - \hat{\mathbf{w}}^T(n)\mathbf{u}(n)\right)\right) \\
=& \mu^{-\frac{1}{2}}\tilde{\mathbf{w}}(n) - \mu^{\frac{1}{2}}\mathbf{u}(n)\left(d(n) - \hat{\mathbf{w}}^T(n)\mathbf{u}(n)\right) \qquad (1)
\end{aligned}$$

*Correction: The question was meant to ask to show that $\mu^{-\frac{1}{2}}\mathbf{w}(\tilde{\mathbf{n}+1})$ equals $\mu^{-\frac{1}{2}}\tilde{\mathbf{w}}(n) - \mu^{\frac{1}{2}}\mathbf{u}(n)\left(d(n) - \hat{\mathbf{w}}^T(n)\mathbf{u}(n)\right)$ not $\mu^{-\frac{1}{2}}\tilde{\mathbf{w}}(n) - \mu^{-\frac{1}{2}}\mathbf{u}(n)\left(d(n) - \hat{\mathbf{w}}^T(n)\mathbf{u}(n)\right)$

b)

The additive noise $\nu(n)$ is defined by

$$\begin{aligned}\nu(n) =& d(n) - \mathbf{w}^T\mathbf{u}(n)\\ =& d(n) - \left(\hat{\mathbf{w}}(n) + \tilde{\mathbf{w}}^T\mathbf{u}(n)\right)\\ =& \left(d(n) - \hat{\mathbf{w}}^T(n)\mathbf{u}(n)\right) - \tilde{\mathbf{w}}^T\mathbf{u}(n)\end{aligned} \tag{2}$$

c)[†]

Next, introduce the algebraic squared Euclidean difference:

$$\mu^{-1}\left\|\tilde{\mathbf{w}}(n+1)\right\|^2 - \nu^2(n) = \left(\mu^{-1/2}\right)^2\tilde{\mathbf{w}}^T(n+1)\tilde{\mathbf{w}}(n+1) - \nu(n)\nu(n)$$

Then, using (1)and (2) defined in parts **a)** and **b)**, write

$$\begin{aligned}\mu^{-1}\left\|\tilde{\mathbf{w}}(n+1)\right\|^2 - \nu^2(n) = \{&\mu^{-1}\left\|\tilde{\mathbf{w}}(n)\right\|^2 + \mu\left\|\mathbf{u}(n)\right\|^2(d(n)\\ &-\hat{\mathbf{w}}^T(n)\mathbf{u}(n)\big)^2 - 2\tilde{\mathbf{w}}^T(n)\mathbf{u}(n)\,(d(n)\\ &-\hat{\mathbf{w}}^T(n)\mathbf{u}(n)\big)\}\\ -\{&\left(d(n) - \hat{\mathbf{w}}^T(n)\mathbf{u}(n)\right)^2 + \left(\tilde{\mathbf{w}}^T(n)\mathbf{u}(n)\right)^2\\ &-2\tilde{\mathbf{w}}^T\mathbf{u}(n)\left(d(n) - \hat{\mathbf{w}}^T(n)\mathbf{u}(n)\right)\}\end{aligned}$$

Hence, simplifying terms, we get

$$\mu^{-1}\left\|\tilde{\mathbf{w}}(n+1)\right\| - \nu^2 = \mu^{-1}\left\|\tilde{\mathbf{w}}(n)\right\|^2 - \left(\tilde{\mathbf{w}}^T(n)\mathbf{u}(n)\right)^2 - \left(1 - \mu\left\|\mathbf{u}(n)\right\|^2\right)\left(d(n) - \hat{\mathbf{w}}^T(n)\mathbf{u}(n)\right)^2 \tag{3}$$

[†]Correction: The last bracketed expression on the right hand side of the question should read $\left(d(n) - \hat{\mathbf{w}}^T(n)\mathbf{u}(n)\right)^2$ not $\left(d(n) - \tilde{\mathbf{w}}^T(n)\mathbf{u}(n)\right)^2$

d)[‡]

Adding up (3) from adaptation cycle $n = 0$ to $n = N$, we first note the following

$$\sum_{n=0}^{N} \|\tilde{\mathbf{w}}(n+1)\|^2 - \sum_{n=0}^{N} \|\tilde{\mathbf{w}}(n)\|^2 = \|\tilde{\mathbf{w}}(N)\|^2 - \|\tilde{\mathbf{w}}(0)\|^2$$

Accordingly, we may express the sum of adaptation cycles as follows:

$$\begin{aligned}\mu^{-1} \|\tilde{\mathbf{w}}(N)\|^2 - \sum_{n=0}^{N} \nu^2(n) =& \mu^{-1} \|\tilde{\mathbf{w}}(0)\|^2 - \sum_{n=0}^{N} (\tilde{\mathbf{w}}(n)\mathbf{u}(n)) \\ & - \sum_{n=0}^{N} \left(1 - \mu \|\mathbf{u}(n)\|^2\right) \left(d(n) - \hat{\mathbf{w}}^T(n)\mathbf{u}(n)\right) \end{aligned} \tag{4}$$

e)

Reformatting (4) in the form of a rational function, we write

$$\frac{\mu^{-1} \|\tilde{\mathbf{w}}(N)\|^2 + \sum_{n=0}^{N} \left(\tilde{\mathbf{w}}^T(n)\mathbf{u}(n)\right)^2 + \sum_{n=0}^{N} \left(1 - \mu \|\mathbf{u}(n)\|^2\right) \left(d(n) - \hat{\mathbf{w}}^T(n)\mathbf{u}(n)\right)^2}{\mu^{-1} \|\tilde{\mathbf{w}}(0)\|^2 + \sum_{n=0}^{N} \nu^2(n)} = 1 \tag{5}$$

Equation (5) sets the stage for the later part of the problem.

f)

In so far as the desired requirement is concerned, the denominator stands as it is. The attention would therefore have to focus on the numerator in Equation (5). To this end, we now make the following two observations:

i) For a prescribed N, the term in the numerator that involves the step-size parameter, $\mu^{-1} \|\tilde{\mathbf{w}}(N)\|^2$ is a constant that is under scrutiny.

ii) The third term in the numerator may also be ignored simply because this term merely involves estimation of the unknown weight vector with no measure of comparison.

[‡]Correction: The last bracketed expression on the right hand side of the question should read $\left(d(n) - \hat{\mathbf{w}}^T(n)\mathbf{u}(n)\right)^2$ not $\left(d(n) - \tilde{\mathbf{w}}^T(n)\mathbf{u}(n)\right)^2$

Accordingly, it follows that insofar as robustness is concerned, it is the second term in the numerator of Equation (5) that requires attention for the following reasons.

In physical terms, the second term is the only prediction-error term, namely,

$$\sum_{n=0}^{N} \left(\tilde{\mathbf{w}}^T(n)\mathbf{u}(n)\right)^2 = \sum_{n=0}^{N} \left[\left(\mathbf{w} - \hat{\mathbf{w}}^T(n)\right)\mathbf{u}(n)\right]^2,$$

which, importantly involves all the vectors that matter except fo the step-size parameter μ.

Finally referring back to (11.6) in the textbook, by definition, we have

$$|\xi_u(n)|^2 \leq \left(\tilde{\mathbf{w}}^T(n)\mathbf{u}(n)\right)^2,$$

it follows that we may simplify (5) by writing

$$\frac{\sum_{n=0}^{N} |\xi_u(n)|^2}{\mu^{-1}\|\tilde{\mathbf{w}}(0)\|^2 + \sum_{n=0}^{N} \nu^2(n)} \leq 1 \tag{6}$$

which is the desired requirement.

Problem 11.2

[§] The starting point of the second problem is defined by the following inequality:

$$\frac{\sum_{n=0}^{i} \left(\hat{d}(n) - \mathbf{w}^T\mathbf{u}(n)\right)}{\mu^{-1}\mathbf{w}^T\mathbf{w} + \sum_{n=0}^{i-1} \left(d(n) - \mathbf{w}^T\mathbf{u}(n)\right)^2} \leq 1 \tag{1}$$

where i refers to an arbitrary instant of time; the weight vector $\mathbf{w}$ is unknown.

[§]Correction: the summing term in the denominator of the question description was suppose to be over the term $\left(d(n) - \mathbf{w}^T\mathbf{u}(n)\right)^2$ not $\nu^2(n)$

Re arranging terms in (1), we may write

$$\mu^{-1}\mathbf{w}^T\mathbf{w} + \sum_{n=0}^{i-1}\left(d(n) - \mathbf{w}^T\mathbf{u}(n)\right)^2 - \sum_{n=0}^{i}\left(\hat{d}(n) - \mathbf{w}^T\mathbf{u}(n)\right)^2 \geq 0 \tag{2}$$

The expression on the left-hand side of (2) is an indefinite quadratic function in the unknown weight vector $\mathbf{w}$ on account of the negative term involving estimate of the desired response.

The quadratic term on the left-hand side of Equation (2) will be of a positive form if, and only if, it has a minimum value of the unknown $\mathbf{w}$. To check that this quadratic term has a minimum, we may differentiate it twice:

i) On differentiating the quadratic term for the first time, we write (following Wirtinger rule of calculus):

$$\begin{aligned}
&\frac{\partial}{\partial \mathbf{w}^T}\left[\mu^{-1}\mathbf{w}^T\mathbf{w} + \sum_{n=0}^{i-1}\left(d(n) - \mathbf{w}^T\mathbf{u}(n)\right)^2 - \sum_{n=0}^{i}\left(\hat{d}(n) - \mathbf{w}^T\mathbf{u}(n)\right)^2\right] \\
&\quad = 2\mu^{-1}\mathbf{w} + \sum_{n=0}^{i-1} 2\left(d(n) - \mathbf{w}^T\mathbf{u}(n)\right)(-\mathbf{u}(n)) - \sum_{n=0}^{i} 2\left(\hat{d}(n) - \mathbf{w}^T\mathbf{u}(n)\right)(-\mathbf{u}(n)) \\
&\quad = 2\mu^{-1}\mathbf{w} - 2\sum_{n=0}^{i-1}\left(d(n) - \mathbf{w}^T\mathbf{u}(n)\right)\mathbf{u}(n) + 2\sum_{n=0}^{i}\left(\hat{d}(n) - \mathbf{w}^T\mathbf{u}(n)\right)\mathbf{u}(n)
\end{aligned} \tag{3}$$

ii) For differentiating the quadratic term a second time, we first rewrite (3) in the equivalent form, recognizing that $\mathbf{w}^T\mathbf{u}(n) = \mathbf{u}(n)^T\mathbf{w}$; thus:

$$2\mu^{-1}\mathbf{w} - 2\sum_{n=0}^{i-1}\mathbf{u}(n)\left(d(n) - \mathbf{u}^T(n)\mathbf{w}(n)\right) + 2\sum_{n=0}^{i}\mathbf{u}(n)\left(\hat{d}(n) - \mathbf{u}^T(n)\mathbf{w}\right) \tag{4}$$

This time, we follow the alternative form of the Wirtinger rule, and differentiate (4) with respect to $\mathbf{w}$, obtaining:

$$\begin{aligned}
&\frac{\partial}{\partial \mathbf{w}}\left[2\mu^{-1}\mathbf{w} - 2\sum_{n=0}^{i-1}\mathbf{u}(n)\left(d(n) - \mathbf{u}^T(n)\mathbf{w}\right) + 2\sum_{n=0}^{i}\mathbf{u}(n)\left(\hat{d}(n) - \mathbf{u}^T(n)\mathbf{w}\right)\right] \\
&\quad = 2\mu^{-1}\mathbf{I} + 2\sum_{n=0}^{i-1}\mathbf{u}(n)\mathbf{u}^T(n) - 2\sum_{n=0}^{i}\mathbf{u}(n)\mathbf{u}^T(n) \\
&\quad = 2\mu^{-1}\mathbf{I} + 2\mathbf{u}(i)\mathbf{u}^T(i)
\end{aligned} \tag{5}$$

where, in the second step of the differentiation we introduced the identity matrix $\mathbf{I}$ to maintain correct dimensionality of the resultant.

On account of the fact that the Jacobian, represented by the indefinite quadratic, namely the outer product $\mathbf{u}(i)\mathbf{u}^T(i)$ has a single non-unity eigenvalue, we conclude by stating the required condition:

$$0 < \mu \leq \frac{1}{\mathbf{u}^T(i)\mathbf{u}(i)} \tag{6}$$

Problem 11.3

There are two important lessons that we have learned from the solution to problems 11.1 and 11.2;

i) First of all, the condition which the step-size parameter has to satisfy in (6) of the solution to Problem 11.2 is previously the same condition described in the formula after (11.6) in section 11.3 of the textbook.

ii) Secondly, the expression on the left-hand side of (6) in the solution to Problem 11.1 is precisely the expression derived previously in (11.12) of Section 11.3 of the textbook. To be more precise, this expression defines the H^∞ norm (i.e., maximum energy gain) of the LMS algorithm with step-size parameter μ.

Problem 11.4

¶ Reproducing the quadratic form on the left-hand side of (2) in the solution to Problem 11.2, for convenience of presentation, we write

$$Q(\mathbf{w}) = \mu^{-1}\mathbf{w}^T\mathbf{w} + \sum_{n=0}^{i-1}\big(d(n) - \mathbf{w}^T\mathbf{u}(n)\big)^2 - \sum_{n=0}^{i}\Big(\hat{d}(n) - \mathbf{w}^T\mathbf{u}(n)\Big)^2 \tag{1}$$

where we have introduced the function $Q(\mathbf{w})$ as the algebraic description of the quadratic form expressed as a function of the unknown weight vector $\mathbf{w}$. Note also that given the

¶Correction: The question should be asking to find the optimizing $\mathbf{w}$ as shown by

$$\mathbf{w} = \big[\mu^{-1}\mathbf{I} - \mathbf{u}(i)\mathbf{u}^T(i)\big]^{-1}\left(\sum_{n=0}^{i-1} e(n)\mathbf{u}(n) - \hat{d}(i)\mathbf{u}(i)\right), \text{ not}$$

$$\mathbf{w} = \big[\mu\mathbf{I} - \mathbf{u}(i)\mathbf{u}^T(i)\big]^{-1}\left(\sum_{n=0}^{i-1} e(n)\mathbf{u}(n) - \hat{d}(i)\mathbf{u}(i)\right)$$

desired response $d(n)$ at iteration $i-1$, an estimate of the desired response is computed on the next iteration i, hence: The first summation involving $d(n)$ extends up to iteration $i-1$ from zero, where the second summation involving $\hat{d}(n)$ extends up to iteration i again from zero.

Differentiating the function $Q(\mathbf{w})$ in (1) with respect to the unknown weight vector following the Wirtiger rule, we write

$$\begin{aligned}\frac{\partial Q(\mathbf{w})}{\partial \mathbf{w}^T} =& 2\mu^{-1}\mathbf{w} - 2\sum_{n=0}^{i-1}\left(d(n) - \mathbf{w}^T\mathbf{u}(n)\right)\mathbf{u}(n) + 2\sum_{n=0}^{i}\left(\hat{d}(n) - \mathbf{w}^T\mathbf{u}(n)\right)\mathbf{u}(n)\\ =& 2\mu^{-1}\mathbf{w} - 2\left(\mathbf{w}^T\mathbf{u}(i)\right)\mathbf{u}(i) - 2\sum_{n=0}^{i-1}\left(d(n) - \mathbf{w}^T\mathbf{u}(n)\right)\mathbf{u}(n)\\ &+ 2\sum_{n=0}^{i-1}\left(\hat{d}(n) - \hat{\mathbf{w}}^T\mathbf{u}(n)\right)\mathbf{u}(n) + 2\hat{d}(i)\mathbf{u}(i)\\ =& 2\mu^{-1}\mathbf{w} - 2\left(\mathbf{w}^T\mathbf{u}(i)\right)\mathbf{u}(i) + 2\sum_{n=0}^{i-1}\left(\hat{d}(n) - d(n)\right)\mathbf{u}(n) + 2\hat{d}(i)\mathbf{u}(i) \quad (2)\end{aligned}$$

By definition, we have

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

and

$$\hat{d}(n) = \hat{\mathbf{w}}^T(n)\mathbf{u}(n)$$

from the combined use of which it follows that

$$\begin{aligned}\hat{d}(n) - d(n) =& \hat{\mathbf{w}}^T\mathbf{u}(n) - \left(e(n) + \hat{\mathbf{w}}^T\mathbf{u}(n)\right)\\ =& -e(n) \quad (3)\end{aligned}$$

Next, we note that

$$\mathbf{w}^T\mathbf{u}(n) = \mathbf{u}^T(n)\mathbf{w} \quad (4)$$

Using (3) and (4) in (2), we obtain

$$\frac{\partial Q(\mathbf{w})}{\partial \mathbf{w}} = 2\left[\mu^{-1}\mathbf{I} - \left(\mathbf{u}(i)\mathbf{u}^T(i)\right)\right]\mathbf{w} - 2\sum_{n=0}^{i-1}e(n)\mathbf{u}(n) + 2\hat{d}(i)\mathbf{u}(i) \quad (5)$$

where we have also introduced the identity matrix $\mathbf{I}$ for consistency of matrix dimensionality.

Finally, setting the partial differential $\partial Q(\mathbf{w})/\partial \mathbf{w}$ equal to zero, and solving for the unknown weight vector $\mathbf{w}$, we get the desired unknown vector $\mathbf{w}$ as follows (after ignoring the common factor of 2):

$$\mathbf{w} = \left[\mu^{-1}\mathbf{I} - \mathbf{u}(i)\mathbf{u}^T(i)\right]^{-1}\left(\sum_{n=0}^{i-1} e(n)\mathbf{u}(n) - \hat{d}(i)\mathbf{u}(i)\right)$$

where it is recognized that the step-size parameter μ is a positive number less than unity.

Problem 11.5

a) [‖]

An important characteristic of the LMS algorithm is that it maps disturbances present in the input data into estimation errors. The essence of this statement is the underlying property that the energy of the estimation error is always less than the energy of disturbance. It is on account of this intrinsic property that the LMS algorithm could be viewed as a "contractive mapper"

b)

The claim that the LMS algorithm can be characterized as a contractive mapper is verified by the experiment conducted in section 6.7, where an adaptive predictive system, which is designed to have a process variance of $\sigma_u^2 = 0.936$, it produces a steady state error variance of approximately 0.02. On the next page, Figure 1 shows the error variance for various values of the step size parameter μ acting on the experimental setup described in section 6.7. Continue on the next page:

[‖]Correction: The experiment is described in Section 6.7 not 6.8

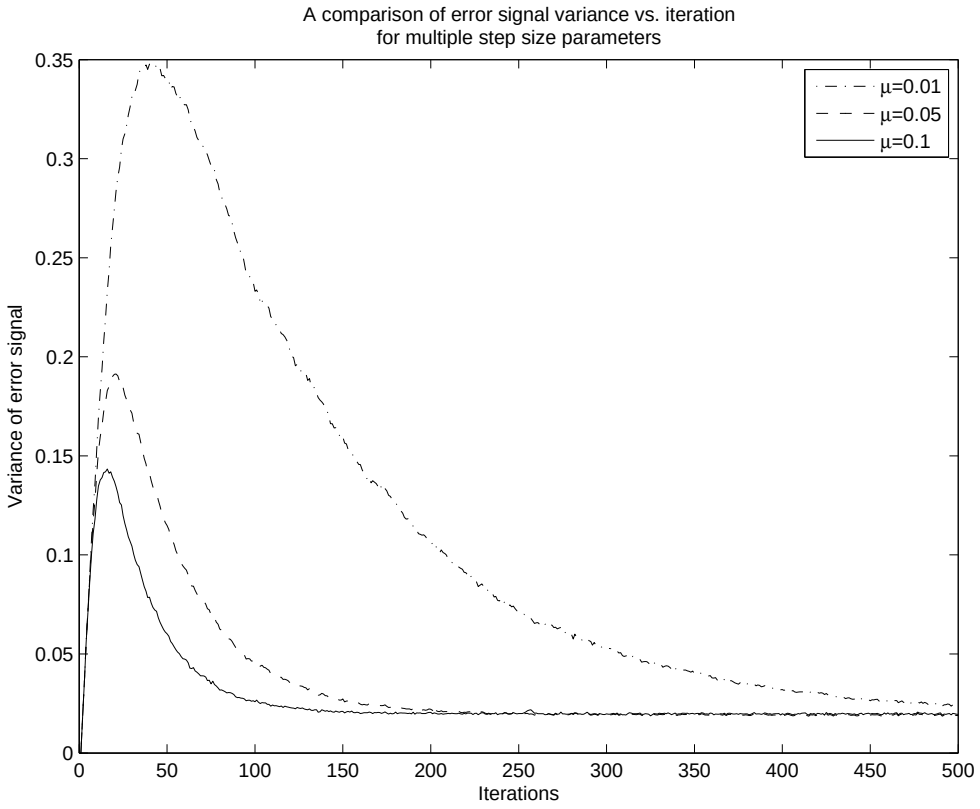


Figure 1

This figure verifies the claim by showing should be noted that the largest variance experienced by the experiment is well below 0.05, which is less than the process variance..

The code for this experimental setup can be found as a solution to Question 6.16.

Problem 11.6

The tightening of the inequality in Equation (11.9) is justified by invoking two relations:

1. The update equation for the LMS algorithm:

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu \mathbf{u}(n) e^*(n)$$

or equivalently

$$\begin{aligned}\tilde{\mathbf{w}}(n+1) =& \tilde{\mathbf{w}}(n) - \mu \mathbf{u}(n) e^*(n) \\ =& \tilde{\mathbf{w}}(n) - \mu \mathbf{u}(n) \left[d^*(n) - \mathbf{u}^H(n) \hat{\mathbf{w}}(n) \right]\end{aligned}$$

2. The multiple regression model

$$d(n) = \mathbf{w}^H \mathbf{u}(n) + \mu(n)$$

where $\mathbf{w}$ is the unknown parameter vector of the model.

Here it is assumed that the step-size parameter μ is maintained smaller than $1/\left\|\mathbf{u}(n)\right\|^2$ in accordance with Equation (11.10), in the textbook.

Problem 11.7

**

In the traditional LMS algorithm, the step-size paramter μ plays a key role. The importance note here is the fact that the dimension in which μ is measured, the dimension is the inverse of the squared Euclidean norm of the input vector $\mathbf{u}(n)$. In any event, when the issue of interest is robustness, μ is conditionally defined as follows:

$$0 < \mu < \frac{1}{\left\|\mathbf{u}(n)\right\|^2}, \qquad \text{which is dimensionally correct.} \tag{1}$$

a)

Suppose now we introduce a new step-size parameter, denoted by $\tilde{\mu}$; this new parameter is purposely made to be dimensionless, as shown by

$$\tilde{\mu} = \left\|\mathbf{u}(n)\right\|^2 \mu \tag{2}$$

Equivalently, we may write

$$\mu = \frac{\tilde{\mu}}{\left\|\mathbf{u}(n)\right\|^2} \tag{3}$$

In a corresponding way, we redefine the LMS algorithm in terms of the dimensionless step-size parameter, $\tilde{\mu}$, as follows

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{\tilde{\mu}}{\left\|\mathbf{u}(n)\right\|^2}\mathbf{u}(n)e(n) \tag{4}$$

**Correction: Tildes are missing from above the step size parameters of the Normalized LMS algorithm entries in table P 11.1

which is immediately recognized to be the normalized LMS algorithm; see (7.10) in Chapter 7 of the textbook.

To address robustness of the generalized LMS algorithm, we may build on the robustness of the conventional LMS algorithm, described in (1). In light of the relationship between μ and $\tilde{\mu}$ presented in (2), it follows that the condition for robustness of the generalized LMS algorithm is shown by

$$0 < \tilde{\mu} < 1 \tag{5}$$

b)

In this second part of the problem, the generalized LMS algorithm of Equation (4) is itself modified, as follows in light of the discussion presented in Section 7.1 of the textbook:

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{\tilde{\mu}\mathbf{u}(n)}{\varepsilon + \|\mathbf{u}(n)\|^2} e(n) \tag{6}$$

Where ϵ is a relatively small parameter with the same dimension as $\|\mathbf{u}(n)\|^2$. Following arguments similar to those that resulted in Equation (1) for the generalized LMS algorithm, we may say that for the modified algorithm described in Equation (6) to be robust, the following conditions must be satisfied:

$$0 < \frac{\tilde{\mu}}{\varepsilon + \|\mathbf{u}(n)\|^2} < 1 \tag{7}$$

To conclude the discussion presented in the solution, we say that (7) includes the conditions for robustness of the conventional LMS and generalized LMS as a special case.

Problem 11.8

Referring to Table P 11.1, we may address parts (a) and (b) of Problem 8 in this chapter, as follows respectively:

a)

The answer to the question depends on the application of interest:

If statistical performance of the LMS or modified LMS algorithm is the issue of interest, then we may respectively choose:

$$0 < \mu < \frac{2}{\lambda_{\text{max}}} \qquad \text{for the LMS algorithm}$$

or

$$0 < \tilde{\mu} < 2 \qquad \text{for the generalized LMS algorithm}$$

Note that in this scenario,

$$\tilde{\mu} < \lambda_{\max}\mu \tag{1}$$

where $\lambda_{\max}$ is the largest eigenvalue of the correlation matrix of the input vector $\mathbf{u}(n)$.

If, on the other hand, robustness of the LMS or generalized LMS algorithm is the issue of interest, then the respective choice is as follows:

$$0 < \mu < \frac{1}{\|\mathbf{u}(n)\|} \qquad \text{for the LMS algorithm}$$

or

$$0 < \tilde{\mu} < 1 \qquad \text{for the generalized LMS algorithm}$$

b)

From a practical perspective, logically speaking, the first issue of interest is to satisfy robustness of the chosen algorithm, LMS or generalized LMS.

After the problem of robustness has been satisfactorily examined then, the issue of statistical performance of the algorithm should be addressed.

Problem 11.9

Consider the difference:

$$X(n) = \epsilon^H(n)\Phi(n)\epsilon(n) - \epsilon^H(n-1)\Phi(n-1)\epsilon(n-1) \tag{1}$$

From RLS theory:

$$\epsilon(n) = \epsilon(n-1) - \Phi^{-1}(n)\mathbf{u}(n)\xi^*(n) \tag{2}$$

We may therefore rewrite Eq. (1) as

$$\begin{aligned}
X(n) = &\left[\epsilon(n-1) - \Phi^{-1}(n)\mathbf{u}(n)\xi^*(n)\right]^H \Phi(n) \left[\epsilon(n-1) - \Phi^{-1}(n)\mathbf{u}(n)\xi^*(n)\right] \\
&- \epsilon^H(n-1)\Phi(n-1)\epsilon(n-1) \\
=&\epsilon^H(n-1)\left[\Phi(n) - \Phi(n-1)\right] - \epsilon(n-1) - \xi(n)\mathbf{u}(n)\Phi^{-1}(n)\Phi(n)\epsilon(n-1) \\
&- \xi^*(n)\epsilon^H(n-1)\Phi(n)\Phi^{-1}(n)\mathbf{u}(n) + |\xi(n)|^2\,\mathbf{u}^H(n)\Phi^{-1}(n)\Phi(n)\Phi^{-1}(n)\mathbf{u}(n) \\
=&\epsilon^H(n-1)\left[\Phi(n) - \Phi(n-1)\right] - \epsilon(n-1) - \xi(n)\mathbf{u}^H(n)\epsilon(n-1) \\
&- \xi^*(n)\epsilon^H(n-1)\mathbf{u}(n) + |\xi(n)|^2\,\mathbf{u}^H(n)\Phi^{-1}(n)\mathbf{u}(n)
\end{aligned} \tag{3}$$

Moreover, from RLS theory:

$$\Phi(n) = \Phi(n-1) + \mathbf{u}(n)\mathbf{u}^H(n), \qquad \lambda = 1 \tag{4}$$

$$r^{-1}(n) = 1 + \mathbf{u}^H(n)\Phi^{-1}(n)\mathbf{u}(n), \qquad \lambda = 1 \tag{5}$$

Hence, using Eqs. (4) and (5) in (3) yields

$$\begin{aligned} X(n) =& \epsilon^H(n-1)\mathbf{u}(n)\mathbf{u}^H(n)\epsilon(n-1) - \xi(n)\mathbf{u}^H(n)\epsilon(n-1) - \xi^*(n)\epsilon^H(n-1)\mathbf{u}(n) \\ &+ |\xi(n)|^2 \left(1 - r^{-1}(n)\right) \end{aligned} \tag{6}$$

Again, from RLS theory:

$$\epsilon^H(n-1)\mathbf{u}(n) = \xi_u(n) \tag{7}$$

$$\nu(n) = \xi(n) - \xi_u(n) \tag{8}$$

Hence, substituting Eqs. (7) and (8) into Eq. (6), we finally get

$$\begin{aligned} X(n) &= |\xi_u(n)|^2 - \xi(n)\xi_u^*(n) - \xi^*(n)\xi_u(n) + |\xi(n)|^2 \left(1 - r^{-1}(n)\right) \\ &= |\xi(n) - \xi_u(n)|^2 - r^{-1}(n)\,|\xi(n)|^2 \\ &= |v(n)|^2 - r^{-1}(n)\,|\xi(n)|^2 \end{aligned} \tag{9}$$

Recalling the definition of $X(n)$ given in Equation (1), we may thus write

$$\epsilon^H(n)\Phi(n)\epsilon(n) - \epsilon^H(n-1)\Phi(n-1)\epsilon(n-1) = |v(n)|^2 - r^{-1}(n)\,|\xi(n)|^2$$

or equivalently

$$\epsilon^H(n)\Phi(n)\epsilon(n) + \frac{|\xi(n)|^2}{r(n)} = \epsilon^H(n-1)\Phi(n-1)\epsilon(n-1) + |\nu(n)|^2$$

which is the desired result.

Problem 11.10

In light of the material covered thus far on the LMS and RLS algorithms, we may make the following statements:

a) The LMS algorithm is typically more robust than the RLS algorithm.

b) In statistical terms, the RLS algorithm is typically more efficient than the LMS algorithm: To be specific, the rate of convergence of the RLS algorithm is faster than that of the LMS algorithm.

c) Computational complexity of the LMS algorithm follows a linear law; whereas computational complexity of the RLS algorithm follows a square law.

In graphical terms, we may distinguish between the LMS and RLS algorithm as depicted in the following Figure.

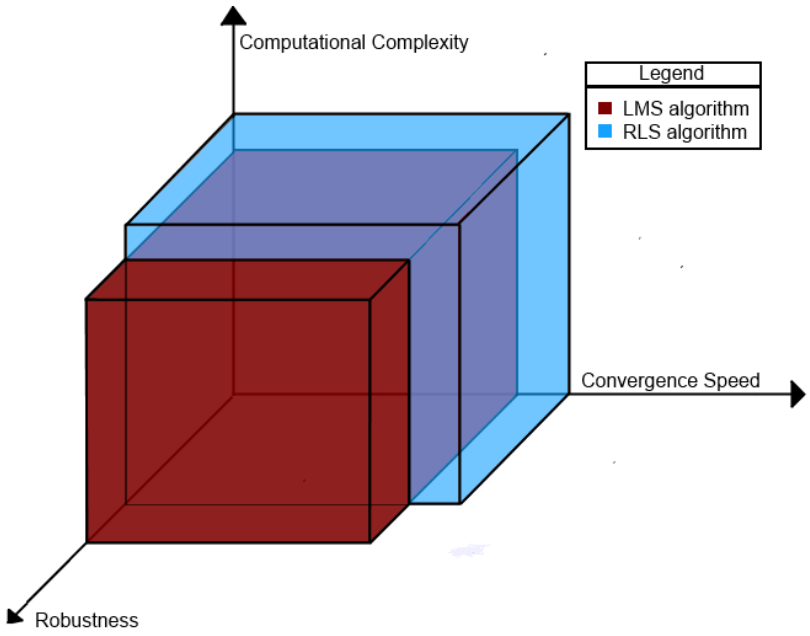


Figure 1

Chapter 12

Problem 12.1

The analog (infinite-precision) model of the LMS algorithm is described by

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu \mathbf{u}(n) \left[d(n) - \mathbf{w}^T(n)\mathbf{u}(n) \right], \qquad n = 0, 1, 2, \ldots$$

where $\hat{\mathbf{w}}(n)$ is the tap-weight vector estimate at time n, $\mathbf{u}(n)$ is the tap-input vector, $d(n)$ is the desired response, and μ is the step-size parameter. The digital (finite-precision) counterpart of this model may be expressed as follows:

$$\hat{\mathbf{w}}_q(n+1) = \hat{\mathbf{w}}_q(n) + Q\left[\mu \mathbf{u}_q(n) e_q(n)\right]$$

where

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

and use of the subscript q signifies the use of finite-precision arithmetic. Let

$$Q\left[\mu \mathbf{u}_q(n) e_q(n)\right] = \mu \mathbf{u}_q(n) e_q(n) + \mathbf{v}(n)$$

where the quantizing noise vector $\mathbf{v}(n)$ is determined by the manner in which the term $\mathbf{u}_q(n)e_q(n)$ is computed. Hence,

$$\hat{\mathbf{w}}_q(n+1) = \hat{\mathbf{w}}_q(n) + \mu \mathbf{u}_q(n) e_q(n) + \mathbf{v}(n) \tag{1}$$

The quantized value of the estimation error $e(n)$ may be expressed as

$$e_q(n) = e(n) - \Delta\mathbf{u}^T(n)\mathbf{w}(n) - \mathbf{u}^T(n)\Delta\mathbf{w}(n) + \zeta(n)$$

where $\zeta(n)$ denotes residual error. The quantized value of $\mathbf{u}(n)$ is given by

$$\mathbf{u}_q(n) = \mathbf{u}(n) + \Delta\mathbf{u}(n)$$

Hence, we may express $\mathbf{u}_q(n)e_q(n)$, ignoring second-order effects, as follows

$$\begin{aligned}\mathbf{u}_q(n)e_q(n) =&\mathbf{u}(n)e(n) + \Delta\mathbf{u}(n)e(n) - \mathbf{u}(n)\Delta\mathbf{u}^T(n)\hat{\mathbf{w}}(n)\\ &- \mathbf{u}(n)\mathbf{u}^T(n)\Delta\hat{\mathbf{w}}(n) + \mathbf{u}(n)\zeta(n)\end{aligned}$$

We may therefore rewrite Eq. (1) as

$$\begin{aligned}\hat{\mathbf{w}}_q(n+1) =&\hat{\mathbf{w}}_q(n) + \mu\mathbf{u}(n)e(n) + \mu\Delta\mathbf{u}(n)e(n) - \mu\mathbf{u}(n)\Delta\mathbf{u}^T(n)\hat{\mathbf{w}}(n)\\ &- \mu\mathbf{u}(n)\mathbf{u}^T(n)\Delta\hat{\mathbf{w}}(n) + \mu\mathbf{u}(n)\zeta(n) + \mathbf{v}(n)\end{aligned} \tag{2}$$

But

$$\hat{\mathbf{w}}_q(n+1) = \hat{\mathbf{w}}(n+1) + \Delta\hat{\mathbf{w}}(n+1)$$

$$\hat{\mathbf{w}}_q(n) = \hat{\mathbf{w}}(n) + \Delta\hat{\mathbf{w}}(n)$$

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu\mathbf{u}(n)e(n)$$

Hence, from Eq. (2) we deduce that

$$\Delta\hat{\mathbf{w}}(n+1) = \mathbf{F}(n)\Delta\mathbf{w}(n) + \mathbf{t}(n)$$

where

$$\mathbf{F}(n) = \mathbf{I} - \mu\mathbf{u}(n)\mathbf{u}^T(n)$$

$$\mathbf{t}(n) = \mu\Delta\mathbf{u}(n)e(n) - \mu\mathbf{u}(n)\hat{\mathbf{w}}^T(n)\Delta\mathbf{u}(n) + \mu\mathbf{u}(n)\zeta(n) + \mathbf{v}(n)$$

Note that the inner product

$$\Delta\mathbf{u}^T(n)\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}^T(n)\Delta\mathbf{u}(n)$$

Problem 12.2

Building on the solution to Problem 12.1, assume that $\Delta\hat{\mathbf{w}}(n)$ is stationary. Then the expectation of $\Delta\hat{\mathbf{w}}(n)$ is zero because the expectation of $\mathbf{t}(n)$ is zero.

Problem 12.3

We note that

$$y_I(n) = \sum_i w_i u(n-i)$$

$$y_{II}(n) = \sum_i w_{iq} u(n-i)$$

Hence,

$$\begin{aligned}\varepsilon(n) =& y_I(n) - y_{II}(n) \\ =& \sum_i (w_i - w_{iq})\, u(n-i)\end{aligned}$$

The mean-squared value of $\varepsilon(n)$ is

$$\begin{aligned}\mathbb{E}\left[\varepsilon^2(n)\right] =& \mathbb{E}\left[\sum_i \sum_j (w_i - w_{iq})(w_j - w_{jq})\, u(n-i)u(n-j)\right] \\ =& \sum_i \sum_j (w_i - w_{iq})(w_j - w_{jq})\, \mathbb{E}\left[u(n-i)u(n-j)\right] \qquad (1)\end{aligned}$$

Assuming that

$$\mathbb{E}\left[u(n-i)u(n-j)\right] = \begin{cases} A_{\text{rms}}^2, & j = i \\ 0, & j \neq i \end{cases}$$

we may then simplify Eq. (1) as follows:

$$\mathbb{E}\left[\varepsilon^2(n)\right] = A_{\text{rms}}^2 \sum_i (w_i - w_{iq})^2\,, \qquad i = 1, \ldots, m-1$$

Problem 12.4

a)

The digital residual error is defined by

$$e_D(n) = \frac{\text{LSB}}{\mu A_{\text{rms}}}$$

With 12-bit quantization, the least significant bit is

$$\text{LSB} = 2^{-12} \approx 0.25 \times 10^{-3}$$

We are given

$$\mu = 0.07$$

$$A_{\text{rms}} = 1$$

Hence, the digital residual error is

$$\begin{aligned} e_d(n) =& \frac{2^{-12}}{0.07 \times 1} \\ \approx& \frac{0.25 \times 10^{-3}}{0.07} \\ =& 0.35 \times 10^{-2} \end{aligned}$$

b)

The rms quantization error is

$$\begin{aligned} (QE)_{\text{rms}} =& \sqrt{\mathbb{E}\left[\varepsilon^2(n)\right]} \\ =& A_{\text{rms}} \left(\sum_{i=0}^{M-1} (w_i - w_{iq})^2 \right)^2 \\ \leq& \left(A_{\text{rms}} M^{1/2}\right) (\text{LSB}) \end{aligned}$$

For the problem at hand we have

$$(QE)_{\text{rms}} \approx \sqrt{17} \times 0.25 \times 10^{-3} \approx 10^{-3}$$

We thus see that $(QE)_{\text{rms}}$ is about 3.5 times worse than the digital residual error.

Problem 12.5

In Chapter 10 on the RLS algorithm, the basic issue of interest was that of statistical performance, and how to compare it versus the LMS algorithm. Therein, the algorithmic study of the RLS algorithm was based on the presumption of working on an analog model, that

is, a model with infinite precision. Referring to the second paragraph under the summary of the RLS algorithm on page 438 of the textbook was the following:

$$\boldsymbol{\pi}(n) = \lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n) \tag{1}$$

Turning next to Table 12.1, we introduced $\boldsymbol{\pi}(n)$ slightly differently from (1), as shown by

$$\boldsymbol{\pi}(n) = \mathbf{P}(n-1)\mathbf{u}(n) \tag{2}$$

To be more complete, two other changes were made in Table 12.1, namely:

$$\lambda(n) = \frac{1}{\lambda + \mathbf{u}^H(n)\boldsymbol{\pi}} \tag{3}$$

$$\mathbf{k}(n) = r(n)\boldsymbol{\pi}(n) \tag{4}$$

In effect the definition of the Kalman gain, $\mathbf{k}(n)$, in Table 10.1, page 439 of the Textbook, was replaced by (2) to (4) in Table 12.1. To reiterate:

i) Simply put, in Table 10.1 the presentation was based on an analog model for formulating the RLS algorithm.

ii) On the other hand, in Table 12.1 the objective is to organize the table so to exploit the use of finite arithmetic in as good of a manner as possible.

Problem 12.6

We start with

$$\begin{aligned}\frac{1}{\kappa_q(n)} =& \frac{1}{\lambda + \pi_q(n)\mathbf{u}(n)} \\ \approx& \frac{1}{\lambda + \pi(n)\mathbf{u}(n)}\left(1 - \frac{\eta_\pi(n)\mathbf{u}(n)}{\lambda + \pi(n)\mathbf{u}(n)}\right)\end{aligned} \tag{1}$$

where it is noted that

$$\pi_q(n) = \pi(n) + \eta_\pi(n)$$

Next, we note that

$$\mathbf{k}_q(n) = \frac{\mathbf{P}_q(n-1)\mathbf{u}(n)}{\kappa_q(n)} \tag{2}$$

Let

$$\mathbf{P}_q(n-1) = \mathbf{P}(n-1) + \eta_P(n-1) \tag{3}$$

Therefore, using Eqs. (1) and (3) in (2), we may write

$$\begin{aligned}
\eta_k(n) =& \mathbf{k}_q(n) - \mathbf{k}(n) \\
=& \left[\left(\mathbf{P}(n-1) + \eta_P(n-1)\right)\mathbf{u}(n)\right]\left[\frac{1}{\lambda + \pi(n)\mathbf{u}(n)}\left(1 - \frac{\eta_\pi(n)\mathbf{u}(n)}{\lambda + \pi(n)\mathbf{u}(n)}\right)\right] - \mathbf{k}(n) \\
\approx& \frac{\mathbf{P}(n-1)\mathbf{u}(n)}{\lambda + \pi(n)\mathbf{u}(n)} + \frac{\eta_P(n-1)\mathbf{u}(n)}{\lambda + \pi(n)\mathbf{u}(n)} - \frac{\mathbf{P}(n-1)\mathbf{u}(n)\eta_\pi(n)\mathbf{u}(n)}{\left(\lambda + \pi(n)\mathbf{u}(n)\right)^2} - \mathbf{k}(n) \\
=& \frac{\eta_P(n-1)\mathbf{u}(n)}{\lambda + \pi(n)\mathbf{u}(n)} - \frac{\mathbf{P}(n-1)\mathbf{u}(n)\eta_\pi(n)\mathbf{u}(n)}{\left(\lambda + \pi(n)\mathbf{u}(n)\right)^2}
\end{aligned}$$

We next introduce the following expression (using the relation $\eta_\pi(n) = \mathbf{u}^H(n)\eta_P(n-1)$)

$$\begin{aligned}
\eta_{P'}(n) =& \eta_k(n)\pi(n) + \mathbf{k}(n)\eta_\pi(n) \\
\approx& \frac{\eta_P(n-1)\mathbf{u}(n)\pi(n)}{\lambda + \pi(n)\mathbf{u}(n)} - \frac{\mathbf{P}(n-1)\mathbf{u}(n)\eta_\pi(n)\mathbf{u}(n)\pi(n)}{\left(\lambda + \pi(n)\mathbf{u}(n)\right)^2} + \frac{\mathbf{P}(n-1)\mathbf{u}(n)\eta_\pi(n)}{\lambda + \pi(n)\mathbf{u}(n)} \\
=& \frac{\eta_P(n-1)\mathbf{u}(n)\pi(n)}{\lambda + \pi(n)\mathbf{u}(n)} - \frac{\mathbf{P}(n-1)\mathbf{u}(n)\mathbf{u}^T(n)\eta_P(n-1)\mathbf{u}(n)\pi(n)}{\left(\lambda + \pi(n)\mathbf{u}(n)\right)^2} \\
&+ \frac{\mathbf{P}(n-1)\mathbf{u}(n)\mathbf{u}^T(n)\eta_P(n-1)}{\lambda + \pi(n)\mathbf{u}(n)}
\end{aligned}$$

Finally, we calculate

$$\begin{aligned}
\eta_P(n) =& \frac{1}{\lambda}\left(\eta_P(n-1) - \eta_{P'}(n-1)\right) \\
=& \frac{1}{\lambda}\left(\eta_P(n-1) - \frac{\eta_P(n-1)\mathbf{u}(n)\pi(n)}{\lambda + \pi(n)\mathbf{u}(n)} + \frac{\mathbf{P}(n-1)\mathbf{u}(n)\mathbf{u}^T(n)\eta_P(n-1)\mathbf{u}(n)\pi(n)}{\left(\lambda + \pi(n)\mathbf{u}(n)\right)^2}\right. \\
&\left. - \frac{\mathbf{P}(n-1)\mathbf{u}(n)\mathbf{u}^T(n)\eta_P(n-1)}{\lambda + \pi(n)\mathbf{u}(n)}\right)
\end{aligned}$$

We now note that

$$\frac{\mathbf{P}(n-1)\mathbf{u}(n)}{\lambda+\pi(n)\mathbf{u}(n)} = \mathbf{k}(n)$$

$$\pi(n) = \mathbf{u}^T(n)\mathbf{P}(n-1)$$

Hence, assuming that $\mathbf{P}(n-1)$ is symmetric:

$$\begin{aligned}\eta_P(n) =& \frac{1}{\lambda}\left(\eta_P(n-1) - \eta_P(n-1)\mathbf{u}(n)\mathbf{k}^T(n) + \mathbf{k}(n)\mathbf{u}^T(n)\eta_P(n-1)\mathbf{u}(n)\mathbf{k}^T(n)\right.\\ &\left.-\mathbf{k}(n)\mathbf{u}^T(n)\eta_P(n-1)\right)\\ =& \frac{1}{\lambda}\left(\mathbf{I}-\mathbf{k}(n)\mathbf{u}^T(n)\right)\eta_P(n) - \frac{1}{\lambda}\left(\mathbf{I}-\mathbf{k}(n)\mathbf{u}^T(n)\right)\eta_P(n-1)\mathbf{u}(n)\mathbf{k}^T(n)\\ =& \frac{1}{\lambda}\left(\mathbf{I}-\mathbf{k}(n)\mathbf{u}^T(n)\right)\eta_P(n-1)\left(\mathbf{I}-\mathbf{k}(n)\mathbf{u}^T(n)\right)^T\end{aligned}$$

From this result we readily see that $\eta_P^T(n) = \eta_P(n)$, which demonstrates the symmetry preserving-property of the RLS algorithm summarized in Table 12.1.

Problem 12.7

The condition for persistent excitation (assuming real data) may be expressed as

$$a\mathbf{I} \leq \sum_{i=n_0}^{n} \lambda^{n-i}\mathbf{u}(i)\mathbf{u}^T(i) \leq b\mathbf{I} \tag{1}$$

where a and b are both positive numbers. Premultiplying Equation (1) by $\mathbf{z}^T$ and postmultiplying by $\mathbf{z}$:

$$a\mathbf{z}^T\mathbf{z} \leq \sum_{i=n_0}^{n} \lambda^{n-i}\mathbf{z}^T\mathbf{u}(i)\mathbf{u}^T(i)\mathbf{z} \leq b\mathbf{z}^T\mathbf{z}$$

We now recognize that

$$\mathbf{z}^T\mathbf{u}(i) = \mathbf{u}^T(i)\mathbf{z}$$

Hence,

$$a\mathbf{z}^T\mathbf{z} \leq \sum_{i=n_0}^{n} \lambda^{n-i}\left|\mathbf{z}^T\mathbf{u}(i)\right|^2 \leq b\mathbf{z}^T\mathbf{z} \tag{2}$$

For a nonzero vector **z**, this condition requires that we have

$$\mathbf{z}^T\mathbf{u}(i) > \alpha \qquad \text{for } n_0 \leq i \leq n$$

Where α is some positive constant that satisfies Equation (2). This is another way of defining the condition of persistent excitation

Chapter 13

Problem 13.1

In an adaptive equalizer, the input signal equals the channel output and the desired response equals the channel input (i.e., transmitted signal). In a stationary environment, both of these signals are stationary with the result that the error-performance surface is fixed in all respects. On the other hand, in a nonstationary environment, the channel output (i.e., equalizer input) is nonstationary with the result that both the correlation matrix $\mathbf{R}$ of the input vector and the cross-correlation vector $\mathbf{p}$ between the input vector and desired response take on time-varying forms. Consequently, the error-performance surface is continually changing its shape; moreover, it is also in a constant state of motion.

Problem 13.2

In adaptive prediction applied to a nonstationary process, both the input vector (defined by a set of past values of the process) and the desired response (defined by the present value of the process) are nonstationary. Accordingly, in such a case the error-performance surface behaves in a manner similar to that described for adaptive equalization in Problem 13.1. Specifically, the error-performance surface constantly changes its shape and constantly moves. In contrast, the error-performance surface for the adaptive prediction of a stationary process is completely fixed.

Problem 13.3

We have, by definition

$$\varepsilon_1(n) = \hat{\mathbf{w}}(n) - \mathbb{E}\left[\hat{\mathbf{w}}(n)\right]$$

$$\varepsilon_2(n) = \mathbb{E}\left[\hat{\mathbf{w}}(n)\right] - \mathbf{w}_0(n)$$

We may therefore expand

$$\begin{aligned}\mathbb{E}\left[\varepsilon_1^H(n)\varepsilon_2(n)\right] =& \mathbb{E}\left[\left(\hat{\mathbf{w}}(n) - \mathbb{E}\left[\hat{\mathbf{w}}(n)\right]\right)^H \left(\mathbb{E}\left[\hat{\mathbf{w}}(n)\right] - \mathbf{w}_0\right)\right] \\ =& \mathbb{E}\left[\hat{\mathbf{w}}^H(n)\mathbb{E}\left[\hat{\mathbf{w}}(n)\right] - \mathbb{E}\left[\hat{\mathbf{w}}^H(n)\right]\mathbb{E}\left[\hat{\mathbf{w}}(n)\right]\right. \\ & \left. -\hat{\mathbf{w}}^H(n)\mathbf{w}_0 + \mathbb{E}\left[\hat{\mathbf{w}}^H(n)\right]\mathbf{w}_0\right] \\ =& -\mathbb{E}\left[\hat{\mathbf{w}}^H(n)\mathbf{w}_0\right] + \mathbb{E}\left[\hat{\mathbf{w}}^H(n)\right]\mathbb{E}\left[\mathbf{w}_0\right]\end{aligned}$$

Invoking the assumption that $\mathbf{w}(n)$ and $\mathbf{w}_0$ are statistically independent, we may go on to write

$$\begin{aligned}\mathbb{E}\left[\varepsilon_1^H(n)\varepsilon_2(n)\right] &= \left(-\mathbb{E}\left[\hat{\mathbf{w}}^H(n)\right]\mathbb{E}\left[\mathbf{w}_0\right]\right) + \mathbb{E}\left[\hat{\mathbf{w}}^H(n)\right]\mathbb{E}\left[\mathbf{w}_0\right] \\ &= \mathbf{0} \end{aligned} \tag{1}$$

From this result we immediately deduce that we also have

$$\mathbb{E}\left[\varepsilon_2^H(n)\varepsilon_1(n)\right] = \mathbf{0} \tag{2}$$

Finally, we note that

$$\begin{aligned}\mathbb{E}\left[\|\varepsilon(n)\|^2\right] =& \mathbb{E}\left[\varepsilon^H(n)\varepsilon(n)\right] \\ =& \mathbb{E}\left[\left(\varepsilon_1(n) + \varepsilon_2(n)\right)^H \left(\varepsilon_1(n) + \varepsilon_2(n)\right)\right] \\ =& \mathbb{E}\left[\|\varepsilon_1(n)\|^2\right] + \mathbb{E}\left[\varepsilon_1^H(n)\varepsilon_2(n)\right] \\ & + \mathbb{E}\left[\varepsilon_2^H(n)\varepsilon_1(n)\right] + \mathbb{E}\left[\|\varepsilon_2(n)\|^2\right] \\ =& \mathbb{E}\left[\|\varepsilon_1(n)\|^2\right] + \mathbb{E}\left[\|\varepsilon_2(n)\|^2\right]\end{aligned}$$

where in the last line we have made use of Equations (1) and (2).

Problem 13.4

Invoking the low-pass filtering action of the LMS filter for small μ, we write that $\varepsilon_1(n)$ and $\varepsilon_2(n)$ are both independent of the input vector $\mathbf{u}(n)$. We may therefore write:

$$\begin{aligned}
\mathbb{E}\left[\varepsilon_1^H(n)\mathbf{u}(n)\mathbf{u}^H(n)\varepsilon_1(n)\right] =& \text{tr}\left\{\mathbb{E}\left[\varepsilon_1^H(n)\mathbf{u}(n)\mathbf{u}^H(n)\varepsilon_1(n)\right]\right\} \\
=& \mathbb{E}\left[\text{tr}\left\{\varepsilon_1^H(n)\mathbf{u}(n)\mathbf{u}^H(n)\varepsilon_1(n)\right\}\right] \\
=& \mathbb{E}\left[\text{tr}\left\{\mathbf{u}(n)\mathbf{u}^H(n)\varepsilon_1(n)\varepsilon_1^H(n)\right\}\right] \\
=& \text{tr}\left\{\mathbb{E}\left[\mathbf{u}(n)\mathbf{u}^H(n)\varepsilon_1(n)\varepsilon_1^H(n)\right]\right\} \\
=& \text{tr}\left\{\mathbb{E}\left[\mathbf{u}(n)\mathbf{u}^H(n)\right]\mathbb{E}\left[\varepsilon_1(n)\varepsilon_1^H(n)\right]\right\} \\
=& \text{tr}\left\{\mathbf{R}\mathbf{K}_1(n)\right\}
\end{aligned}$$

Similarly, we may show that

$$\mathbb{E}\left[\varepsilon_2^H(n)\mathbf{u}(n)\mathbf{u}^H(n)\varepsilon_2(n)\right] = \text{tr}\left\{\mathbf{R}\mathbf{K}_2(n)\right\}$$

$$\begin{aligned}
\mathbb{E}\left[\varepsilon_1^H(n)\mathbf{u}(n)\mathbf{u}^H(n)\varepsilon_2(n)\right] =& \text{tr}\left\{\mathbb{E}\left[\varepsilon_1^H(n)\mathbf{u}(n)\mathbf{u}^H(n)\varepsilon_2(n)\right]\right\} \\
=& \text{tr}\left\{\mathbb{E}\left[\mathbf{u}(n)\mathbf{u}^H(n)\right]\mathbb{E}\left[\varepsilon_2(n)\varepsilon_1^H(n)\right]\right\} \\
=& \text{tr}\left\{\mathbf{R}\mathbb{E}\left[\varepsilon_2(n)\varepsilon_1^H(n)\right]\right\}
\end{aligned}$$

Next we note that

$$\mathbb{E}\left[\varepsilon_2(n)\varepsilon_1^H(n)\right] = \mathbf{0}$$

It follows therefore that

$$\mathbb{E}\left[\varepsilon_1^H(n)\mathbf{u}(n)\mathbf{u}^H(n)\varepsilon_2(n)\right] = \mathbf{0}$$

Similarly, we may show that

$$\mathbb{E}\left[\varepsilon_2^H(n)\mathbf{u}(n)\mathbf{u}^H(n)\varepsilon_1(n)\right] = \mathbf{0}$$

Problem 13.5

Evaluating the mean-square values of both sides of Equation (13.27) in the textbook yields

$$\begin{aligned}\mathbb{E}\left[|\nu_k(n+1)|^2\right] &= (1-\mu\lambda_{u,k})^2\,\mathbb{E}\left[|\nu_k(n)|^2\right] + \mathbb{E}\left[|\phi_k(n)|^2\right] \\ &= \left(1-2\mu\lambda_{u,k}+\mu^2\lambda_{u,k}^2\right)\mathbb{E}\left[|\nu_k(n)|^2\right] + \sigma_{\phi_k}^2 \end{aligned} \tag{1}$$

For small μ, we may justifiably ignore the term $\mu^2\lambda_{u,k}^2$ in comparison to the unity term, and so approximate Equation (1) as

$$\mathbb{E}\left[|\nu_k(n+1)|^2\right] \approx (1-2\mu\lambda_{u,k})\,\mathbb{E}\left[|\nu_k(n)|^2\right] + \sigma_{\phi_k}^2 \tag{2}$$

Under steady-state conditions, $\nu_k(n+1) \to \nu_k(n)$ as $n \to \infty$, in which case Equation (2) reduces further to

$$\begin{aligned}2\mu\lambda_{u,k}\mathbb{E}\left[|\nu_k(n)|^2\right] \approx& \sigma_{\phi_k}^2 \\ \approx& \mu^2\sigma_\nu^2\lambda_{u,k} + \lambda_{\omega,k}\end{aligned}$$

Solving for $\mathbb{E}\left[|\nu_k(n)|^2\right]$, we get

$$\mathbb{E}\left[|\nu_k(n)|^2\right] \approx \frac{\sigma_\nu^2}{2}\mu + \frac{\lambda_{\omega,k}}{2\mu\lambda_{u,k}} \tag{3}$$

Hence, the mean-square deviation is (see Equation (13.31) of the textbook)

$$\begin{aligned}\mathscr{D}(n) =& \mathbb{E}\left[\|\varepsilon_0(n)\|^2\right] \\ =& \sum_{k=1}^{M}\mathbb{E}\left[|\nu_k(n)|^2\right] \\ =& \frac{M}{2}\sigma_\nu^2\mu + \frac{1}{2}\sum_{k=1}^{M}\frac{\lambda_{\omega,k}}{\lambda_{u,k}} \\ =& \frac{M}{2}\sigma_\nu^2\mu + \frac{1}{2}\text{tr}\left[\mathbf{R}_u^{-1}\mathbf{R}_\omega\right]\end{aligned} \tag{4}$$

where

$$\mathbf{R}_u = \mathbb{E}\left[\mathbf{u}(n)\mathbf{u}^H(n)\right], \quad \mathbf{u}(n) = \text{ tap-input vector}$$

$$\mathbf{R}_\omega = \mathbb{E}\left[\omega(n)\omega^H(n)\right], \quad \omega(n) = \text{ process noise vector}$$

Problem 13.6

The misadjustment of the LMS algorithm is given by (see Equation (13.36)of the textbook)

$$\mathscr{M} = \frac{1}{\sigma_\nu^2} \sum_{k=1}^{M} \lambda_{u,k} \mathbb{E}\left[|\nu_k(n)|^2\right] \tag{1}$$

where M is the filter length. From the solution to Problem 13.5, we have

$$\mathbb{E}\left[|\nu_k(n)|^2\right] \approx \frac{\mu}{2}\sigma_\nu^2 + \frac{1}{2\mu}\frac{\lambda_{\omega,k}}{\lambda_{u,k}} \tag{2}$$

Substituting Equation (2) into (1), we get

$$\begin{aligned}
\mathscr{M} \approx & \frac{1}{\sigma_\nu^2} \sum_{k=1}^{M} \lambda_{u,k} \left(\frac{\mu}{2}\sigma_\nu^2 + \frac{1}{2\mu}\frac{\lambda_{\omega,k}}{\lambda_{u,k}} \right) \\
= & \frac{\mu}{2} \sum_{k=1}^{M} \lambda_{u,k} + \frac{1}{2\mu\sigma_\nu^2} \sum_{k=1}^{M} \lambda_{\omega,k} \\
= & \frac{\mu}{2} \text{tr}\left\{\mathbf{R}_u\right\} + \frac{1}{2\mu\sigma_\nu^2} \text{tr}\left\{\mathbf{R}_\omega\right\}
\end{aligned} \tag{3}$$

which is the desired result.

Problem 13.7

To simplify the presentation, we use the following notation of solving this problem:

$$\mathbf{R}_u = \mathbf{R} \text{ and } \mathbf{R}_\omega = \mathbf{Q}$$

The minimum misadjustment for the LMS algorithm is

$$\mathscr{M}_{\text{min}}^{\text{LMS}} = \frac{1}{\sigma_\nu} \sqrt{\text{tr}\left\{\mathbf{R}\right\} \text{tr}\left\{\mathbf{Q}\right\}} \tag{1}$$

$$\mathscr{M}_{\text{min}}^{\text{RLS}} = \frac{1}{\sigma_\nu} \sqrt{M \text{tr}\left\{\mathbf{RQ}\right\}} \tag{2}$$

For $\mathbf{Q} = c_1\mathbf{R}$, Equations (1) and (2) yield the ratio:

$$\frac{\mathscr{M}_{\text{min}}^{\text{LMS}}}{\mathscr{M}_{\text{min}}^{\text{RLS}}} = \frac{\text{tr}\left\{\mathbf{R}\right\}}{\sqrt{M \text{tr}\left\{\mathbf{R}^2\right\}}} \tag{3}$$

Now

$$\mathbf{R} = \sum_{i=1}^{M} \lambda_i \mathbf{q}_i \mathbf{q}_i^H$$

$$\mathbf{R}^2 = \sum_{i=1}^{M} \lambda_i^2 \mathbf{q}_i \mathbf{q}_i^H$$

We may therefore write

$$\text{tr}\left\{\mathbf{R}\right\} = \sum_{i=1}^{M} \lambda_i \tag{4}$$

$$\text{tr}\left\{\mathbf{R^2}\right\} = \sum_{i=1}^{M} \lambda_i^2 \tag{5}$$

Let

$$\lambda = \begin{bmatrix} \lambda_1 & \lambda_2 & \dots & \lambda_M \end{bmatrix}^T$$

$$\mathbf{1} = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}^T$$

We may then reformulate Equations (4) and (5) as follows, respectively:

$$\text{tr}\left\{\mathbf{R}\right\} = \lambda^T \mathbf{1}$$

$$\text{tr}\left\{\mathbf{R}^2\right\} = \lambda^T \lambda = \|\lambda\|^2$$

Applying the Cauchy-Schwarz inequality to the matrix product $\lambda^T \mathbf{1}$:

$$\left|\lambda^T \mathbf{1}\right|^2 \leq \|\lambda\|^2 \cdot \|\mathbf{1}\|^2$$

Since $\|\mathbf{1}\|^2 = M$, it follows that

$$\left(\text{tr}\left\{\mathbf{R}\right\}\right)^2 \leq \text{tr}\left\{\mathbf{R}^2\right\} \cdot M$$

or, equivalently,

$$\frac{\text{tr}\left\{\mathbf{R}\right\}}{\sqrt{M \text{tr}\left\{\mathbf{R}^2\right\}}} \leq 1$$

Accordingly, we may rewrite Equation (3) as

$$\frac{\mathscr{M}_{\text{min}}^{\text{LMS}}}{\mathscr{M}_{\text{min}}^{\text{RLS}}} \leq 1 \tag{6}$$

or

$$\mathscr{M}_{\text{min}}^{\text{LMS}} \leq \mathscr{M}_{\text{min}}^{\text{RLS}}, \qquad \mathbf{Q} = c_1\mathbf{R}$$

Consider next the minimum mean-square deviation as the criterion of interest. For the LMS algorithm, we have

$$\mathscr{D}_{\text{min}}^{\text{LMS}} = \sigma_\nu \sqrt{M\text{tr}\left\{\mathbf{R}^{-1}\mathbf{Q}\right\}}$$

and for the RLS algorithm:

$$\mathscr{D}_{\text{min}}^{\text{RLS}} = \sigma_\nu \sqrt{\text{tr}\left\{\mathbf{R}^{-1}\right\}\text{tr}\left\{\mathbf{Q}\right\}}$$

Therefore,

$$\frac{\mathscr{D}_{\text{min}}^{\text{LMS}}}{\mathscr{D}_{\text{min}}^{\text{RLS}}} = \sqrt{\frac{M\text{tr}\left\{\mathbf{R}^{-1}\mathbf{Q}\right\}}{\text{tr}\left\{\mathbf{R}^{-1}\right\}\text{tr}\left\{\mathbf{Q}\right\}}}$$

For $\mathbf{Q} = c_2\mathbf{R}^{-1}$,

$$\frac{\mathscr{D}_{\text{min}}^{\text{LMS}}}{\mathscr{D}_{\text{min}}^{\text{RLS}}} = \frac{\sqrt{M\text{tr}\left\{\mathbf{R}^{-2}\right\}}}{\text{tr}\left\{\mathbf{R}^{-1}\right\}} \tag{7}$$

Since,

$$\mathbf{R}^{-1} = \sum_{i=1}^{M} \lambda_i^{-1}\mathbf{q}_i\mathbf{q}_i^H$$

and

$$\mathbf{R}^{-2} = \sum_{i=1}^{M} \lambda_i^{-2}\mathbf{q}_i\mathbf{q}_i^H$$

it follows that

$$\text{tr}\left\{\mathbf{R}^{-1}\right\} = \sum_{i=1}^{M} \lambda_i^{-1}$$

and

$$\text{tr}\left\{\mathbf{R}^{-2}\right\} = \sum_{i=1}^{M} \lambda_i^{-2}$$

Let

$$\lambda_{\text{inv}} = \begin{bmatrix} \lambda_1^{-1} & \lambda_2^{-1} & \dots & \lambda_M^{-1} \end{bmatrix}^T$$

$$\mathbf{1} = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}^T$$

Hence,

$$\text{tr}\left\{\mathbf{R}^{-1}\right\} = \lambda_{\text{inv}}^T \mathbf{1}$$

$$\text{tr}\left\{\mathbf{R}^{-2}\right\} = \lambda_{\text{inv}}^T \lambda_{\text{inv}} = \|\lambda_{\text{inv}}\|^2$$

Applying the Cauchy-Schwarz inequality to the matrix product $\lambda_{\text{inv}}^T \mathbf{1}$, we may write

$$\left|\lambda_{\text{inv}}^T \mathbf{1}\right|^2 = \|\lambda_{\text{inv}}\|^2 \cdot \|\mathbf{1}\|^2$$

That is,

$$\left(\text{tr}\left\{\mathbf{R}^{-1}\right\}\right)^2 \leq \text{tr}\left\{\mathbf{R}^{-2}\right\} \cdot M$$

or equivalently,

$$\frac{\sqrt{M \text{tr}\left\{\mathbf{R}^{-2}\right\}}}{\text{tr}\left\{\mathbf{R}^{-1}\right\}} \geq 1 \tag{8}$$

Accordingly, we may rewrite Equation (7) as

$$\frac{\mathscr{D}_{\text{min}}^{\text{LMS}}}{\mathscr{D}_{\text{min}}^{\text{RLS}}} \geq 1$$

That is,

$$\mathscr{D}_{\text{min}}^{\text{LMS}} \geq \mathscr{D}_{\text{min}}^{\text{RLS}}, \qquad \mathbf{Q} = c_2 \mathbf{R}^{-1}$$

Problem 13.8

As with Problem 13.7, here again we simplify the presentation by using the notations

$$\mathbf{R}_u = \mathbf{R} \text{ and } \mathbf{R}_\omega = \mathbf{Q}$$

a) LMS Algorithm
For $\mathbf{Q} = c_1\mathbf{R}$, or equivalently, $\mathbf{R}^{-1}\mathbf{Q} = c_1\mathbf{I}$:

(i)

$$\begin{aligned}\mathscr{D}_{\text{min}}^{\text{LMS}} =& \sigma_\nu\sqrt{M}\left(\text{tr}\left\{\mathbf{R}^{-1}\mathbf{Q}\right\}\right)^{1/2}\\ =& \sigma_\nu\sqrt{M}\left(\text{tr}\left\{c_1\mathbf{I}\right\}\right)^{1/2}\\ =& \sigma_\nu M\sqrt{c_1}\end{aligned} \tag{1}$$

$$\begin{aligned}\mu_{\text{opt}} =& \frac{1}{\sigma_\nu\sqrt{M}}\left(\text{tr}\left\{\mathbf{R}^{-1}\mathbf{Q}\right\}\right)^{1/2}\\ =& \sqrt{c_1}/\sigma_\nu\end{aligned} \tag{2}$$

(ii)

$$\begin{aligned}\mathscr{M}_{\text{min}}^{\text{LMS}} =& \frac{1}{\sigma_\nu}\left(\text{tr}\left\{\mathbf{RQ}\right\}\right)^{1/2}\\ =& \frac{\sqrt{c_1}}{\sigma_\nu}\text{tr}\left\{\mathbf{R}\right\}\end{aligned} \tag{3}$$

Given the two-by-two correlation matrix

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{21} \\ r_{21} & r_{22} \end{bmatrix}$$

we may write

$$\text{tr}\left\{\mathbf{R}\right\} = r_{11} + r_{22} \tag{4}$$

Therefore, substituting Equation (4) into Equation (3):

$$\mathscr{M}_{\text{min}}^{\text{LMS}} = \frac{\sqrt{c_1}\left(r_{11} + r_{22}\right)}{\sigma_\nu} \tag{5}$$

The optimum step-size parameter is therefore

$$\mu_{\text{opt}} = \frac{1}{\sigma_\nu}\left(\frac{\text{tr}\{\mathbf{Q}\}}{\text{tr}\{\mathbf{R}\}}\right)^{1/2} = \frac{\sqrt{c_1}}{\sigma_\nu} \tag{6}$$

Which is the same as the μ_{opt} for minimum $\mathscr{D}^{\text{LMS}}$.

Consider next the case of

$$\mathbf{Q} = c_2\mathbf{R}^{-1} \text{ or equivalently } \mathbf{QR} = c_2\mathbf{I}$$

(iii)

Consider next the case

$$\mathscr{D}_{\text{min}}^{\text{LMS}} = \sigma_\nu\sqrt{M}\left(\text{tr}\left\{\mathbf{R}^{-1}\mathbf{Q}\right\}\right)^{1/2} = \sigma_\nu\sqrt{Mc_2}\left(\text{tr}\left\{\mathbf{R}^{-2}\right\}\right)^{1/2} \tag{7}$$

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{21} \\ r_{21} & r_{22} \end{bmatrix}$$

$$\mathbf{R}^{-1} = \frac{1}{\Delta_r}\begin{bmatrix} r_{22} & -r_{21} \\ -r_{21} & r_{11} \end{bmatrix}, \qquad \Delta_r = r_{11}r_{22} - r_{21}^2$$

$$\mathbf{R}^{-2} = \mathbf{R}^{-1}\mathbf{R}^{-1} = \frac{1}{\Delta_r^2}\begin{bmatrix} r_{22}^2 + r_{21}^2 & -r_{21}\left(r_{11} + r_{22}\right) \\ -r_{21}\left(r_{11} + r_{22}\right) & r_{11}^2 + r_{21}^2 \end{bmatrix}$$

$$\text{tr}\left\{\mathbf{R}^{-2}\right\} = \frac{1}{\Delta_r^2}\left(r_{11}^2 + 2r_{21}^2 + r_{22}^2\right) \tag{8}$$

Substituting Equation (8) into Equation (7), and noting that $M = 2$:

$$\mathscr{D}_{\text{min}}^{\text{LMS}} = \sigma_\nu\sqrt{2c_2}\frac{\sqrt{r_{11}^2 + 2r_{21}^2 + r_{22}^2}}{r_{11}r_{22} - r_{21}^2} \tag{9}$$

The optimum step-size parameter is given by

$$\begin{aligned}\mu_{\text{opt}} =& \frac{1}{\sigma_\nu\sqrt{M}}\left(\text{tr}\left\{\mathbf{R}^{-1}\mathbf{Q}\right\}\right)^{1/2} \\ =& \frac{1}{\sigma_\nu}\sqrt{\frac{c_2}{2}}\left(\text{tr}\left\{\mathbf{R}^{-2}\right\}\right)^{1/2} \\ =& \frac{1}{\sigma_\nu}\sqrt{\frac{c_2}{2}}\frac{\sqrt{r_{11}^2 + 2r_{21}^2 + r_{22}^2}}{r_{11}r_{22} - r_{21}^2}\end{aligned} \tag{10}$$

(iv)

$$\begin{aligned}\mathscr{M}_{\text{min}}^{\text{LMS}} =& \frac{1}{\sigma_\nu}\left(\text{tr}\left\{\mathbf{R}\right\}\text{tr}\left\{\mathbf{Q}\right\}\right)^{1/2}\\ =& \frac{\sqrt{c_2}}{\sigma_\nu}\left(\text{tr}\left\{\mathbf{R}\right\}\text{tr}\left\{\mathbf{R}^{-1}\right\}\right)^{1/2}\\ =& \frac{\sqrt{c_2}}{\sigma_\nu}\frac{r_{11}+r_{22}}{\left(r_{11}r_{22}-r_{21}^2\right)^{1/2}}\end{aligned} \tag{11}$$

$$\begin{aligned}\mu_{\text{opt}}^{\text{LMS}} =& \frac{1}{\sigma_\nu}\left(\frac{\text{tr}\left\{\mathbf{Q}\right\}}{\text{tr}\left\{\mathbf{R}\right\}}\right)^{1/2}\\ =& \frac{\sqrt{c_2}}{\sigma_\nu}\left(\frac{\text{tr}\left\{\mathbf{R}^{-1}\right\}}{\text{tr}\left\{\mathbf{R}\right\}}\right)^{1/2}\\ =& \frac{\sqrt{c_2}}{\sigma_\nu}\left(r_{11}r_{22}-r_{21}^2\right)^{-1/2}\end{aligned} \tag{12}$$

which is again different from the μ_{opt} for minimum $\mathscr{D}^{\text{LMS}}$.

b) RLS Algorithm

For $\mathbf{Q}=c_1\mathbf{R}$:

(i)

$$\begin{aligned}\mathscr{D}_{\text{min}}^{\text{RLS}} =& \sigma_\nu\left(\text{tr}\left\{\mathbf{R}^{-1}\right\}\text{tr}\left\{\mathbf{Q}\right\}\right)^{1/2}\\ =& \sigma_\nu\sqrt{c_1}\left(\text{tr}\left\{\mathbf{R}^{-1}\right\}\text{tr}\left\{\mathbf{R}\right\}\right)^{1/2}\\ =& \sigma_\nu\sqrt{c_1}\frac{r_{11}+r_{22}}{\left(r_{11}r_{22}-r_{21}^2\right)^{1/2}}\end{aligned} \tag{13}$$

$$\begin{aligned}\lambda_{\text{opt}} =& 1-\frac{1}{\sigma_\nu}\left(\frac{\text{tr}\left\{\mathbf{Q}\right\}}{\text{tr}\left\{\mathbf{R}^{-1}\right\}}\right)^{1/2}\\ =& 1-\frac{\sqrt{c_1}}{\sigma_\nu}\sqrt{r_{11}r_{22}-r_{12}^2}\end{aligned} \tag{14}$$

(ii)

$$\begin{aligned}\mathscr{M}_{\text{min}}^{\text{RLS}} =& \frac{1}{\sigma_\nu}\left(M\text{tr}\left\{\mathbf{RQ}\right\}\right)^{1/2} \\ =& \frac{\sqrt{c_1}}{\sigma_\nu}\left(2\text{tr}\left\{\mathbf{R}^2\right\}\right)^{1/2}, \qquad M=2 \\ =& \frac{\sqrt{2c_1}}{\sigma_\nu}\sqrt{r_{11}^2+2r_{21}^2+r_{22}^2}\end{aligned} \tag{15}$$

$$\begin{aligned}\lambda_{\text{opt}} =& 1-\frac{1}{\sigma_\nu}\left(\frac{1}{M}\text{tr}\left\{\mathbf{RQ}\right\}\right)^{1/2}, \qquad M=2 \\ =& 1-\frac{1}{\sigma_\nu}\sqrt{\frac{c_1}{2}}\sqrt{r_{11}^2+2r_{21}^2+r_{22}^2}\end{aligned} \tag{16}$$

which is different from the λ_{opt} for minimum $\mathscr{D}^{\text{RLS}}$.

Consider next the case of $\mathbf{Q}=c_2\mathbf{R}^{-1}$ or equivalently $\mathbf{RQ}=c_2\mathbf{I}$:

(iii)

$$\begin{aligned}\mathscr{D}_{\text{min}}^{\text{RLS}} =& \sigma_\nu\left(\text{tr}\left\{\mathbf{R}^{-1}\right\}\text{tr}\left\{\mathbf{Q}\right\}\right)^{1/2} \\ =& \sigma_\nu\sqrt{c_2}\left(\text{tr}\left\{\mathbf{R}^{-1}\right\}\right) \\ =& \sigma_\nu\sqrt{c_2}\frac{r_{11}+r_{22}}{r_{11}r_{22}-r_{21}^2}\end{aligned} \tag{17}$$

$$\begin{aligned}\lambda_{\text{opt}} =& 1-\frac{1}{\sigma_\nu}\left(\frac{\text{tr}\left\{\mathbf{Q}\right\}}{\text{tr}\left\{\mathbf{R}^{-1}\right\}}\right)^{1/2} \\ =& 1-\frac{\sqrt{c_2}}{\sigma_\nu}\sqrt{r_{11}r_{22}-r21^2}\end{aligned} \tag{18}$$

(iv)

$$\begin{aligned}\mathscr{M}_{\text{min}}^{\text{RLS}} =& \frac{1}{\sigma_\nu}\left(M\text{tr}\left\{\mathbf{RQ}\right\}\right)^{1/2}, \qquad M=2 \\ =& \frac{2}{\sigma_\nu}\sqrt{c_2}\end{aligned} \tag{19}$$

$$\begin{aligned}\lambda_{\text{opt}} =& 1 - \frac{1}{\sigma_\nu}\left(\frac{1}{M}\text{tr}\left\{c_2\mathbf{I}\right\}\right), \qquad M = 2\\ =& 1 - \frac{\sqrt{c_2}}{\sigma_\nu}\end{aligned} \tag{20}$$

which is differnt from the λ_{opt} for minimum $\mathscr{D}_{\text{min}}^{\text{RLS}}$.

Comparisons of LMS and RLS algorithms:

1. $\mathbf{Q} = c_1\mathbf{R}$

$$\frac{\mathscr{D}_{\text{min}}^{\text{LMS}}}{\mathscr{D}_{\text{min}}^{\text{RLS}}} = \frac{2\sqrt{r_{11}r_{22} - r_{21}^2}}{r_{11} + r_{22}}$$

$$\frac{\mathscr{M}_{\text{min}}^{\text{LMS}}}{\mathscr{M}_{\text{min}}^{\text{RLS}}} = \frac{r_{11} + r_{22}}{\sqrt{2}\sqrt{r_{11} + 2r_{21}^2 + r_{22}^2}}$$

2. $\mathbf{Q} = c_2\mathbf{R}^{-1}$

$$\frac{\mathscr{D}_{\text{min}}^{\text{LMS}}}{\mathscr{D}_{\text{min}}^{\text{RLS}}} = \sqrt{2}\frac{\sqrt{r_{11}^2 + 2r_{21}^2 + r_{22}^2}}{r_{11} + r_{22}}$$

$$\frac{\mathscr{M}_{\text{min}}^{\text{LMS}}}{\mathscr{M}_{\text{min}}^{\text{RLS}}} = \frac{r_{11} + r_{22}}{2\sqrt{r_{11}r_{22} - r_{21}^2}}$$

Problem 13.9

a)

```
# IDBD
kappa = 0.001
T = 100
period = 10
wstar = -1*ones(T+1)
u = zeros(T)
y = zeros(T)
w = zeros(T+1)
mu = ones(T+1)*0.01
h = 0
for n in range(T):
  u[n] = 10
```

```
  wstar[n+1] = wstar[n]
  if n%period==0:
    wstar[n+1] = -wstar[n]
  y[n] = wstar[n+1]*u[n] + randn()
  error = (y[n] - w[n]*u[n])
  mu[n+1] = mu[n]*exp(kappa*error*u[n]*h)
  w[n+1] = w[n] + mu[n+1]*error*u[n]
  h = h*(1-mu[n+1]*u[n]*u[n]) + mu[n+1]*error*u[n]
figure()
plot(wstar)
plot(w)
figure()
plot(mu)
```

b.i)

There are two stochastic gradient-descent updates involved in the IDBD, namely the update of weights and the update of the step-size parameter. In order that these two updates do not diverge, the step-size parameter for the first update and the meta-step-size parameter for the second update are small enough:

- How small the value of the step-size parameter, $\boldsymbol{\mu}$ should be, it depends on the scale of the magnitude of the input. $\mathbf{u}$.
- How small the value of meta-step-size parameter, $\boldsymbol{\kappa}$, needs to be, it depends on the scale of the magnitude of the target $\mathbf{y}$.

b.ii)

In order to mitigate divergence of both the initial step-size and the meta-step-size should be small.

Problem 13.10

```
# Autostep
kappa = 0.01
gamma = 0.0001
T = 10000
period = 10
wstar = -1*ones(T+1)
u = zeros(T)
y = zeros(T)
w = zeros(T+1)
mu = ones(T+1)*0.1
```

```
h = 0
v = zeros(T+1)
for n in range(T):
  u[n] = 10
  wstar[n+1] = wstar[n]
  if n%period==0:
   wstar[n+1] = -wstar[n]
  y[n] = wstar[n+1]*u[n] + randn()
  error = (y[n] - w[n]*u[n])
  v[n+1] = max(abs(error*u[n]*h), v[n] + gamma*mu[n]*u[n]*u[n]*(abs(error*u[n]*h)- v[n]
  if v[n+1] != 0:
    mu[n+1] = mu[n]*exp(kappa*error*u[n]*h/v[n+1])
  else:
    mu[n+1] = mu[n]
  M = max(1, mu[n+1]*u[n]*u[n])
  mu[n+1] = mu[n+1]/M
  w[n+1] = w[n] + mu[n+1]*error*u[n]
  h = h*(1-mu[n+1]*u[n]*u[n]) + mu[n+1]*error*u[n]

figure()
plot(wstar[10:])
plot(w[10:])
figure()
plot(mu[10:])
figure()
plot(v)
```

a)

Change the input **u** or the variance of the white noise to the values for which the IDBD algorithm diverged in **9 b)**. try even more extreme values. Next, the Autostep method, however, does not diverge for such values. Empirically, it has not yet been found for which case Autostep can diverge.

b)

The main difference between the Autostep method and the IDBD algorithm is that the Autostep method automatically protects the weight update and the step-size update from growing large in magnitude, so that the updates do not diverge. To this end, the Autostep method, takes care of the two issues mentioned in **9 b)**. Details of how these two issues are taken care of are, given in the text and in Mahmood et al. (2012). In short, the Autostep method scales both the step-size parameter and the effective meta-step-size parameter according to the scale of the magnitude of inputs and target outputs. It is done incrementally and online, and no prior knowledge about the scale of the magnitude of them is required.

Problem 13.11

a)

The step-size update is of a multiplicative kind. If the sequence of data is such that the multiplicative update always reduces the step-size parameter, even then the step-size value will go to zero only when time goes to infinity.

b)

However, due to finite precisions of computers, step-size value can become smaller than the smallest value representable, hereby causing the step size to be zero in finite time. This is problematic, because the step size will remain to be zero from then on and no further learning can occur.

c)

Accordingly, it is typical to use a lower bound for the step-size parameter to prevent the occurrence of such situations. This can be easily accomplished by taking the maximum between the multiplicative step-size update and a small value such as 1.0×10^{-20}.

d)

Underflow in other parameters is not crucial because no other updates are multiplicative. However, if overflow occurs, it will stop the algorithm from working. It can happen if the algorithm diverges.

Problem 13.12

```
'''
Created on Jan 21, 2012

@author: ashique
'''

import sys
sys.path.append("../../")

from numpy import *
from numpy.random import *
#from matplotlib.pyplot import *
from environments.stationaryprob import *
from algorithms.autostep import *
```

```
from algorithms.bmp import *
from algorithms.lms import *
from algorithms.rls import *
from algorithms.idbd import *
import os
from time import *

display = True

def main():
    M = 20
    N = 25000
    to = 0
    nofuls = M/4
    nruns = 50
    sigma2_u2 = 1.0
    sigma2_v = 1.0
    if len(sys.argv)>1:
        sigma2_u1 = double(sys.argv[1])
    else:
        sigma2_u1 = 1.0
    alphas = array([10**-7,     10**-6, 10**-5, 10**-4, 2*10**-4,\
      5*10**-4, 10**-3, 2*10**-3, 5*10**-3, 10**-2, 2*10**-2,\
      5*10**-2, 10**-1])
    mses = zeros((len(alphas), nruns))
    savedir = "../../../data/lms/"
    dirname = "prob_trans.inputs.sigma2_u1."+   str(sigma2_u1)+\
      ".sigma2_u2."+str(sigma2_u2)+".sigma2_v."+str(sigma2_v)+\
      ".m."+str(M)+".nofuls."+str(nofuls)+".N."+str(N)
    if not os.path.exists(savedir + dirname):
        os.makedirs(savedir + dirname)
    savetxt(savedir + dirname + "/stepsizes.txt", alphas)
    for run in range(nruns):
        for alpha in alphas:
            t = clock()
            seed(1)
            prob = StationaryProb(sigma2_u1=sigma2_u1,\
        sigma2_u2=sigma2_u2, sigma2_v=sigma2_v, n=M,\
        nofuls=nofuls)
            lms = LMS(w0 = zeros(M), alpha = alpha)
            w = zeros((M, N))
            mse = 0
            dev = 0
            seed(run)
            for n in range(N):
                ret = prob.step()
```

```
                d = ret["d"]
                u = ret["u"]
                ret = lms.step(yt = d, xt = u)
                pred = ret["pred"]
                w[:, n] = ret["w"]
                error = d - pred
                if (n>=to):
                    mse = mse + 1.0*(error**2 - mse)/(n+1-to)

            mses[alpha==alphas, run==array(range(nruns))] = mse
            print "run" + str(run) + ": " + str(clock() - t) + "sec"
        savetxt(savedir + dirname+"/run"+str(run)+".txt",\
         sqrt(mses[:,run]))
    print sqrt(mean(mses, 1))
    savetxt(savedir + dirname+"/avg_rmse.txt", sqrt(mean(mses, 1)))
    if display==True:
        plot(alphas, sqrt(mean(mses, 1)))
        title("lms")
        xscale("log")
        ylim([0, 20])

if __name__ == '__main__':
    main()
    if display==True:
        show()



'''
Created on Jan 21, 2012

@author: ashique
'''

import sys
sys.path.append("../../")

from numpy import *
from numpy.random import *
#from matplotlib.pyplot import *
from environments.stationaryprob import *
from algorithms.autostep import *
from algorithms.bmp import *
from algorithms.lms import *
from algorithms.rls import *
from algorithms.idbd import *
```

```
import os
from time import *

display = True

def main():
    M = 20
    N = 25000
    to = 0
    nofu1s = M/4
    nruns = 50
    sigma2_u2 = 1.0
    sigma2_v = 1.0
    if len(sys.argv)>1:
        sigma2_u1 = double(sys.argv[1])
    else:
        sigma2_u1 = 1.0
    gamas = array([10**-4, 10**-3, 2*10**-3, 5*10**-3, 10**-2,\
      2*10**-2, 5*10**-2, 10**-1, 2*10**-1, 5*10**-1, 9*10**-1])
    mses = zeros((len(gamas), nruns))
    savedir = "../../../data/rls/"
    dirname = "prob_trans.inputs.sigma2_u1."+str(sigma2_u1)+\
    ".sigma2_u2."+str(sigma2_u2)+".sigma2_v."+str(sigma2_v)+\
    ".m."+str(M)+".nofu1s."+str(nofu1s)+".N."+str(N)
    if not os.path.exists(savedir + dirname):
        os.makedirs(savedir + dirname)
    savetxt(savedir + dirname + "/one_minus_gammas.txt", gamas)
    for run in range(nruns):
        for gama in gamas:
            t = clock()
            seed(1)
            prob = StationaryProb(sigma2_u1=sigma2_u1,
         sigma2_u2=sigma2_u2, sigma2_v=sigma2_v,
         n=M, nofu1s=nofu1s)
            rls = RLS(w0=zeros(M), gama=1-gama)
            w = zeros((M, N))
            mse = 0
            dev = 0
            seed(run)
            for n in range(N):
                ret = prob.step()
                d = ret["d"]
                u = ret["u"]
                ret = rls.step(yt = d, xt = u)
                pred = ret["pred"]
                w[:, n] = ret["w"]
```

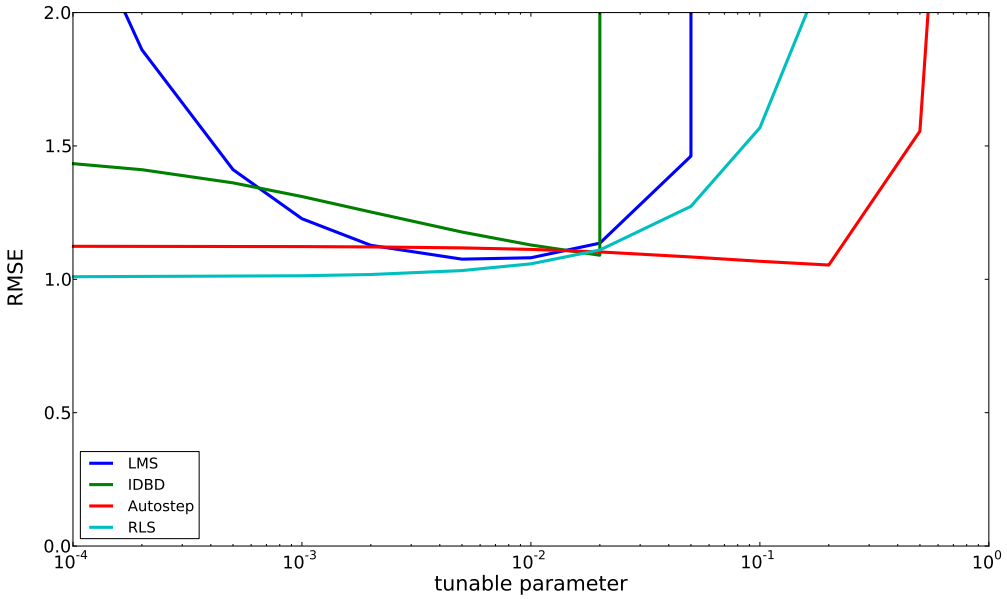
```
                error = d - pred
                if (n>=to):
                    mse = mse + 1.0*(error**2 - mse)/(n+1-to)

            mses[gama==gamas, run==array(range(nruns))] = mse
            print "run" + str(run) + ": " + str(clock() - t) + "sec"
        savetxt(savedir + dirname+"/run"+str(run)+".txt",\
     sqrt(mses[:,run]))
    print sqrt(mean(mses, 1))
    savetxt(savedir + dirname+"/avg_rmse.txt", sqrt(mean(mses, 1)))
    if display==True:
        plot(gamas, sqrt(mean(mses, 1)))
        title("rls")
        #xscale("log")
        ylim([0, 20])

if __name__ == '__main__':
    main()
    if display==True:
        show()



'''
Created on Jan 21, 2012

@author: ashique
'''

import sys
sys.path.append("../../")

from numpy import *
from numpy.random import *
#from matplotlib.pyplot import *
from environments.stationaryprob import *
from algorithms.autostep import *
from algorithms.bmp import *
from algorithms.lms import *
from algorithms.rls import *
from algorithms.idbd import *
import os
from time import *

display = True
```

```
def main():
    M = 20
    N = 25000
    to = 0
    nofu1s = M/4
    nruns = 50
    sigma2_u2 = 1.0
    sigma2_v = 1.0
    if len(sys.argv)>1:
        sigma2_u1 = double(sys.argv[1])
    else:
        sigma2_u1 = 1.0
    mus = array([10**-11, 10**-9, 3*10**-9, 10**-8, 3*10**-8,\
    10**-7, 2*10**-7, 5*10**-7, 10**-6, 2*10**-6, 5*10**-6,\
    10**-5, 2*10**-5, 5*10**-5, 10**-4, 2*10**-4, 5*10**-4,\
    10**-3, 2*10**-3, 5*10**-3, 10**-2, 2*10**-2, 5*10**-2,\
    10**-1])
    mses = zeros((len(mus), nruns))
    savedir = "../../../data/idbd/"
    dirname = "prob_trans.inputs.sigma2_u1."+str(sigma2_u1)+\
    ".sigma2_u2."+str(sigma2_u2)+".sigma2_v."+str(sigma2_v)+\
    ".m."+str(M)+".nofu1s."+str(nofu1s)+".N."+str(N)
    if not os.path.exists(savedir + dirname):
        os.makedirs(savedir + dirname)
    savetxt(savedir + dirname + "/metastepsizes.txt", mus)
    for run in range(nruns):
        for mu in mus:
            t = clock()
            seed(1)
            prob = StationaryProb(sigma2_u1=sigma2_u1, sigma2_u2\
            =sigma2_u2, sigma2_v=sigma2_v, n=M,\
            nofu1s=nofu1s)
            idbd = IDBD(w0=zeros(M), alphainit=1.0/(M*sigma2_u1)\
            , theta=mu)
            w = zeros((M, N))
            mse = 0
            dev = 0
            seed(run)
            for n in range(N):
                ret = prob.step()
                d = ret["d"]
                u = ret["u"]
                ret = idbd.step(yt = d, xt = u)
                pred = ret["pred"]
                w[:, n] = ret["w"]
```

```
                error = d - pred
                if (n>=to):
                    mse = mse + 1.0*(error**2 - mse)/(n+1-to)

            mses[mu==mus, run==array(range(nruns))] = mse
            print "run" + str(run) + ": " + str(clock() - t) + "sec"
        savetxt(savedir + dirname+"/run"+str(run)+".txt",\
        sqrt(mses[:,run]))
    print sqrt(mean(mses, 1))
    savetxt(savedir + dirname+"/avg_rmse.txt", sqrt(mean(mses, 1)))
    if display==True:
        plot(mus, sqrt(mean(mses, 1)))
        title("idbd")
        xscale("log")
        ylim([0, 20])

if __name__ == '__main__':
    main()
    if display==True:
        show()



'''
Created on Jan 21, 2012

@author: ashique
'''

import sys
sys.path.append("../../")

from numpy import *
from numpy.random import *
#from matplotlib.pyplot import *
from environments.stationaryprob import *
from algorithms.autostep import *
from algorithms.bmp import *
from algorithms.lms import *
from algorithms.rls import *
from algorithms.idbd import *
import os
from time import *

display = True
```

```
def main():
    M = 20
    N = 25000
    to = 0
    nofu1s = M/4
    nruns = 50
    sigma2_u2 = 1.0
    sigma2_v = 1.0
    if len(sys.argv)>1:
        sigma2_u1 = double(sys.argv[1])
    else:
        sigma2_u1 = 1.0
    mus = array([10**-12, 10**-7, 10**-4, 10**-3, 2*10**-3, 5*10**-3,\
        10**-2, 2*10**-2, 5*10**-2, 10**-1, 2*10**-1, 5*10**-1, 1.0])
    mses = zeros((len(mus), nruns))
    savedir = "../../../data/autostep/"
    dirname = "prob_trans.inputs.sigma2_u1."+str(sigma2_u1)+\
       ".sigma2_u2."+str(sigma2_u2)+".sigma2_v."+str(sigma2_v)\
       +".m."+str(M)+".nofu1s."+str(nofu1s)+".N."+str(N)
    if not os.path.exists(savedir + dirname):
        os.makedirs(savedir + dirname)
    savetxt(savedir + dirname + "/metastepsizes.txt", mus)
    for run in range(nruns):
        for mu in mus:
            t = clock()
            seed(1)
            prob = StationaryProb(sigma2_u1=sigma2_u1, \
                sigma2_u2=sigma2_u2, sigma2_v=sigma2_v,\
                n=M, nofu1s=nofu1s)
            autostep = Autostep(w0=zeros(M), lmbda=10**-4,\
                        mu=mu, alpha=1.0/(M*sigma2_u1))
            w = zeros((M, N))
            mse = 0
            dev = 0
            seed(run)
            for n in range(N):
                ret = prob.step()
                d = ret["d"]
                u = ret["u"]
                ret = autostep.step(yt = d, xt = u)
                pred = ret["pred"]
                w[:, n] = ret["w"]
                error = d - pred
                if (n>=to):
                    mse = mse + 1.0*(error**2 - mse)/(n+1-to)
```

```
            mses[mu==mus, run==array(range(nruns))] = mse
            print "run" + str(run) + ": " + str(clock() - t) + "sec"
        savetxt(savedir + dirname+"/run"+str(run)+".txt",\
                        sqrt(mses[:,run]))
    print sqrt(mean(mses, 1))
    savetxt(savedir + dirname+"/avg_rmse.txt", sqrt(mean(mses, 1)))
    if display==True:
        plot(mus, sqrt(mean(mses, 1)))
        title("autostep")
        xscale("log")
        ylim([0, 20])

if __name__ == '__main__':
    main()
    if display==True:
        show()
```



Discussion: Upon having tested the LMS, RLS, IDBD and Autostep on the problem in the text.

The tunable parameters were varied from 1 to 10^{-12} with equal logarithmic intervals. These parameters are: the step size parameter μ for LMS, the meta-step-size parameter α for IDBD and Autostep, and the exponential weighting factor γ for RLS (figure shows results for $1-\gamma$). We have set time scale parameter τ of Autostep to 10^{-4} as recommended.

All the estimated weights of all methods are initialized at zero. The initial step size for BMP, IDBD and Autostep is set to 1/(M)=0.05.

Our measure of performance is RMSE, averaged over the whole 25000 samples. Here, we consider the transient performance instead of asymptotic performance because it is a stationary problem. In the asymptote, all methods are expected to find the solution and the difference would be small.

The figure above shows the performance of all the methods as RMSE vs. tunable parameter values. All of them perform almost similarly.

The table below shows the best RMSE for different methods.

	RMSE	parameters
RLS	≈ 1.01	$1 - \gamma \leq 10^{-3}$
Autostep	≈ 1.09	$\alpha = 0.02$
LMS	≈ 1.08	$\mu = 0.005 \approx 0.01$
IDBD	≈ 1.09	$\alpha = 0.02$

Table 1: Best is RLS and it achieves RMSE close to minimum (1.0). But others have similar performance.

Problem 13.13

```
'''
Created on Jan 21, 2012

@author: ashique
'''

import sys
sys.path.append("../../")

from numpy import *
from numpy.random import *
from matplotlib.pyplot import *
from environments.stationaryprob import *
from algorithms.autostep import *
from algorithms.bmp import *
from algorithms.lms import *
from algorithms.rls import *
from algorithms.idbd import *
import os
```

```
from time import *

display = True

def main():
    M = 20
    N = 25000
    to = N/2
    nofuls = M/4
    nruns = 2
    sigma2_u2 = 1.0
    sigma2_v = 1.0
    if len(sys.argv)>1:
        sigma2_u1 = double(sys.argv[1])
    else:
        sigma2_u1 = 100.0
    alphas = array([10**-7, 10**-6, 10**-5, 10**-4, 2*10**-4,\
    5*10**-4, 10**-3, 2*10**-3, 5*10**-3,\
    10**-2, 2*10**-2, 5*10**-2, 10**-1])
    mses = zeros((len(alphas), nruns))
    savedir = "../../../data/lms/"
    dirname = "prob.inputs.sigma2_u1."+str(sigma2_u1)+".sigma2_u2."\
    +str(sigma2_u2)+".sigma2_v."+str(sigma2_v)+".m."+str(M)\
    +".nofuls."+str(nofuls)+".N."+str(N)
    if not os.path.exists(savedir + dirname):
        os.makedirs(savedir + dirname)
    savetxt(savedir + dirname + "/stepsizes.txt", alphas)
    for run in range(nruns):
        seed(run)
        for alpha in alphas:
            t = clock()
            prob = StationaryProb(sigma2_u1=sigma2_u1, sigma2_u2\
            =sigma2_u2, sigma2_v=sigma2_v, n=M,\
            nofuls=nofuls)
            lms = LMS(w0 = zeros(M), alpha = alpha)
            w = zeros((M, N))
            mse = 0
            dev = 0
            for n in range(N):
                ret = prob.step()
                d = ret["d"]
                u = ret["u"]
                ret = lms.step(yt = d, xt = u)
                pred = ret["pred"]
                w[:, n] = ret["w"]
                error = d - pred
```

```
                if (n>=to):
                    mse = mse + 1.0*(error**2 - mse)/(n+1-to)

            mses[alpha==alphas, run==array(range(nruns))] = mse
            print "run" + str(run) + ": " + str(clock() - t) + "sec"
        savetxt(savedir + dirname+"/run"+str(run)+".txt",\
        sqrt(mses[:,run]))
    print sqrt(mean(mses, 1))
    savetxt(savedir + dirname+"/avg_rmse.txt", sqrt(mean(mses, 1)))
    if display==True:
        plot(alphas, sqrt(mean(mses, 1)))
        title("lms")
        xscale("log")
        ylim([0, 20])

if __name__ == '__main__':
    main()
    if display==True:
        show()



'''
Created on Jan 21, 2012

@author: ashique
'''

import sys
sys.path.append("../../")

from numpy import *
from numpy.random import *
#from matplotlib.pyplot import *
from environments.stationaryprob import *
from algorithms.autostep import *
from algorithms.bmp import *
from algorithms.lms import *
from algorithms.rls import *
from algorithms.idbd import *
import os
from time import *

display = True

def main():
```

```
M = 20
N = 25000
to = N/2
nofu1s = M/4
nruns = 50
sigma2_u2 = 1.0
sigma2_v = 1.0
if len(sys.argv)>1:
    sigma2_u1 = double(sys.argv[1])
else:
    sigma2_u1 = 100.0
gamas = array([10**-4, 10**-3, 2*10**-3, 5*10**-3, 10**-2,\
2*10**-2, 5*10**-2, 10**-1, 2*10**-1, 5*10**-1, 9*10**-1])
mses = zeros((len(gamas), nruns))
savedir = "../../../data/rls/"
dirname = "prob.inputs.sigma2_u1."+str(sigma2_u1)+".sigma2_u2."\
+str(sigma2_u2)+".sigma2_v."+str(sigma2_v)+".m."+str(M)\
+".nofu1s."+str(nofu1s)+".N."+str(N)
if not os.path.exists(savedir + dirname):
    os.makedirs(savedir + dirname)
savetxt(savedir + dirname + "/one_minus_gammas.txt", gamas)
for run in range(nruns):
    seed(run)
    for gama in gamas:
        t = clock()
        prob = StationaryProb(sigma2_u1=sigma2_u1,\
        sigma2_u2=sigma2_u2, sigma2_v=sigma2_v,\
        n=M, nofu1s=nofu1s)
        rls = RLS(w0=zeros(M), gama=1-gama)
        w = zeros((M, N))
        mse = 0
        dev = 0
        for n in range(N):
            ret = prob.step()
            d = ret["d"]
            u = ret["u"]
            ret = rls.step(yt = d, xt = u)
            pred = ret["pred"]
            w[:, n] = ret["w"]
            error = d - pred
            if (n>=to):
                mse = mse + 1.0*(error**2 - mse)/(n+1-to)

        mses[gama==gamas, run==array(range(nruns))] = mse
        print "run" + str(run) + ": " + str(clock() - t) + "sec"
    savetxt(savedir + dirname+"/run"+str(run)+".txt",\
```

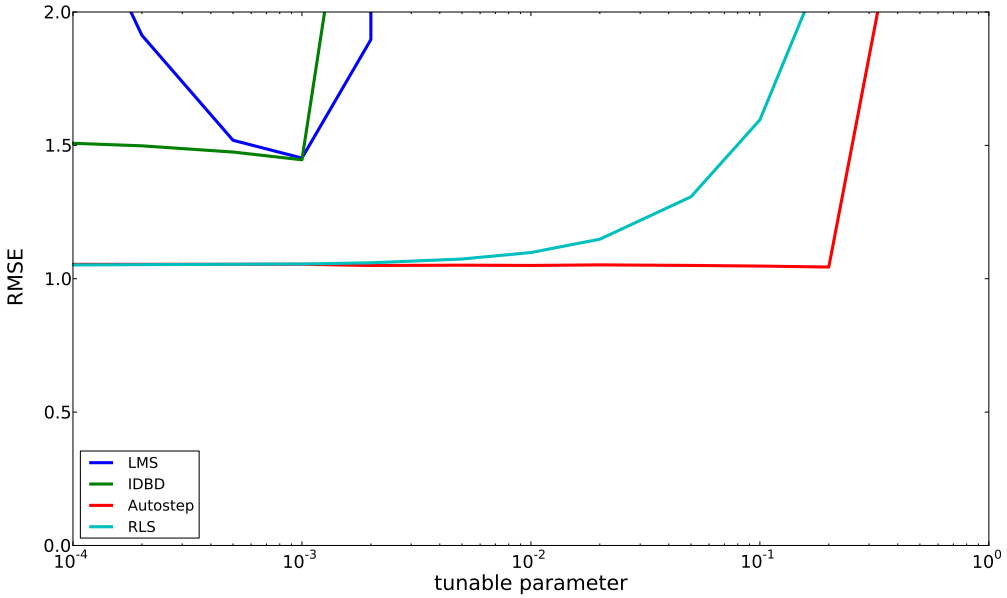
```
            sqrt(mses[:,run]))
        print sqrt(mean(mses, 1))
        savetxt(savedir + dirname+"/avg_rmse.txt", sqrt(mean(mses, 1)))
        if display==True:
            plot(gamas, sqrt(mean(mses, 1)))
            title("rls")
            xscale("log")
            ylim([0, 20])

if __name__ == '__main__':
    main()
    if display==True:
        show()



'''
Created on Jan 21, 2012

@author: ashique
'''

import sys
sys.path.append("../../")

from numpy import *
from numpy.random import *
#from matplotlib.pyplot import *
from environments.stationaryprob import *
from algorithms.autostep import *
from algorithms.bmp import *
from algorithms.lms import *
from algorithms.rls import *
from algorithms.idbd import *
import os
from time import *

display = True


def main():
    M = 20
    N = 25000
    to = N/2
    nofuls = M/4
    nruns = 50
```

```
sigma2_u2 = 1.0
sigma2_v = 1.0
if len(sys.argv)>1:
    sigma2_u1 = double(sys.argv[1])
else:
    sigma2_u1 = 100.0
mus = array([10**-11, 10**-9, 3*10**-9, 10**-8, 3*10**-8,\
10**-7, 2*10**-7, 5*10**-7, 10**-6, 2*10**-6, 5*10**-6,\
10**-5, 2*10**-5, 5*10**-5, 10**-4, 2*10**-4, 5*10**-4,\
10**-3, 2*10**-3, 5*10**-3, 10**-2, 2*10**-2, 5*10**-2, 10**-1])
mses = zeros((len(mus), nruns))
savedir = "../../../data/idbd/"
dirname = "prob.inputs.sigma2_u1."+str(sigma2_u1)+".sigma2_u2."\
+str(sigma2_u2)+".sigma2_v."+str(sigma2_v)+".m."+str(M)\
+".nofu1s."+str(nofu1s)+".N."+str(N)
if not os.path.exists(savedir + dirname):
    os.makedirs(savedir + dirname)
savetxt(savedir + dirname + "/metastepsizes.txt", mus)
for run in range(nruns):
    seed(run)
    for mu in mus:
        t = clock()
        prob = StationaryProb(sigma2_u1=sigma2_u1, sigma2_u2=\
        sigma2_u2, sigma2_v=sigma2_v, n=M,\
        nofu1s=nofu1s)
        idbd = IDBD(w0=zeros(M), alphainit=1.0/(M*sigma2_u1),\
        theta=mu)
        w = zeros((M, N))
        mse = 0
        dev = 0
        for n in range(N):
            ret = prob.step()
            d = ret["d"]
            u = ret["u"]
            ret = idbd.step(yt = d, xt = u)
            pred = ret["pred"]
            w[:, n] = ret["w"]
            error = d - pred
            if (n>=to):
                mse = mse + 1.0*(error**2 - mse)/(n+1-to)

        mses[mu==mus, run==array(range(nruns))] = mse
        print "run" + str(run) + ": " + str(clock() - t) + "sec"
    savetxt(savedir + dirname+"/run"+str(run)+".txt",\
    sqrt(mses[:,run]))
print sqrt(mean(mses, 1))
```

```
    savetxt(savedir + dirname+"/avg_rmse.txt", sqrt(mean(mses, 1)))
    if display==True:
        plot(mus, sqrt(mean(mses, 1)))
        title("idbd")
        xscale("log")
        ylim([0, 20])

if __name__ == '__main__':
    main()
    if display==True:
        show()



'''
Created on Jan 21, 2012

@author: ashique
'''

import sys
sys.path.append("../../")

from numpy import *
from numpy.random import *
from environments.stationaryprob import *
from algorithms.autostep import *
from algorithms.bmp import *
from algorithms.lms import *
from algorithms.rls import *
from algorithms.idbd import *
import os
from time import *

display = True

def main():
    M = 20
    N = 25000
    to = N/2
    nofuls = M/4
    nruns = 50
    sigma2_u2 = 1.0
    sigma2_v = 1.0
    if len(sys.argv)>1:
        sigma2_u1 = double(sys.argv[1])
```

```
else:
    sigma2_u1 = 100.0
mus = array([10**-12, 10**-7, 10**-4, 10**-3, 2*10**-3, 5*10**-3,\
10**-2, 2*10**-2, 5*10**-2, 10**-1, 2*10**-1, 5*10**-1, 1.0])
mses = zeros((len(mus), nruns))
savedir = "../../../data/autostep/"
dirname = "prob.inputs.sigma2_u1."+str(sigma2_u1)+".sigma2_u2."\
+str(sigma2_u2)+".sigma2_v."+str(sigma2_v)+".m."+str(M)\
+".nofu1s."+str(nofu1s)+".N."+str(N)
if not os.path.exists(savedir + dirname):
    os.makedirs(savedir + dirname)
savetxt(savedir + dirname + "/metastepsizes.txt", mus)
for run in range(nruns):
    seed(run)
    for mu in mus:
        t = clock()
        prob = StationaryProb(sigma2_u1=sigma2_u1, sigma2_u2=\
        sigma2_u2, sigma2_v=sigma2_v, n=M, nofu1s=nofu1s)
        autostep = Autostep(w0=zeros(M), lmbda=10**-4, mu=mu,\
        alpha=1.0/(M*sigma2_u1))
        w = zeros((M, N))
        mse = 0
        dev = 0
        for n in range(N):
            ret = prob.step()
            d = ret["d"]
            u = ret["u"]
            ret = autostep.step(yt = d, xt = u)
            pred = ret["pred"]
            w[:, n] = ret["w"]
            error = d - pred
            if (n>=to):
                mse = mse + 1.0*(error**2 - mse)/(n+1-to)

        mses[mu==mus, run==array(range(nruns))] = mse
        print "run" + str(run) + ": " + str(clock() - t) + "sec"
    savetxt(savedir + dirname+"/run"+str(run)+".txt",\
    sqrt(mses[:,run]))
print sqrt(mean(mses, 1))
savetxt(savedir + dirname+"/avg_rmse.txt", sqrt(mean(mses, 1)))
if display==True:
    plot(mus, sqrt(mean(mses, 1)))
    title("autostep")
    xscale("log")
    ylim([0, 20])
```

```
if __name__ == '__main__':
    main()
    if display==True:
        show()
```



Discussion: Upon having tested the LMS, RLS, IDBD and Autostep on the problem in the text. The tunable parameters were varied from 1 to 10^{-12} with equal logarithmic intervals. These parameters are: the step size parameter μ for LMS, the meta-step-size parameter α for IDBD and Autostep, and the exponential weighting factor γ for RLS (figure shows results for $1-\gamma$). We have set time scale parameter τ of Autostep to 10^{-4} as recommended. All the estimated weights of all methods are initialized at zero. The initial step size for BMP, IDBD and Autostep is set to 1/(M)=0.0005.

Our measure of performance is RMSE, averaged over the whole 25000 samples. Here, we consider the transient performance instead of asymptotic performance because it is a stationary problem. In the asymptote, all methods are expected to find the solution and the difference would be small.

The figure above shows the performance of all the methods as RMSE vs. tunable parameter values. All of them performs almost similarly.

The table below shows the best RMSE for different methods.

	RMSE	parameters
RLS	≈ 1.05	$1 - \gamma \leq 10^{-3}$
Autostep	≈ 1.05	$\alpha = 0.2$
LMS	≈ 1.45	$\mu = 0.001$
IDBD	as above	$\alpha = 0.001$

Table 1: Best is RLS and it achieves RMSE close to minimum (1.0).

But the other algorithms have similar performance: RLS performed the best and Autostep the second best.

Note that, the best parameters for all the methods except autostep are different by orders of magnitude than the other results.

Also note that, if the variance of the different input elements was the same, then the objective function would have been spherical and all the methods would have essentially followed gradient descent directions. Hence, a vector step size would not be beneficial and the best performances, in that case, would be similar for all the first order methods. Our results on such cases confirm that. However, Autostep would still be preferable because of not requiring any tuning of its parameters. The other methods achieve the same performance as Autostep only after tuning their parameters.

Problem 13.14

```
'''
Created on Jan 21, 2012

@author: ashique
'''

import sys
sys.path.append("../../")

from numpy import *
from numpy.random import *
#from matplotlib.pyplot import *
from environments.stationarycorrprob import *
from algorithms.autostep import *
from algorithms.bmp import *
from algorithms.lms import *
from algorithms.rls import *
from algorithms.idbd import *
```

```
import os
from time import *

display = True

def main():
    M = 20
    N = 25000
    to = 0
    nofuls = M/4
    nruns = 50
    sigma2_u2 = 1.0
    sigma2_v = 1.0
    if len(sys.argv)>1:
        sigma2_u1 = double(sys.argv[1])
    else:
        sigma2_u1 = 1.0
    alphas = array([10**-7, 10**-6, 10**-5, 10**-4, 2*10**-4,\
    5*10**-4,10**-3, 2*10**-3, 5*10**-3, 10**-2, 2*10**-2,\
    5*10**-2,10**-1])
    mses = zeros((len(alphas), nruns))
    savedir = "../../../data/lms/"
    dirname = "prob_trans.corrinputs.sigma2_u1."+str(sigma2_u1)\
    +".sigma2_u2."+str(sigma2_u2)+".sigma2_v."\
    +str(sigma2_v)+".m."+str(M)+".nofuls."\
    +str(nofuls)+".N."+str(N)
    if not os.path.exists(savedir + dirname):
        os.makedirs(savedir + dirname)
    savetxt(savedir + dirname + "/stepsizes.txt", alphas)
    seed(1)
    prob = StationaryCorrProb(sigma2_u1=sigma2_u1, sigma2_u2=\
    sigma2_u2, sigma2_v=sigma2_v, n=M,\
    nofuls=nofuls)
    savetxt(savedir + dirname + "/covmatrix.txt", prob.getcovm())
    for run in range(nruns):
        for alpha in alphas:
            t = clock()
            lms = LMS(w0 = zeros(M), alpha = alpha)
            w = zeros((M, N))
            mse = 0
            dev = 0
            seed(run)
            for n in range(N):
                ret = prob.step()
                d = ret["d"]
                u = ret["u"]
```

```
                ret = lms.step(yt = d, xt = u)
                pred = ret["pred"]
                w[:, n] = ret["w"]
                error = d - pred
                if (n>=to):
                    mse = mse + 1.0*(error**2 - mse)/(n+1-to)

            mses[alpha==alphas, run==array(range(nruns))] = mse
            print "run" + str(run) + ": " + str(clock() - t) + "sec"
        savetxt(savedir + dirname+"/run"+str(run)+".txt",\
        sqrt(mses[:,run]))
    print sqrt(mean(mses, 1))
    savetxt(savedir + dirname+"/avg_rmse.txt", sqrt(mean(mses, 1)))
    if display==True:
        plot(alphas, sqrt(mean(mses, 1)))
        title("lms")
        xscale("log")
        ylim([0, 20])

if __name__ == '__main__':
    main()
    if display==True:
        show()



'''
Created on Jan 21, 2012

@author: ashique
'''

import sys
sys.path.append("../../")

from numpy import *
from numpy.random import *
#from matplotlib.pyplot import *
from environments.stationarycorrprob import *
from algorithms.autostep import *
from algorithms.bmp import *
from algorithms.lms import *
from algorithms.rls import *
from algorithms.idbd import *
import os
from time import *
```

```
display = True

def main():
    M = 20
    N = 25000
    to = 0
    nofu1s = M/4
    nruns = 50
    sigma2_u2 = 1.0
    sigma2_v = 1.0
    if len(sys.argv)>1:
        sigma2_u1 = double(sys.argv[1])
    else:
        sigma2_u1 = 1.0
    gamas = array([10**-4, 10**-3, 2*10**-3, 5*10**-3, 10**-2,\
    2*10**-2, 5*10**-2, 10**-1, 2*10**-1, 5*10**-1, 9*10**-1])
    mses = zeros((len(gamas), nruns))
    savedir = "../../../data/rls/"
    dirname = "prob_trans.corrinputs.sigma2_u1."+str(sigma2_u1)+\
    ".sigma2_u2."+str(sigma2_u2)+".sigma2_v."+str(sigma2_v)+\
    ".m."+str(M)+".nofu1s."+str(nofu1s)+".N."+str(N)
    if not os.path.exists(savedir + dirname):
        os.makedirs(savedir + dirname)
    savetxt(savedir + dirname + "/one_minus_gammas.txt", gamas)
    seed(1)
    prob = StationaryCorrProb(sigma2_u1=sigma2_u1, sigma2_u2=\
    sigma2_u2, sigma2_v=sigma2_v, n=M,\
    nofu1s=nofu1s)
    savetxt(savedir + dirname + "/covmatrix.txt", prob.getcovm())
    for run in range(nruns):
        for gama in gamas:
            t = clock()
            rls = RLS(w0=zeros(M), gama=1-gama)
            w = zeros((M, N))
            mse = 0
            dev = 0
            seed(run)
            for n in range(N):
                ret = prob.step()
                d = ret["d"]
                u = ret["u"]
                ret = rls.step(yt = d, xt = u)
                pred = ret["pred"]
                w[:, n] = ret["w"]
                error = d - pred
```

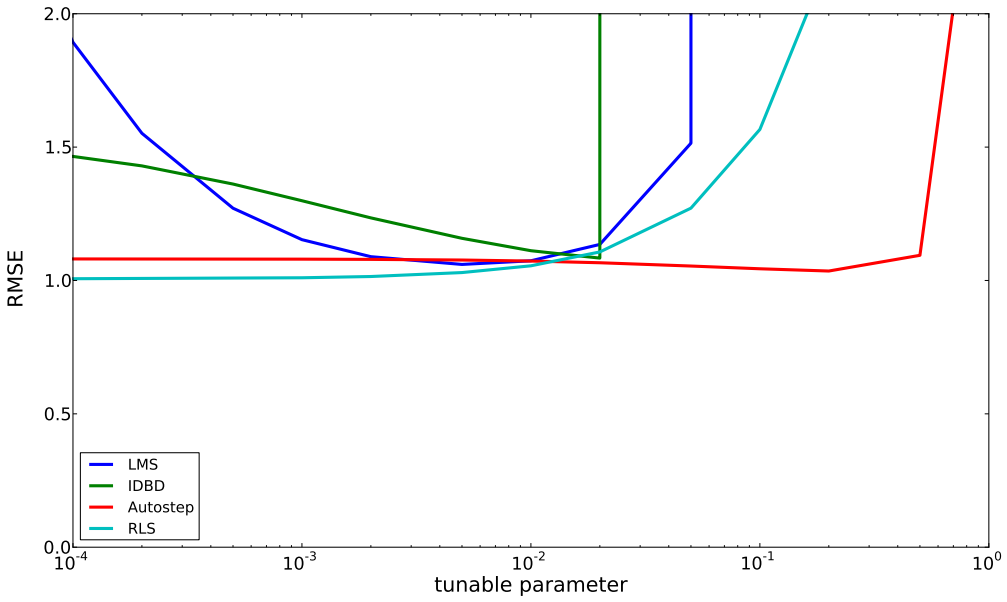
```
                if (n>=to):
                    mse = mse + 1.0*(error**2 - mse)/(n+1-to)

            mses[gama==gamas, run==array(range(nruns))] = mse
            print "run" + str(run) + ": " + str(clock() - t) + "sec"
        savetxt(savedir + dirname+"/run"+str(run)+".txt",\
        sqrt(mses[:,run]))
    print sqrt(mean(mses, 1))
    savetxt(savedir + dirname+"/avg_rmse.txt", sqrt(mean(mses, 1)))
    if display==True:
        plot(gamas, sqrt(mean(mses, 1)))
        title("rls")
        #xscale("log")
        ylim([0, 20])

if __name__ == '__main__':
    main()
    if display==True:
        show()



'''
Created on Jan 21, 2012

@author: ashique
'''

import sys
sys.path.append("../../")

from numpy import *
from numpy.random import *
#from matplotlib.pyplot import *
from environments.stationarycorrprob import *
from algorithms.autostep import *
from algorithms.bmp import *
from algorithms.lms import *
from algorithms.rls import *
from algorithms.idbd import *
import os
from time import *

display = True
```

```
def main():
    M = 20
    N = 25000
    to = 0
    nofu1s = M/4
    nruns = 50
    sigma2_u2 = 1.0
    sigma2_v = 1.0
    if len(sys.argv)>1:
        sigma2_u1 = double(sys.argv[1])
    else:
        sigma2_u1 = 1.0
    mus = array([10**-11, 10**-9, 3*10**-9, 10**-8, 3*10**-8,\
    10**-7, 2*10**-7, 5*10**-7, 10**-6, 2*10**-6, 5*10**-6, \
    10**-5, 2*10**-5, 5*10**-5, 10**-4, 2*10**-4, 5*10**-4, \
    10**-3, 2*10**-3, 5*10**-3, 10**-2, 2*10**-2, 5*10**-2, \
    10**-1])
    mses = zeros((len(mus), nruns))
    savedir = "../../../data/idbd/"
    dirname = "prob_trans.corrinputs.sigma2_u1."+str(sigma2_u1)+\
    ".sigma2_u2."+str(sigma2_u2)+".sigma2_v."+str(sigma2_v)\
    +".m."+str(M)+".nofu1s."+str(nofu1s)+".N."+str(N)
    if not os.path.exists(savedir + dirname):
        os.makedirs(savedir + dirname)
    savetxt(savedir + dirname + "/metastepsizes.txt", mus)
    seed(1)
    prob = StationaryCorrProb(sigma2_u1=sigma2_u1,\
    sigma2_u2=sigma2_u2, sigma2_v=sigma2_v, n=M,\
    nofu1s=nofu1s)
    savetxt(savedir + dirname + "/covmatrix.txt", prob.getcovm())
    for run in range(nruns):
        for mu in mus:
            t = clock()
            idbd = IDBD(w0=zeros(M), alphainit=1.0/(M*sigma2_u1),\
            theta=mu)
            w = zeros((M, N))
            mse = 0
            dev = 0
            seed(run)
            for n in range(N):
                ret = prob.step()
                d = ret["d"]
                u = ret["u"]
                ret = idbd.step(yt = d, xt = u)
                pred = ret["pred"]
                w[:, n] = ret["w"]
```

```
                error = d - pred
                if (n>=to):
                    mse = mse + 1.0*(error**2 - mse)/(n+1-to)

            mses[mu==mus, run==array(range(nruns))] = mse
            print "run" + str(run) + ": " + str(clock() - t) + "sec"
        savetxt(savedir + dirname+"/run"+str(run)+".txt",\
        sqrt(mses[:,run]))
    print sqrt(mean(mses, 1))
    savetxt(savedir + dirname+"/avg_rmse.txt", sqrt(mean(mses, 1)))
    if display==True:
        plot(mus, sqrt(mean(mses, 1)))
        title("idbd")
        xscale("log")
        ylim([0, 20])

if __name__ == '__main__':
    main()
    if display==True:
        show()



'''
Created on Jan 21, 2012

@author: ashique
'''

import sys
sys.path.append("../../")

from numpy import *
from numpy.random import *
#from matplotlib.pyplot import *
from environments.stationarycorrprob import *
from algorithms.autostep import *
from algorithms.bmp import *
from algorithms.lms import *
from algorithms.rls import *
from algorithms.idbd import *
import os
from time import *

display = True
```

```
def main():
    M = 20
    N = 25000
    to = 0
    nofu1s = M/4
    nruns = 50
    sigma2_u2 = 1.0
    sigma2_v = 1.0
    if len(sys.argv)>1:
        sigma2_u1 = double(sys.argv[1])
    else:
        sigma2_u1 = 1.0
    mus = array([10**-12, 10**-7, 10**-4, 10**-3, 2*10**-3,\
    5*10**-3, 10**-2, 2*10**-2, 5*10**-2,\
    10**-1, 2*10**-1, 5*10**-1, 1.0])
    mses = zeros((len(mus), nruns))
    savedir = "../../../data/autostep/"
    dirname = "prob_trans.corrinputs.sigma2_u1."+str(sigma2_u1)+\
    ".sigma2_u2."+str(sigma2_u2)+".sigma2_v."+\
    str(sigma2_v)+".m."+str(M)+".nofu1s."+\
    str(nofu1s)+".N."+str(N)
    if not os.path.exists(savedir + dirname):
        os.makedirs(savedir + dirname)
    savetxt(savedir + dirname + "/metastepsizes.txt", mus)
    seed(1)
    prob = StationaryCorrProb(sigma2_u1=sigma2_u1, sigma2_u2=\
    sigma2_u2, sigma2_v=sigma2_v, n=M,\
    nofu1s=nofu1s)
    savetxt(savedir + dirname + "/covmatrix.txt", prob.getcovm())
    for run in range(nruns):
        for mu in mus:
            t = clock()
            autostep = Autostep(w0=zeros(M), lmbda=10**-4, mu=mu,\
            alpha=1.0/(M*sigma2_u1))
            w = zeros((M, N))
            mse = 0
            dev = 0
            seed(run)
            for n in range(N):
                ret = prob.step()
                d = ret["d"]
                u = ret["u"]
                ret = autostep.step(yt = d, xt = u)
                pred = ret["pred"]
                w[:, n] = ret["w"]
                error = d - pred
```

```
                if (n>=to):
                    mse = mse + 1.0*(error**2 - mse)/(n+1-to)

            mses[mu==mus, run==array(range(nruns))] = mse
            print "run" + str(run) + ": " + str(clock() - t) + "sec"
        savetxt(savedir + dirname+"/run"+str(run)+".txt",\
        sqrt(mses[:,run]))
    print sqrt(mean(mses, 1))
    savetxt(savedir + dirname+"/avg_rmse.txt", sqrt(mean(mses, 1)))
    if display==True:
        plot(mus, sqrt(mean(mses, 1)))
        title("autostep")
        xscale("log")
        ylim([0, 20])

if __name__ == '__main__':
    main()
    if display==True:
        show()
```



Upon having tested the LMS, RLS, IDBD and Autostep on the problem in the text.

The tunable parameters were varied from 1 to 10^{-12} with equal logarithmic intervals. These parameters are: the step size parameter μ for LMS, the meta-step-size parameter

α for IDBD and Autostep, and the exponential weighting factor γ for RLS (figure shows results for $1-\gamma$). We have set time scale parameter τ of Autostep to 10^{-4} as recommended. All the estimated weights of all methods are initialized at zero. The initial step size for BMP, IDBD and Autostep is set to 1/(M)=0.05.

Our measure of performance is RMSE, vs. tunable parameter values. All of them performed almost identically. In fact, the more that the objective function becomes elliptical the greater the advantage is swayed towards the RLS algorithm. Apparently the objective function is not elliptical enough to adequately illustrate this phenomenon. It is possible that this is because the covariance matrix was generated randomly and as such the covariances between the input were mostly small.

Table 1 shows the best RMSE for different methods.

	RMSE	parameters
RLS	≈ 1.01	$1-\gamma \leq 10^{-3}$
Autostep	≈ 1.04	$\alpha = 0.2$
LMS	≈ 1.06	$\mu = 0.005$
IDBD	1.08	$\alpha = 0.02$

Table 1: Best is RLS and it achieves RMSE close to minimum (1.0). But others have similar performance.

Problem 13.15

```
'''
Created on Jan 21, 2012

@author: ashique
'''

import sys
sys.path.append("../../")

from numpy import *
from numpy.random import *
#from matplotlib.pyplot import *
from environments.nonstationary import *
from algorithms.autostep import *
from algorithms.bmp import *
from algorithms.lms import *
from algorithms.rls import *
```

```
from algorithms.idbd import *
import os
from time import *

display = True

def main():
    N = 50000
    M = 20
    o1s = M/4
    nruns = 50
    to = N/2
    sigma2_u = 1.0
    a = 0.9998
    sigma2_v = 1.0
    if len(sys.argv)>1:
        sigma2_o1 = double(sys.argv[1])
    else:
        sigma2_o1 = 100.0
    if len(sys.argv)>2:
        sigma2_o2 = double(sys.argv[2])
    else:
        sigma2_o2 = 1.0
    alphas = array([10**-7, 10**-6, 10**-5, 10**-4, 10**-3,\
    2*10**-3, 5*10**-3, 10**-2, 2*10**-2, 5*10**-2, 10**-1])
    mses = zeros((len(alphas), nruns))
    savedir = "../../../data/lms/"
    dirname = "prob.s2u."+str(sigma2_u)+".a."+str(a)+".s2v."\
    +str(sigma2_v)+".s2o1."+str(sigma2_o1)+".s2o1."\
    +str(sigma2_o2)+ ".m."+str(M)+".o1s."+str(o1s)+".N."+str(N)
    if not os.path.exists(savedir + dirname):
        os.makedirs(savedir + dirname)
    savetxt(savedir + dirname + "/stepsizes.txt", alphas)
    for run in range(nruns):
        for alpha in alphas:
            t = clock()
            seed(1)
            prob = Nonstationary(sigma2_u=sigma2_u, a=a,\
            sigma2_v=sigma2_v, sigma2_o1=sigma2_o1,\
            sigma2_o2=sigma2_o2, n=M, o1s=o1s)
            lms = LMS(w0 = zeros(M), alpha = alpha)
            w = zeros((M, N))
            mse = 0
            dev = 0
            seed(run)
            for n in range(N):
```

```
                ret = prob.step()
                d = ret["d"]
                u = ret["u"]
                ret = lms.step(yt = d, xt = u)
                pred = ret["pred"]
                w[:, n] = ret["w"]
                error = d - pred
                if (n>=to):
                    mse = mse + 1.0*(error**2 - mse)/(n+1-to)

            mses[alpha==alphas, run==array(range(nruns))] = mse
            print "run" + str(run) + ": " + str(clock() - t) + "sec"
        savetxt(savedir + dirname+"/run"+str(run)+".txt",\
        sqrt(mses[:,run]))
    print sqrt(mean(mses, 1))
    savetxt(savedir + dirname+"/avg_rmse.txt", sqrt(mean(mses, 1)))
    if display==True:
        plot(alphas, sqrt(mean(mses, 1)))
        title("lms")
        xscale("log")
        ylim([0, 20])

if __name__ == '__main__':
    main()
    if display==True:
        show()



'''
Created on Jan 21, 2012

@author: ashique
'''

import sys
sys.path.append("../../")

from numpy import *
from numpy.random import *
#from matplotlib.pyplot import *
from environments.nonstationary import *
from algorithms.autostep import *
from algorithms.bmp import *
from algorithms.lms import *
from algorithms.rls import *
```

```
from algorithms.idbd import *
import os
from time import *

display = True

def main():
    N = 50000
    M = 20
    o1s = M/4
    nruns = 50
    to = N/2
    sigma2_u = 1.0
    a = 0.9998
    sigma2_v = 1.0
    if len(sys.argv)>1:
        sigma2_o1 = double(sys.argv[1])
    else:
        sigma2_o1 = 100.0
    if len(sys.argv)>2:
        sigma2_o2 = double(sys.argv[2])
    else:
        sigma2_o2 = 1.0
    gamas = array([5*10**-3, 10**-2, 2*10**-2, 5*10**-2, 10**-1,\
    2*10**-1, 5*10**-1, 9*10**-1])
    mses = zeros((len(gamas), nruns))
    savedir = "../../../data/rls/"
    dirname = "prob.s2u."+str(sigma2_u)+".a."+str(a)+".s2v."+\
    str(sigma2_v)+".s2o1."+str(sigma2_o1)+".s2o1."+\
    str(sigma2_o2)+".m."+str(M)+".o1s."+str(o1s)+".N."+str(N)
    if not os.path.exists(savedir + dirname):
        os.makedirs(savedir + dirname)
    savetxt(savedir + dirname + "/one_minus_gammas.txt", gamas)
    for run in range(nruns):
        for gama in gamas:
            t = clock()
            seed(1)
            prob = Nonstationary(sigma2_u=sigma2_u, a=a,\
            sigma2_v=sigma2_v, sigma2_o1=sigma2_o1,\
            sigma2_o2=sigma2_o2, n=M, o1s=o1s)
            rls = RLS(w0=zeros(M), gama=1-gama)
            w = zeros((M, N))
            mse = 0
            dev = 0
            seed(run)
            for n in range(N):
```

```
                ret = prob.step()
                d = ret["d"]
                u = ret["u"]
                ret = rls.step(yt = d, xt = u)
                pred = ret["pred"]
                w[:, n] = ret["w"]
                error = d - pred
                if (n>=to):
                    mse = mse + 1.0*(error**2 - mse)/(n+1-to)

            mses[gama==gamas, run==array(range(nruns))] = mse
            print "run" + str(run) + ": " + str(clock() - t) + "sec"
        savetxt(savedir + dirname+"/run"+str(run)+".txt", \
        sqrt(mses[:,run]))
    print sqrt(mean(mses, 1))
    savetxt(savedir + dirname+"/avg_rmse.txt", sqrt(mean(mses, 1)))
    if display==True:
        plot(gamas, sqrt(mean(mses, 1)))
        title("rls")
        xscale("log")
        ylim([0, 20])

if __name__ == '__main__':
    main()
    if display==True:
        show()



'''
Created on Jan 21, 2012

@author: ashique
'''

import sys
sys.path.append("../../")

from numpy import *
from numpy.random import *
from matplotlib.pyplot import *
from environments.nonstationary import *
from algorithms.autostep import *
from algorithms.bmp import *
from algorithms.lms import *
from algorithms.rls import *
```

```
from algorithms.idbd import *
import os
from time import *

display = True


def main():
    N = 50000
    M = 20
    o1s = M/4
    nruns = 50
    to = N/2
    sigma2_u = 1.0
    a = 0.9998
    sigma2_v = 1.0
    if len(sys.argv)>1:
        sigma2_o1 = double(sys.argv[1])
    else:
        sigma2_o1 = 100.0
    if len(sys.argv)>2:
        sigma2_o2 = double(sys.argv[2])
    else:
        sigma2_o2 = 1.0
    mus = array([10**-11, 10**-9, 3*10**-9, 10**-8, 3*10**-8,\
    10**-7, 2*10**-7, 5*10**-7, 10**-6, 2*10**-6, 5*10**-6,\
    10**-5, 2*10**-5, 5*10**-5, 10**-4, 2*10**-4, 5*10**-4,\
    10**-3, 2*10**-3, 5*10**-3, 10**-2, 2*10**-2])
    mses = zeros((len(mus), nruns))
    savedir = "../../../data/idbd/"
    dirname = "prob.s2u."+str(sigma2_u)+".a."+str(a)+".s2v."+\
    str(sigma2_v)+".s2o1."+str(sigma2_o1)+".s2o1."+\
    str(sigma2_o2)+".m."+str(M)+".o1s."+str(o1s)+".N."+str(N)
    if not os.path.exists(savedir + dirname):
        os.makedirs(savedir + dirname)
    savetxt(savedir + dirname + "/metastepsizes.txt", mus)
    for run in range(nruns):
        for mu in mus:
            t = clock()
            seed(1)
            prob = Nonstationary(sigma2_u=sigma2_u, a=a,\
            sigma2_v=sigma2_v, sigma2_o1=sigma2_o1,\
            sigma2_o2=sigma2_o2, n=M, o1s=o1s)
            idbd = IDBD(w0=zeros(M), alphainit=1.0/M, theta=mu)
            w = zeros((M, N))
            mse = 0
```

```
            dev = 0
            seed(run)
            for n in range(N):
                ret = prob.step()
                d = ret["d"]
                u = ret["u"]
                ret = idbd.step(yt = d, xt = u)
                pred = ret["pred"]
                w[:, n] = ret["w"]
                error = d - pred
                if (n>=to):
                    mse = mse + 1.0*(error**2 - mse)/(n+1-to)

            mses[mu==mus, run==array(range(nruns))] = mse
            print "run" + str(run) + ": " + str(clock() - t) + "sec"
        savetxt(savedir + dirname+"/run"+str(run)+".txt",\
        sqrt(mses[:,run]))
    print sqrt(mean(mses, 1))
    savetxt(savedir + dirname+"/avg_rmse.txt", sqrt(mean(mses, 1)))
    if display==True:
        plot(mus, sqrt(mean(mses, 1)))
        title("idbd")
        xscale("log")
        ylim([0, 20])

if __name__ == '__main__':
    main()
    if display==True:
        show()



'''
Created on Jan 21, 2012

@author: ashique
'''

import sys
sys.path.append("../../")

from numpy import *
from numpy.random import *
#from matplotlib.pyplot import *
from environments.nonstationary import *
from algorithms.autostep import *
```

```
from algorithms.bmp import *
from algorithms.lms import *
from algorithms.rls import *
from algorithms.idbd import *
import os
from time import *

display = True

def main():
    N = 50000
    M = 20
    o1s = M/4
    nruns = 50
    to = N/2
    sigma2_u = 1.0
    a = 0.9998
    sigma2_v = 1.0
    if len(sys.argv)>1:
        sigma2_o1 = double(sys.argv[1])
    else:
        sigma2_o1 = 100.0
    if len(sys.argv)>2:
        sigma2_o2 = double(sys.argv[2])
    else:
        sigma2_o2 = 1.0
    mus = array([10**-12, 10**-7, 10**-4, 10**-3, 2*10**-3, 5*10**-3,\
    10**-2, 2*10**-2, 5*10**-2, 10**-1, 2*10**-1, 5*10**-1, 1.0])
    mses = zeros((len(mus), nruns))
    savedir = "../../../data/autostep/"
    dirname = "prob.s2u."+str(sigma2_u)+".a."+str(a)+".s2v."+\
    str(sigma2_v)+".s2o1."+str(sigma2_o1)+".s2o1."+str(sigma2_o2)+\
    ".m."+str(M)+".o1s."+str(o1s)+".N."+str(N)
    if not os.path.exists(savedir + dirname):
        os.makedirs(savedir + dirname)
    savetxt(savedir + dirname + "/metastepsizes.txt", mus)
    for run in range(nruns):
        for mu in mus:
            t = clock()
            seed(1)
            prob = Nonstationary(sigma2_u=sigma2_u, a=a, sigma2_v=\
            sigma2_v, sigma2_o1=sigma2_o1, \
            sigma2_o2=sigma2_o2, n=M, o1s=o1s)
            autostep = Autostep(w0=zeros(M), lmbda=10**-4, mu=mu,\
            alpha=1.0/M)
            w = zeros((M, N))
```

```
            mse = 0
            dev = 0
            seed(run)
            for n in range(N):
                ret = prob.step()
                d = ret["d"]
                u = ret["u"]
                ret = autostep.step(yt = d, xt = u)
                pred = ret["pred"]
                w[:, n] = ret["w"]
                error = d - pred
                if (n>=to):
                    mse = mse + 1.0*(error**2 - mse)/(n+1-to)

            mses[mu==mus, run==array(range(nruns))] = mse
            print "run" + str(run) + ": " + str(clock() - t) + "sec"
        savetxt(savedir + dirname+"/run"+str(run)+".txt",\
        sqrt(mses[:,run]))
    print sqrt(mean(mses, 1))
    savetxt(savedir + dirname+"/avg_rmse.txt", sqrt(mean(mses, 1)))
    if display==True:
        plot(mus, sqrt(mean(mses, 1)))
        title("autostep")
        xscale("log")
        ylim([0, 20])

if __name__ == '__main__':
    main()
    if display==True:
        show()
```

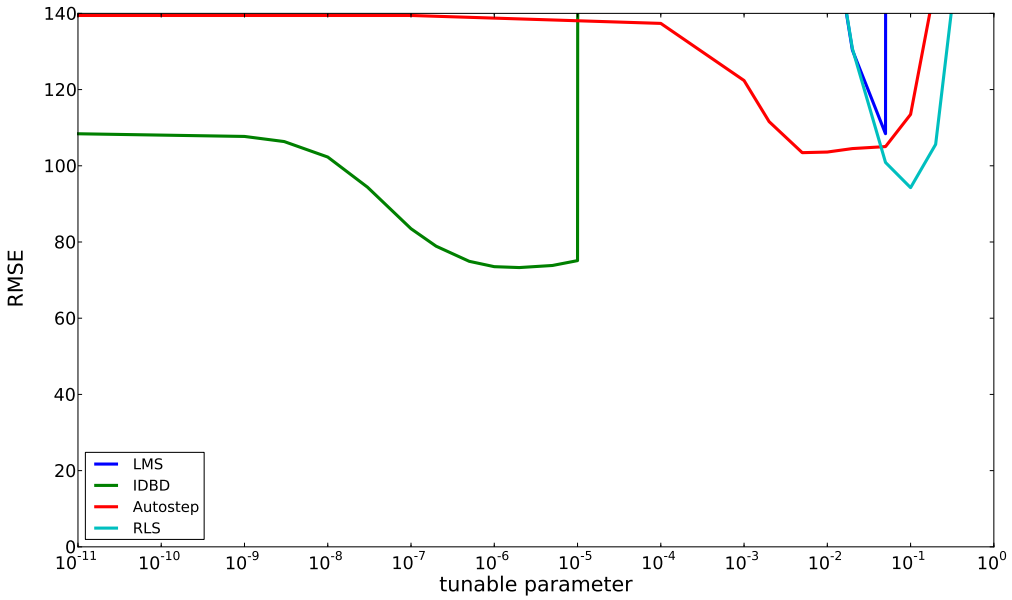


Figure 1

Discussion: We have tested LMS, RLS, IDBD and Autostep on the problem stated in the text. after generating 50000 sequential sample from both problem and each method was run on the data set separately.

For each method, we varied the tunable parameters from 1 to 10^{-12} with equal logarithmic intervals. These parameters are: the step size parameter μ for LMS, the meta-step size parameter α for IDBD and Autostep, and the exponential weighting factor γ for RLS (figure shows results for $1-\gamma$). We have set time-scale parameter τ of Autostep to 10^{-4} as recommended. All the estimated weights of all methods are initialized at zero. The initial step size for BMP, IDBD and Autostep is set to $1/M = 0.05$.

Our measure of performance is RMSe, averaged over last 25000 smaples.

Figure 1 shows the performance of all the methods as RMSE vs. tunable paramter values. The figure shows that IDBD achieves the best performance among all. Performance of Autostep is close to that of IDBD and Autostep learns a vector step size and hence can adapte to different tracking speed of the tap weights. LMS uses only a scalar step size and hence has to use the same speed of tracking for each element.

	RMSE	parameters
RLS	≈ 94.2	$1-\gamma = 0.1$
Autostep	≈ 103.5	$\alpha = 0.02$
LMS	≈ 108.4	$\mu = 0.05$
IDBD	≈ 73.3	$\alpha = 5 \times 10^{-6}$

Table 1: shows the best RMSE for different methods.

Note that if the speed of nonstationary is the same fo all the elements, then the vector step size will not have an advantage over a scalar step size. Therefore, the best performance of all the first order methods here will be similar. Our results confirm that.

Chapter 14

Problem 14.1

Let

$$\alpha(1) = \mathbf{y}(1) \tag{1}$$

$$\alpha(2) = \mathbf{y}(2) + \mathbf{A}_{1,1}\mathbf{y}(1) \tag{2}$$

where the matrix $\mathbf{A}_{1,1}$ is to be determined. This matrix is chosen so as to make the innovation process $\alpha(1)$ and $\alpha(2)$ uncorrelated with each other, that is:

$$\mathbb{E}\left[\alpha(2)\alpha^H(1)\right] = \mathbf{0} \tag{3}$$

Substitute Equations (1) and (2) into (3):

$$\mathbb{E}\left[\mathbf{y}(2)\mathbf{y}^H(1)\right] + \mathbf{A}_{1,1}\mathbb{E}\left[\mathbf{y}(1)\mathbf{y}^H(1)\right] = \mathbf{0}$$

Postmultiplying both sides of this equation by the inverse of $\mathbb{E}\left[\mathbf{y}(1)\mathbf{y}^H(1)\right]$ and rearranging:

$$\mathbf{A}_{1,1} = -\mathbb{E}\left[\mathbf{y}(2)\mathbf{y}^H(1)\right]\left\{\mathbb{E}\left[\mathbf{y}(1)\mathbf{y}^H(1)\right]\right\}^{-1}$$

We may now rewrite Equations (1) and (2) in the compact form

$$\begin{bmatrix}\alpha(1)\\ \alpha(2)\end{bmatrix} = \begin{bmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{A}_{1,1} & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{y}(1)\\ \mathbf{y}(2)\end{bmatrix}$$

This relation shows that given the observation vectors $\mathbf{y}(1)$ and $\mathbf{y}(2)$, we may compute the innovation process $\alpha(1)$ and $\alpha(2)$. The block lower triangular transformation matrix

$$\begin{bmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{A}_{1,1} & \mathbf{I}\end{bmatrix}$$

is invertible since its determinant equals 1. Hence, we may recover $\mathbf{y}(1)$ and $\mathbf{y}(2)$ from $\alpha(1)$ and $\alpha(2)$ by using the relation:

$$\begin{aligned}\begin{bmatrix}\mathbf{y}(1)\\ \mathbf{y}(2)\end{bmatrix} &= \begin{bmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{A}_{1,1} & \mathbf{I}\end{bmatrix}^{-1}\begin{bmatrix}\alpha(1)\\ \alpha(2)\end{bmatrix}\\ &= \begin{bmatrix}\mathbf{I} & \mathbf{0}\\ -\mathbf{A}_{1,1} & \mathbf{I}\end{bmatrix}\begin{bmatrix}\alpha(1)\\ \alpha(2)\end{bmatrix}\end{aligned}$$

In general, we may express the innovation process $\alpha(n)$ as a linear combination of the observations vectors $\mathbf{y}(1), \mathbf{y}(2), \ldots, \mathbf{y}(n)$ as follows:

$$\begin{aligned}\alpha(n) =& \mathbf{y}(n) + \mathbf{A}_{1,1}\mathbf{y}(n-1) + \ldots + \mathbf{A}_{n-1,n-1}\mathbf{y}(1)\\ =& \sum_{k=1}^{n} \mathbf{A}_{n,1;k}\mathbf{y}\left(n-k+1\right), \qquad n = 1, 2, \ldots\end{aligned}$$

where $\mathbf{A}_{n-1,0} = \mathbf{I}$. The set of matrices $\{\mathbf{A}_{n-1,k}\}$ is chosen to satisfy the following conditions:

$$\mathbb{E}\left[\alpha(n+1)\alpha^H(n)\right] = \mathbf{0}, \qquad n = 1, 2, \ldots$$

We may thus write

$$\begin{bmatrix}\alpha(1)\\ \alpha(2)\\ \vdots\\ \alpha(n)\end{bmatrix} = \begin{bmatrix}\mathbf{I} & \mathbf{0} & \ldots & \mathbf{0}\\ \mathbf{A}_{1,1} & \mathbf{I} & \ldots & \mathbf{0}\\ \vdots & \vdots & \ddots & \vdots\\ \mathbf{A}_{n-1,n-1} & \mathbf{A}_{n-1,n-2} & \ldots & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{y}(1)\\ \mathbf{y}(2)\\ \vdots\\ \mathbf{y}(n)\end{bmatrix}$$

The block lower triangular transformation matrix is invertible since its determinant equals one. Hence, we may go back and forth between the given set of observation vectors $\{\mathbf{y}(1), \mathbf{y}(2), \ldots, \mathbf{y}(n)\}$ and the corresponding set of innovation processes $\{\alpha(1), \alpha y(2), \ldots, \alpha y(n)\}$ without any loss of information.

Problem 14.2

First, we note that

$$\mathbb{E}\left[\varepsilon(n, n-1)\mathbf{v}_1^H(n)\right] = \mathbb{E}\left[\mathbf{x}(n)\mathbf{v}_1^H(n)\right] - \mathbb{E}\left[\hat{\mathbf{x}}\left(n|y_{n-1}\right)\mathbf{v}_1^H(n)\right]$$

Since the estimate $\hat{\mathbf{x}}\,(n|y_{n-1})$ consists of a linear combination of observations vectors $\mathbf{y}(1),\ldots,\mathbf{y}(n-1)$, and since

$$\mathbb{E}\left[\mathbf{y}(k)\mathbf{v}_1^H\right]=\mathbf{0},\qquad 0\leq k\leq n$$

it follows that

$$\mathbb{E}\left[\hat{\mathbf{x}}\,(n|y_{n-1})\,\mathbf{v}_1^H(n)\right]=\mathbf{0}$$

We also have

$$\mathbb{E}\left[\hat{\mathbf{x}}\,(n)\,\mathbf{v}_1^H(n)\right]=\Phi(n,0)\mathbb{E}\left[\mathbf{x}(0)\mathbf{v}_1^H(n)\right]+\sum_{i=1}^{n-1}\Phi(n,i)\mathbb{E}\left[\mathbf{v}_1(i)\mathbf{v}_1^H(n)\right]$$

Since, by hypothesis

$$\mathbb{E}\left[\mathbf{x}(0)\mathbf{v}_1^H(n)\right]=\mathbf{0}$$

and

$$\mathbb{E}\left[\mathbf{v}_1(i)\mathbf{v}_1^H(n)\right]=\mathbf{0},\qquad 0\leq i\leq n$$

it follows that

$$\mathbb{E}\left[\mathbf{x}(n)\mathbf{v}_1^H(n)\right]=\mathbf{0}$$

Accordingly, we deduce that

$$\mathbb{E}\left[\varepsilon(n,n-1)\mathbf{v}_1^H(n)\right]=\mathbf{0}$$

Next, we note that

$$\mathbb{E}\left[\varepsilon(n,n-1)\mathbf{v}_2^H(n)\right]=\mathbb{E}\left[\mathbf{x}(n)\mathbf{v}_2^H(n)\right]-\mathbb{E}\left[\hat{\mathbf{x}}\,(n|y_{n-1})\,\mathbf{v}_2^H(n)\right]$$

We have

$$\mathbb{E}\left[\mathbf{x}(n)\mathbf{v}_2^H(n)\right]=\mathbf{0}$$

Also, since $(\hat{\mathbf{x}}(n)|y_{n-1})$ consists of a linear combination of $\mathbf{y}(1),\ldots,\mathbf{y}(n-1)$ and since

$$\mathbb{E}\left[\mathbf{y}(k)\mathbf{v}_2^H(n)\right]=\mathbf{0},\qquad 1\leq k\leq n-1$$

it follows that

$$\mathbb{E}\left[\hat{\mathbf{x}}\,(n|y_{n-1})\,\mathbf{v}_2^H(n)\right]=\mathbf{0}$$

We therefore conclude that

$$\mathbb{E}\left[\varepsilon(n,n-1)\mathbf{v}_2^H(n)\right]=\mathbf{0}$$

Problem 14.3

The estimated state-error vector equals

$$\begin{aligned}\varepsilon(i,n) =&\mathbf{x}(i) - \hat{\mathbf{x}}\left(i|\mathbf{y}_n\right)\\ =&\mathbf{x}(i) - \sum_{k=1}^{n}\mathbf{b}_i(k)\alpha(k)\end{aligned}$$

The expected value of the squared norm of $\varepsilon(i,n)$ is given by

$$\begin{aligned}\mathbb{E}\left[\|\varepsilon(i,n)\|^2\right] =&\mathbb{E}\left[\varepsilon^H(i,n)\varepsilon(i,n)\right]\\ =&\sum_{k=1}^{n}\sum_{l=1}^{n}\mathbf{b}_i^H(k)\mathbf{b}_i(l)\mathbb{E}\left[\alpha^*(k)\alpha(l)\right] - \sum_{k=1}^{n}\mathbf{b}_i^H(k)\mathbb{E}\left[\mathbf{x}(i)\alpha^*(k)\right]\\ &- \sum_{k=1}^{n}\mathbb{E}\left[\mathbf{x}^H(i)\alpha(k)\right]\mathbf{b}_i(k) + \mathbb{E}\left[\mathbf{x}^H(i)\mathbf{x}\left(i\right)\right]\end{aligned}$$

Differentiating this index of performance with respect to the vector $\mathbf{b}_i(k)$, based on the Wirtinger calculus, and setting the result equal to zero, we find that the optimum value of $\mathbf{b}_i(k)$ is determined by

$$2\mathbf{b}_i(k)\mathbb{E}\left[\alpha(k)\alpha^*(k)\right] - 2\mathbb{E}\left[\mathbf{x}(i)\alpha^*(k)\right] = \mathbf{0}$$

Hence, the optimum value of $\mathbf{b}_i(k)$ equals

$$\mathbf{b}_i(k) = \mathbb{E}\left[\mathbf{x}(i)\alpha^*(k)\right]\sigma_\alpha^{-2}$$

where

$$\sigma_\alpha^{-2} = \mathbb{E}\left[|\alpha(k)|^2\right]$$

Correspondingly, the estimate of the state vector is given by

$$\begin{aligned}\hat{\mathbf{x}}\left(i|y_n\right) =&\sum_{k=1}^{n}\mathbb{E}\left[\mathbf{x}(i)\alpha^*(k)\right]\sigma_\alpha^{-2}\alpha(k)\\ =&\sum_{k=1}^{n}\mathbb{E}\left[\mathbf{x}(i)\phi^*(k)\right]\phi(k)\end{aligned}$$

where $\phi(k)$ is the normalized innovation:

$$\phi(k) = \frac{\alpha(k)}{\sigma_\alpha}$$

Problem 14.4

a)

The matrices $\mathbf{K}(n, n-1)$ and $\mathbf{Q}_2(n)$ are both correlation matrices and therefore nonnegative definite. In particular, we note

$$\mathbf{K}(n, n-1) = \mathbb{E}\left[\varepsilon(n, n-1)\varepsilon^H(n, n-1)\right]$$

$$\mathbf{Q}_2(n) = \mathbb{E}\left[\mathbf{v}_2(n)\mathbf{v}_2^H(n)\right]$$

where $\varepsilon(n, n-1)$ is the predicted state-error vector, and $\mathbf{v}_2(n)$ is the measurement noise vector. We may therefore express $\mathbf{R}(n)$ in the equivalent form

$$\mathbf{R}(n) = \mathbb{E}\left[\mathbf{e}(n)\mathbf{e}^H(n)\right]$$

where

$$\mathbf{e}(n) = \mathbf{C}(n)\varepsilon(n, n-1) + \mathbf{v}_2(n)$$

where it is noted that $\varepsilon(n, n-1)$ and $\mathbf{v}_2(n)$ are uncorrelated. We now see that $\mathbf{R}(n)$ is a correlation matrix, which is, nonnegative definite.

b)

For $\mathbf{R}(n)$ to be nonsingular, and therefor the inverse matrix $\mathbf{R}^{-1}$ to exist, we demand that $\mathbf{Q}_2(n)$ be positive definite such that the determinant of $\mathbf{C}(n)\mathbf{K}(n, n-1)\mathbf{C}^H(n) + \mathbf{Q}_2(n)$ is nonzero. This requirement, in effect, says that no measurement is exact, hence the unavoidable presence of measurement noise and therefore $\mathbf{Q}_2(n)$. Such a requirement is reasonable on physical grounds.

Problem 14.5

In the limit, when n approaches infinity, we may put

$$\mathbf{K}(n+1, n) = \mathbf{K}(n, n-1) = \mathbf{K}$$

Under this condition, Equation (14.54) of the textbook simplifies to

$$\mathbf{K} = (\mathbf{I} - \mathbf{GC})\,\mathbf{K}\left(\mathbf{I} - \mathbf{C}^H\mathbf{G}^H\right) + \mathbf{Q}_1 + \mathbf{G}\mathbf{Q}_2\mathbf{G}^H \tag{1}$$

where it is assumed that the state-transition matrix equals the identity matrix, and $\mathbf{G}$, $\mathbf{C}$, $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are the limiting values of the matrices$\mathbf{G}(n)$, $\mathbf{C}(n)$, $\mathbf{Q}_1(n)$ and $\mathbf{Q}_2(n)$,

respectively. From Equations (14.49) and (14.35) we find that the limiting value of the Kalman gain is

$$\mathbf{G} = \mathbf{K}\mathbf{C}^H \left(\mathbf{C}\mathbf{K}\mathbf{C}^H + \mathbf{Q}_2\right)^{-1} \tag{2}$$

Expanding Equation (1) and then using Equation (2) to eliminate $\mathbf{G}$, we get the desired result:

$$\mathbf{K}\mathbf{C}^H \left(\mathbf{C}\mathbf{K}\mathbf{C}^H + \mathbf{Q}_2\right)^{-1} \mathbf{C}\mathbf{K} - \mathbf{Q} = \mathbf{0}$$

Problem 14.6

$$\mathbf{x}(n+1) = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \mathbf{x}(n) + \mathbf{v}_1(n) \qquad \mathbf{Q}_1 = \mathbf{I}; \quad \mathbb{E}\left[\mathbf{v}_1\right] = \mathbf{0}$$

$$\mathbf{y}(n) = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x}(n) + \mathbf{v}_2(n) \qquad \mathbf{Q}_2 = 1; \quad \mathbb{E}\left[\mathbf{v}_2\right] = \mathbf{0}$$

a)

Using Table 14.2, we formulate the recursion for computing the Kalman filter's known parameters:

$$\mathbf{F}(n+1, n) = \mathbf{F} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

$$\mathbf{c}(n) = \mathbf{c} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

$$\mathbf{Q}_1(n) = \mathbf{I}$$

$$\mathbf{Q}_2(n) = 1$$

For the unknown parameters, we have

$$\mathbf{K}(n, n-1) = \mathbf{K} = \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix}$$

We may then write

$$\mathbf{G}(n) = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \mathbf{K}(n, n-1) \begin{bmatrix} 1 \\ 0 \end{bmatrix} \left(\begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{K}(n, n-1) \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 1 \right)^{-1}$$

$$= \begin{bmatrix} \dfrac{k_{21}(n, n-1)}{k_{11}(n, n-1) + 1} \\ \dfrac{k_{11}(n, n-1) + k_{21}(n, n-1)}{k_{11}(n, n-1) + 1} \end{bmatrix}$$

$$\begin{aligned}\alpha(n) =& y(n) - \begin{bmatrix}1 & 0\end{bmatrix}\hat{\mathbf{x}}(n|\mathbf{Y}_{n-1}) \\ =& y(n) - \hat{\mathbf{x}}_1(n|\mathbf{Y}_{n-1})\end{aligned}$$

$$\begin{aligned}\hat{\mathbf{x}}(n+1|\mathbf{Y}_n) &= \begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}\hat{\mathbf{x}}(n|\mathbf{Y}_{n-1}) + \mathbf{G}(n)\alpha(n) \\ &= \begin{bmatrix} \dfrac{\{k_{11}(n,n-1) - k_{21}(n,n-1)\}\,\hat{x}_1(n|Y_{n-1}) + k_{21}(n,n-1)y(n)}{k_{11}(n,n-1)+1} \\ \dfrac{\{1 - k_{21}(n,n-1)\}\,\hat{x}_1(n|Y_{n-1}) + \{k_{11}(n,n-1) + k_{21}(n,n-1)y)(n)\}}{k_{11}(n,n-1)+1}\end{bmatrix}\end{aligned} \tag{1}$$

$$\begin{aligned}\mathbf{K}(n) =& \mathbf{K}(n,n-1) - \begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}\mathbf{G}(n)\begin{bmatrix}1 & 0\end{bmatrix}\mathbf{K}(n,n-1) \\ =& \mathbf{K}(n,n-1) - \begin{bmatrix}\dfrac{k_{11}+k_{21}}{k_{11}+1} & 0 \\ \dfrac{k_{11}+2k_{21}}{k_{11}+1} & 0\end{bmatrix}\mathbf{K}(n,n-1) \\ =& \mathbf{K}(n,n-1) - \begin{bmatrix}k_{11}\dfrac{k_{11}+k_{21}}{k_{11}+1} & k_{12}\dfrac{k_{11}+k_{21}}{k_{11}+1} \\ k_{11}\dfrac{k_{11}+2k_{21}}{k_{11}+1} & k_{12}\dfrac{k_{11}+k_{21}}{k_{11}+1}\end{bmatrix} \\ =& \begin{bmatrix} k_{11}\dfrac{-k_{21}+1}{k_{11}+1} & k_{12}\dfrac{-k_{21}+1}{k_{11}+1} \\ k_{21} - k_{11}\dfrac{k_{11}+2k_{21}}{k_{11}+1} & k_{22} - k_{12}\dfrac{k_{11}+2k_{21}}{k_{11}+1}\end{bmatrix}\end{aligned}$$

$$\begin{aligned}\mathbf{K}(n+1,n) &= \begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}\mathbf{K}(n)\begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix} + \mathbf{I} \\ &= \begin{bmatrix} 1 + k_{22} - k_{12}\left(\dfrac{k_{11}+2k_{21}}{k_{11}+1}\right) & k_{21} + k_{22} - (k_{11}+k_{12})\left(\dfrac{k_{11}+2k_{21}}{k_{11}+1}\right) \\ k_{22} - k_{12}\left(\dfrac{k_{11}+k_{21}}{k_{11}+1}\right) & 1 + k_{21} + k_{22} + (k_{11}+k_{12})\left(\dfrac{1-k_{11}+k_{21}}{k_{11}+1}\right)\end{bmatrix}\end{aligned} \tag{2}$$

For this particular case, the Kalman filtering process is entirely described by the pair of Equations (1) and (2).

b)

The generalized form of the algebraic Riccati equation is given by

$$\mathbf{K} - \mathbf{F}\mathbf{K}\mathbf{F}^H + \mathbf{F}\mathbf{K}\mathbf{C}^H \left(\mathbf{C}\mathbf{K}\mathbf{C}^H + \mathbf{Q}_2\right)^{-1} \mathbf{C}\mathbf{K}\mathbf{F}^H - \mathbf{Q}_1 = \mathbf{0}$$

For our problem:

$$\mathbf{F} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}, \qquad \mathbf{Q}_1 = \mathbf{I}, \qquad \mathbf{K} = \begin{bmatrix} k_1 & k_2 \\ k_2 & k_3 \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 \end{bmatrix}, \qquad \mathbf{Q}_2 = [1],$$

Therefore, we have:

$$\begin{aligned} \mathbf{K} = & \overbrace{\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} - \underbrace{\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} k_1 & k_2 \\ k_2 & k_3 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}}_{\mathbf{B}}}^{\mathbf{C}} \\ & + \underbrace{\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} k_1 & k_2 \\ k_2 & k_3 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \left(\begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} k_1 & k_2 \\ k_2 & k_3 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} k_1 & k_2 \\ k_2 & k_3 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}}_{\mathbf{A}} - \mathbf{I} \\ = & \mathbf{0} \end{aligned}$$

$$\begin{aligned} \mathbf{A} = & \frac{1}{k_1 + 1} \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} k_1 & k_2 \\ k_2 & k_3 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} k_1 & k_2 \\ k_2 & k_3 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \\ = & \frac{1}{k_1 + 1} \begin{bmatrix} k_2 & k_3 \\ k_1 + k_2 & k_2 + k_3 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} k_2 & k_1 + k_2 \\ k_3 & k_2 + k_3 \end{bmatrix} \\ = & \frac{1}{k_1 + 1} \begin{bmatrix} k_2 & 0 \\ k_1 + k_2 & 0 \end{bmatrix} \begin{bmatrix} k_2 & k_1 + k_2 \\ k_3 & k_2 + k_3 \end{bmatrix} \\ = & \frac{1}{k_1 + 1} \begin{bmatrix} k_2^2 & k_2(k_1 + k_2) \\ k_2(k_1 + k_2) & (k_1 + k_2)^2 \end{bmatrix} \\ = & \begin{bmatrix} \frac{k_2^2}{k_1 + 1} & \frac{k_2(k_1 + k_2)}{k_1 + 1} \\ \frac{k_2(k_1 + k_2)}{k_1 + 1} & \frac{(k_1 + k_2)^2}{k_1 + 1} \end{bmatrix} \end{aligned}$$

$$\begin{aligned}\mathbf{B} &= \begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}\begin{bmatrix}k_1 & k_2\\ k_2 & k_3\end{bmatrix}\begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}\\ &= \begin{bmatrix} k_2 & k_3\\ k_1+k_2 & k_2+k_3\end{bmatrix}\begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}\\ &= \begin{bmatrix} k_2 & k_2+k_3\\ k_2+k_3 & k_1+2k_2+k_3\end{bmatrix}\end{aligned}$$

$$\begin{aligned}\mathbf{C} =& \mathbf{K}-\mathbf{B}\\ &= \begin{bmatrix}k_1-k_3 & -k_3\\ -k_3 & -k_1-2k_2\end{bmatrix}\end{aligned}$$

$$\mathbf{C}+\mathbf{A}-\mathbf{I}=\mathbf{0}$$

$$\begin{bmatrix}\left(\frac{k_2^2}{k_1+1}+k_1-k_3-1\right) & \left(\frac{k_2(k_1+k_2)}{k_1+1}-k_3\right)\\ \left(\frac{k_2(k_1+k_2)}{k_1+1}-k_3\right) & \left(\frac{(k_1+k_2)^2}{k_1+1}-k_1-2k_2-1\right)\end{bmatrix}=\begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix}$$

$$\begin{cases}\dfrac{k_2(k_1+k_2)}{k_1+1} = k_3\\ \dfrac{k_2^2-k_2(k_1+k_2)}{k_1+1}+k_1-1 = 0\\ \dfrac{(k_1+k_2)^2}{k_1+1}-k_1-2k_2-1 = 0\end{cases}\rightarrow\begin{cases}\dfrac{-k_1k_2}{k_1+1}+k_1-1 = 0\\ \dfrac{(k_1+k_2)^2}{k_1+1}-k_1-2k_2-1 = 0\end{cases}\rightarrow$$

$$\begin{cases}k_2=\frac{k_1^2-1}{k_1}\\ \cancel{k_1^2}+\cancel{2k_1k_2}+k_2^2-\cancel{k_1^2}-k_1-\cancel{2k_1k_2}-2k_2-k_1-1=0\end{cases}\rightarrow\begin{cases}k_2=\frac{k_1^2-1}{k_1}\\ k_2^2-2k_2-2k_1=0\end{cases}$$

$$\frac{\left(k_1^2-1\right)^2}{k_1^2}-2\frac{k_1^2-1}{k_1}-2k_1-1=0$$

$$k_1^4-2k_1^2+1-2k_1^3+2k_1-2k_1^3-2k_1^2=0$$

$$k_1^4-4k_1^3-3k_1^2+2k_1+1=0$$

This equation has four different real-valued solutions, but it is easy to show that only one of them is meaningful in the context of our problem.

Problem 14.7

We start with

$$\alpha(n) = \mathbf{y}(n) - \mathbf{C}(n)\hat{\mathbf{x}}(n|\mathbf{Y}_{n-1}) \tag{1}$$

$$\hat{\mathbf{x}}(n+1|\mathbf{Y}_n) = \mathbf{F}(n+1,n)\hat{\mathbf{x}}(n|\mathbf{Y}_{n-1}) + \mathbf{G}(n)\alpha(n) \tag{2}$$

Substituting Equation (1) into Equation (2):

$$\hat{\mathbf{x}}(n+1|\mathbf{Y}_n) = \mathbf{F}(n+1,n)\hat{\mathbf{x}}(n|\mathbf{Y}_{n-1}) + \mathbf{G}(n)\left[\mathbf{y}(n) - \mathbf{C}(n)\mathbf{x}(n|\mathbf{Y}_{n-1})\right]$$

The filtered state estimate is given by

$$\begin{aligned}\hat{\mathbf{x}}(n|\mathbf{Y}_n) =& \mathbf{F}(n,n+1)\hat{\mathbf{x}}(n+1|\mathbf{Y}_n)\\ =& \hat{\mathbf{x}}(n|\mathbf{Y}_{n-1}) + \mathbf{F}(n,n+1)\mathbf{G}(n)\left[\mathbf{y}(n) - \mathbf{C}(n)\hat{\mathbf{x}}(n|\mathbf{Y}_{n-1})\right]\\ =& \left[\mathbf{I} - \mathbf{F}(n,n+1)\mathbf{G}(n)\mathbf{C}(n)\right]\hat{\mathbf{x}}(n|\mathbf{Y}_{n-1}) + \mathbf{F}(n,n+1)\mathbf{G}(n)\mathbf{y}(n)\end{aligned}$$

But

$$\mathbf{y}(n) = \mathbf{C}(n)\mathbf{x}(n) + \mathbf{v}_2(n)$$

Hence,

$$\begin{aligned}\hat{\mathbf{x}}(n|\mathbf{Y}_n) = &\left[\mathbf{I} - \mathbf{F}(n,n+1)\mathbf{G}(n)\mathbf{C}(n)\right]\hat{\mathbf{x}}(n|\mathbf{Y}_{n-1}) + \mathbf{F}(n,n+1)\mathbf{G}(n)\mathbf{C}(n)\mathbf{x}(n)\\ &+ \mathbf{F}(n,n+1)\mathbf{G}(n)\mathbf{v}_2(n)\end{aligned}$$

Taking expectations, and recognizing that the measurement noise vector $\mathbf{v}_2(n)$ has zero mean:

$$\begin{aligned}\mathbb{E}\left[\hat{\mathbf{x}}(n|\mathbf{Y}_n)\right] = &\left[\mathbf{I} - \mathbf{F}(n,n+1)\mathbf{G}(n)\mathbf{C}(n)\right]\mathbb{E}\left[\hat{\mathbf{x}}(n|\mathbf{Y}_{n-1})\right]\\ &+ \mathbf{F}(n,n+1)\mathbf{G}(n)\mathbf{C}(n)\mathbb{E}\left[\mathbf{x}(n)\right]\end{aligned}$$

For $n = 1$, we thus get

$$\begin{aligned}\mathbb{E}\left[\hat{\mathbf{x}}(1|\mathbf{Y}_1)\right] = &\left[\mathbf{I} - \mathbf{F}(1,2)\mathbf{G}(1)\mathbf{C}(1)\right]\mathbb{E}\left[\hat{\mathbf{x}}(1|\mathbf{Y}_0)\right]\\ &+ \mathbf{F}(1,2)\mathbf{G}(1)\mathbf{C}(1)\mathbb{E}\left[\mathbf{x}(1)\right]\end{aligned} \tag{1}$$

Since the one-step prediction $\hat{\mathbf{x}}(1|\mathbf{Y}_0)$ must be specified, we have

$$\mathbb{E}\left[\hat{\mathbf{x}}(1|\mathbf{Y}_0)\right] = \hat{\mathbf{x}}(1|\mathbf{Y}_0)$$

Accordingly, substituting the choice of the initial condition

$$\hat{\mathbf{x}}(1|\mathbf{Y}_0) = \mathbb{E}[\mathbf{x}(1)]$$

in Equation (1) yields

$$\begin{aligned}\mathbb{E}[\hat{\mathbf{x}}(1|\mathbf{Y}_1)] =& [\mathbf{I} - \mathbf{F}(1,2)\mathbf{G}(1)\mathbf{C}(1)]\,\mathbb{E}[\mathbf{x}(1)] \\ &+ \mathbf{F}(1,2)\mathbf{G}(1)\mathbf{C}(1)\mathbb{E}[\mathbf{x}(1)] \\ =& \mathbb{E}[\mathbf{x}(1)]\end{aligned}$$

By induction, we may go on to show that in general:

$$\mathbb{E}[\hat{\mathbf{x}}|\mathbf{Y}_n)] = \mathbb{E}[\mathbf{x}(n)]$$

In other words, the filtered estimate $\hat{\mathbf{x}}(n|\mathbf{y}_n)$ produced by the Kalman filter is an unbiased estimate for the specified method of initialization.

Problem 14.8

In this problem, we are required to derive the MAP version of the Kalman filter.

a)

$$f_X(\mathbf{x}(n)|\mathbf{y}(n)) = \frac{f_{XY}(\mathbf{x}(n), \mathbf{Y}_n)}{f_Y(\mathbf{Y}_n)} = \frac{f_{XY}(\mathbf{x}(n), \mathbf{y}(n), \mathbf{Y}_{n-1})}{f_Y(\mathbf{y}(n), \mathbf{Y}_{n-1})} \tag{1}$$

where

$$\begin{aligned}f_{XY}(\mathbf{x}(n), \mathbf{y}(n), \mathbf{Y}_{n-1}) =& f_Y(\mathbf{y}(n)|\mathbf{x}(n), \mathbf{Y}_{n-1}) f_{XY}(\mathbf{x}(n), \mathbf{Y}_{n-1}) \\ =& f_Y(\mathbf{y}(n)|\mathbf{x}(n), \mathbf{Y}_{n-1}) f_X(\mathbf{x}(n)|\mathbf{Y}_{n-1}) f_Y(\mathbf{Y}_{n-1}) \\ =& f_Y(\mathbf{y}(n)|\mathbf{x}(n)) f_X(\mathbf{x}(n)|\mathbf{Y}_{n-1}) f_Y(\mathbf{Y}_{n-1})\end{aligned}$$

where we have used the fact that $\mathbf{y}(n) = \mathbf{C}(n)\mathbf{x}(n) + \mathbf{v}_2(n)$ does not depend on $\mathbf{Y}_{n-1}$. Consequently,

$$f_{\mathbf{Y}}(\mathbf{y}_(n), \mathbf{Y}_{n-1}) = f_{\mathbf{Y}}(\mathbf{y}_n|\mathbf{Y}_{n-1})\, f(\mathbf{Y}_{n-1}) \tag{2}$$

Substituting Equation (2) into Equation (1), we thus get

$$\begin{aligned}f_X(\mathbf{x}(n)|\mathbf{Y}_n) =& \frac{f_Y(\mathbf{y}(n)|\mathbf{x}(n)) f_X(\mathbf{x}(n)|\mathbf{Y}_{n-1}) f_Y(Y_{n-1})}{f_Y(\mathbf{y}(n), \mathbf{Y}_{n-1})} \\ =& \frac{f_Y(\mathbf{y}(n)|\mathbf{x}(n)) f_X(\mathbf{x}(n)|\mathbf{Y}_{n-1}) \cancel{f_Y(Y_{n-1})}}{f_Y(\mathbf{y}(n)|\mathbf{Y}_{n-1}) \cancel{f_Y(\mathbf{Y}_{n-1})}} \\ =& \frac{f_Y(\mathbf{y}(n)|\mathbf{x}(n)) f_X(\mathbf{x}(n)|\mathbf{Y}_{n-1})}{f_Y(\mathbf{y}(n)|\mathbf{Y}_{n-1})}\end{aligned} \tag{3}$$

b)

Examine first $f_Y(\mathbf{y}(n)|\mathbf{x}(n))$; its mean is

$$\begin{aligned}\mathbb{E}\left[\mathbf{y}(n)|\mathbf{x}(n)\right] =& \mathbb{E}\left[\mathbf{C}(n)\mathbf{x}(n)+\mathbf{v}_2(n)|\mathbf{x}(n)\right]\\ =& \mathbf{C}(n)\mathbf{x}(n)\end{aligned}$$

The variance is

$$\text{var}\left[\mathbf{y}(n)|\mathbf{x}(n)\right] = \text{var}\left[\mathbf{v}_2(n)|\mathbf{x}(n)\right] = \mathbf{Q}_2(n),$$

where $\mathbf{Q}_2(n)$= correlation matrix of measurement noise $\mathbf{v}_2(n)$. Thus, assuming Gaussianity, we may write

$$f_Y(\mathbf{y}(n)|\mathbf{x}(n)) = A_1 \exp\left(-\frac{1}{2}\left(\mathbf{y}(n)-\mathbf{C}(n)\mathbf{x}(n)\right)^H \mathbf{Q}_2^{-1}(n)\left(\mathbf{y}(n)-\mathbf{C}(n)\mathbf{x}(n)\right)\right) \quad (4)$$

where the constant A_1 is an appropriate scaling factor. Consider next $f_{\mathbf{x}}(\mathbf{x}|\mathbf{Y}_{n-1})$; its mean is given by

$$\begin{aligned}\mathbb{E}\left[\mathbf{x}(n)|\mathbf{Y}_{n-1}\right] =& \mathbb{E}\left[\mathbf{F}(n,n-1)\hat{\mathbf{x}}(n-1)+\mathbf{v}_1(n-1)|\mathbf{Y}_{n-1}\right]\\ =& \mathbb{E}\left[\mathbf{F}(n,n-1)\hat{\mathbf{x}}(n-1)|\mathbf{Y}_{n-1}\right]\\ =& \hat{\mathbf{x}}(n|\mathbf{Y}_{n-1})\end{aligned}$$

The variance is therefore

$$\begin{aligned}\text{var}\left[\mathbf{x}(n)|\mathbf{Y}_{n-1}\right] =& \text{var}\left[\mathbf{x}(n)-\hat{\mathbf{x}}(n)|\mathbf{Y}_{n-1}\right]\\ =& \text{var}\left[\varepsilon(n,n-1)\right]\end{aligned}$$

where $\varepsilon(n,n-1)$ is the state-error vector. Denote this variance by $\mathbf{K}(n,n-1)$, which is to be determined.

Again, assuming Gaussianity, we may write

$$f_X(\mathbf{x}(n)|\mathbf{Y}_{n-1}) = A_2 \exp\left(-\frac{1}{2}\left(\mathbf{x}(n)-\hat{\mathbf{x}}(n)|\mathbf{Y}_{n-1}\right)^H \mathbf{K}^{-1}(n,n-1)\left(\mathbf{x}(n)-\hat{\mathbf{x}}(n)|\mathbf{Y}_{n-1}\right)\right) \quad (5)$$

where A_2 is another appropriate scaling factor. Thus, substituting Equation (4) and (5) into Equation (3), we get

$$f_X(\mathbf{x}(n)|\mathbf{Y}_n) = A \exp\left(-\frac{1}{2} \left(\mathbf{y}(n) - \mathbf{C}(n)\mathbf{x}(n)\right)^H \mathbf{Q}_2^{-1}(n) \left(\mathbf{C}(n)\mathbf{x}(n)\right) \right.$$
$$\left. -\frac{1}{2} \left(\mathbf{x}(n) - \hat{\mathbf{x}}(n)|\mathbf{Y}_{n-1}\right)^H \mathbf{K}^{-1}\left(n, n-1\right) \left(\mathbf{x}(n) - \hat{\mathbf{x}}(n)|\mathbf{Y}_{n-1}\right) \right) \tag{6}$$

Where the new constant A is defined by $A = A_1 A_2$. Note also that for convenience of presentation, we have put aside the numerator $f_Y(\mathbf{y}(n)|\mathbf{Y}_{n-1})$ aside as it does not involve the state $\mathbf{x}(n)$.

c)

By definition, the MAP estimate of the state is defined by the condition

$$\left. \frac{\partial \ln f_X(\mathbf{x}(n)|\mathbf{Y}_n)}{\partial(\mathbf{x}^H(n))} \right|_{\mathbf{x}(n)=\hat{\mathbf{x}}_{\text{MAP}}(n)} = \mathbf{0} \tag{7}$$

Hence, substituting Equation (6) into Equation (7) and using the Wirtinger calculus, we see the MAP estimate of $\mathbf{x}(n)$ as shown by

$$\hat{\mathbf{x}}_{\text{MAP}}(n) = \left[\mathbf{C}^H(n)\mathbf{Q}_2^{-1}(n)\mathbf{C}(n) + \mathbf{K}^{-1}(n, n-1)\right]^{-1} \left[\mathbf{K}^{-1}(n, n-1)\hat{\mathbf{x}}(n|\mathbf{Y}_{n-1}) \right.$$
$$\left. + \mathbf{C}^H(n)\mathbf{Q}_2^{-1}(n)\mathbf{y}(n)\right] \tag{8}$$

From a computational point of view, we need to put the first inverse matrix in Equation (8) into a more convenient form. To this end, we apply the matrix inversion lemma, which may be stated as follows (see section 10.2 in the textbook):

If

$$\mathbf{A} = \mathbf{B}^{-1} + \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^H$$

then

$$\mathbf{A}^{-1} = \mathbf{B} - \mathbf{B}\mathbf{C}\left(\mathbf{D} + \mathbf{C}^H\mathbf{B}\mathbf{C}\right)^{-1}\mathbf{C}^H\mathbf{B}$$

With the expression $\left[\mathbf{C}^H(n)\mathbf{Q}_2^{-1}(n)\mathbf{C}(n) + \mathbf{K}^{-1}(n, n-1)\right]^{-1}$ as the issue of concern, we note the following

$$\mathbf{B} = \mathbf{K}(n, n-1)$$

$$\mathbf{C} = \mathbf{C}^H(n)$$
$$\mathbf{D} = \mathbf{Q}_2(n)$$

Hence, applying the matrix inversion lemma, we get

$$\begin{aligned}\left[\mathbf{C}^H(n)\mathbf{Q}_2^{-1}(n)\mathbf{C}(n) + \mathbf{K}^{-1}(n, n-1)\right]^{-1} = \\ \mathbf{K}(n, n-1) - \mathbf{K}(n, n-1)\mathbf{C}^H(n)\left(\mathbf{Q}_2(n) + \mathbf{C}(n)\mathbf{K}(n, n-1)\mathbf{C}^H(n)\right)^{-1} \\ \times\mathbf{C}(n)\mathbf{K}(n, n-1)\end{aligned} \tag{9}$$

Next, substituting Equation (9) into Equation (8), and then going through some lengthy but straightforward algebraic manipulations, we get

$$\hat{\mathbf{x}}_{\text{MAP}}(n) = \hat{\mathbf{x}}(n|\mathbf{Y}_{n-1}) + \mathbf{G}(n)\left[\mathbf{y}(n) - \mathbf{C}(n)\hat{\curvearrowleft}(n|\mathbf{Y}_{n-1})\right] \tag{10}$$

where $\mathbf{G}(n)$ is the Kalman gain defined by

$$\mathbf{G}(n) = \mathbf{F}(n+1, n)\mathbf{K}(n, n-1)\mathbf{C}^H(n)\left[\mathbf{C}(n)\mathbf{K}(n, n-1)\mathbf{C}^H(n) + \mathbf{Q}_2(n)\right]^{-1} \tag{11}$$

The one issue that is yet to be determined is $\mathbf{K}(n, n-1)$. Here we note:

$$\begin{aligned}\epsilon(n, n-1) =& \mathbf{x}(n) - \hat{\mathbf{x}}(n|\mathbf{Y}_{n-1}) \\ =& \underbrace{\mathbf{F}(n, n-1)\mathbf{x}(n-1) + \mathbf{v}_1(n-1)}_{\mathbf{x}(n)} - \mathbf{F}(n, n-1)\hat{\mathbf{x}}_{\text{MAP}}(n-1) \\ =& \mathbf{F}(n, n-1)\epsilon_{\text{MAP}}(n-1) + \mathbf{v}_1(n-1)\end{aligned}$$

Therefore, referring to Equation (14.55) in the textbook, we may write

$$\mathbf{K}(n, n-1) = \mathbf{F}(n, n-1)\mathbf{K}(n-1)\mathbf{F}^H(n, n-1) + \mathbf{Q}_1(n-1) \tag{12}$$

where

$$\mathbf{K}(n) = \text{var}\left[\epsilon_{\text{MAP}}(n)\right]$$

There only remains for us to determine $\mathbf{K}(n-1)$. Here we note

$$\begin{aligned}\epsilon_{\text{MAP}}(n) =& \mathbf{x}(n) - \hat{\mathbf{x}}_{\text{MAP}}(n) \\ =& \mathbf{x} + (-\hat{\mathbf{x}}(n|\mathbf{Y}_{n-1})) + \mathbf{G}(n)\left[\mathbf{y}(n) - \mathbf{C}(n)\hat{\mathbf{x}}(n|\mathbf{Y}_{n-1})\right]\end{aligned}$$

But

$$\mathbf{y}(n) = \mathbf{C}(n)\mathbf{x}(n) + \mathbf{v}_2(n)$$

Hence, noting that $\epsilon(n, n-1) = \mathbf{x}(n) - \hat{\mathbf{x}}(n|\mathbf{Y}_{n-1})$

$$\begin{aligned}\epsilon_{\text{MAP}}(n) =& \epsilon(n, n-1) - \mathbf{G}(n)\left[\mathbf{C}(n)\epsilon(n, n-1) + \mathbf{v}_2\right] \\ =& \left[\mathbf{I} - \mathbf{G}(n)\mathbf{C}(n)\right]\epsilon(n, n-1) - \mathbf{g}(n)\mathbf{v}_2(n)\end{aligned}$$

which, after some manipulation, yields the desired formula:

$$\mathbf{K}(n) = \mathbf{K}(n, n-1) - \mathbf{F}(n, n+1)\mathbf{G}(n)\mathbf{C}(n)\mathbf{K}(n, n-1) \tag{13}$$

The algorithm for computing the MAP estimate $\hat{\mathbf{x}}_{\text{MAP}}(n)$ is now complete. It is made up of Equations (10) through (13). Indeed, comparing these equations with the Kalman filtering algorithm summarized in Table 14.2 of the textbook, we find that the MAP estimate $\hat{\mathbf{x}}_{\text{MAP}}(n)$ is nothing but the filtered estimate $\hat{\mathbf{x}}(n)$ in the standard Kalman filter theory.

d)
Finally the second-order derivative, namely $\partial^2 \ln f_X(\mathbf{x}(n)|\mathbf{Y}_n)/\partial^2 \mathrm{x}(n)$ is defined by

$$\mathbf{K}^{-1}(n, n-1)\hat{\mathbf{x}}(n, n-1)$$

This expression is always negative. Hence, the condition of Equation (2), is satisfied by the MAP estimate, thereby completing the solution to Problem 10.8. $\hat{\mathbf{x}}_{\text{MAP}}(n)$

Problem 14.9

We are given the noiseless sytem:

$$\mathbf{x}(n+1) = \mathbf{F}\mathbf{x}(n) \tag{1}$$

$$\mathbf{y}(n) = \mathbf{C}\mathbf{x}(n) \tag{2}$$

a) Show that

$$\hat{\mathbf{x}}(n|\mathbf{Y}_n) = \mathbf{F}\left(\mathbf{I} - \mathbf{G}(n)\mathbf{C}\right)\hat{\mathbf{x}}(n|\mathbf{Y}_{n-1}) + \mathbf{C}\mathbf{G}(n)\mathbf{y}(n) \tag{3}$$

$$\alpha(n) = \mathbf{y}(n) - \mathbf{C}\hat{\mathbf{x}}(n|\mathbf{Y}_{n-1}) \tag{4}$$

By definition, the innovation is

$$\alpha(n) = \mathbf{y}(n) - \hat{\mathbf{y}}(n|\mathbf{Y}_{n-1})$$

Following the same reasoning used to derive Equation (14.31) of the textbook, we have

$$\hat{\mathbf{y}}(n|\mathbf{Y}_{n-1}) = \mathbf{C}\hat{\mathbf{x}}(n|\mathbf{Y}_{n-1})$$

which confirms the second equation of the problem statement.

Next, using Equations (2), (5), and (6), we may write

$$\begin{aligned}\alpha(n) =&\mathbf{C}\mathbf{x}(n) - \mathbf{C}\hat{\mathbf{x}}(n)\\ =&\mathbf{C}\left[\mathbf{x}(n) - \hat{\mathbf{x}}(n|\mathbf{Y}_{n-1})\right]\\ =&\mathbf{C}\varepsilon(n, n-1)\end{aligned}$$

On the basis that we go on to write

$$\begin{aligned}\mathbf{R}(n) =&\mathbb{E}\left[\alpha(n)\alpha^H(n)\right]\\ =&\mathbf{C}\mathbb{E}\left[\underset{\sim}{\varepsilon}(n, n-1)\underset{\sim}{\varepsilon}^H(n, n-1)\right]\mathbf{C}^H\\ =&\mathbf{C}\mathbf{K}(n, n-1)\mathbf{C}^H\end{aligned}$$

Similar to Equations (14.45) and (14.49) from the textbook, we may write:

$$\hat{\mathbf{x}}(n+1|\mathbf{Y}_{n-1}) = \mathbf{F}\hat{\mathbf{x}}(n|\mathbf{Y}_{n-1}) + \mathbf{G}(n)\alpha(n)$$

$$\mathbf{G}(n) =\mathbf{F}\mathbf{K}(n, n-1)\mathbf{C}^H\mathbf{R}^{-1}(n)$$

where $\mathbf{G}(n)$ is the definition questioned in the Problem statement.

Moreover in a manner, similar to Equations (14.59), (14.45) and (14.35), we may finally write:

$$\begin{aligned}\hat{\mathbf{x}}(n|\mathbf{Y}_n) =&\mathbf{F}^{-1}\hat{\mathbf{x}}(n+1|\mathbf{Y}_n)\\ =&\hat{\mathbf{x}}(n|\mathbf{Y}_{n-1}) + \mathbf{F}^{-1}\mathbf{G}(n)\alpha(n)\\ =&\hat{\mathbf{x}}(n|\mathbf{Y}_{n-1}) + \mathbf{F}^{-1}\mathbf{G}(n)\left(\mathbf{y}(n) - \mathbf{C}\tilde{\mathbf{x}}\left(n|\mathbf{Y}(n-1)\right)\right)\\ =&\left(\mathbf{I} - \mathbf{F}^{-1}\mathbf{G}(n)\mathbf{C}\right)\hat{\mathbf{x}}(n|\mathbf{Y}_{n-1}) + \mathbf{F}^{-1}\mathbf{G}(n)\mathbf{y}(n)\end{aligned}$$

Note also, in the third line we used the $\alpha(n)$ as follows:

$$\alpha(n) = \mathbf{y}(n) - \mathbf{C}\tilde{\mathbf{x}}(n|\mathbf{Y}_{n-1})$$

b)

The innovation sequence $\alpha(1), \alpha(2), \ldots, \alpha(n)$ consists of samples that are orthogonal to each other; see Property 2, p542 of the text. Orthogonality is synonymous with whitening. Moreover, invoking the very definition of the innovation process, we may refer to the Kalman filter as a whitening filter.

Problem 14.10

Let us consider the unforced dynamical model

$$\mathbf{x}(n+1) = \mathbf{F}(n+1,n)\mathbf{x}(n) \tag{1}$$

$$y(n) = \mathbf{u}^H(n)\mathbf{x}(n) + \nu(n) \tag{2}$$

where $\mathbf{F}(n+1,n) = \dfrac{1}{\sqrt{\lambda}}\mathbf{I}$

From Equation (1), we see that

$$\mathbf{x}(n) = \lambda^{-n/2}\mathbf{x}(0) \tag{3}$$

a)

Considering a deterministic multiple regression model $d(n) = e_0(n) + \mathbf{w}_0^H\mathbf{u}(n)$, we may write:

$$\left.\begin{array}{rcl} d^*(0) & = & \mathbf{u}^H(0)\mathbf{w}_0 + e_0^*(0) \\ d^*(1) & = & \mathbf{u}^H(1)\mathbf{w}_0 + e_0^*(1) \\ \vdots & \vdots & \vdots \\ d^*(n) & = & \mathbf{u}^H(n)\mathbf{w}_0 + e_0^*(n) \end{array}\right\}$$

which represents a deterministic system of linear equations

b)

From Equations (1), (2) and (3), we have:

$$\left.\begin{array}{c} y(0) = \mathbf{u}^H(0)\mathbf{x}(0) + \nu(0) \\ y(1) = \mathbf{u}^H(1)\mathbf{x}(0)\lambda^{-1/2} + \nu(1) \\ \vdots \quad\quad \vdots \quad\quad \vdots \\ y(n) = \mathbf{u}^H(n)\mathbf{x}(0)\lambda^{-n/2} + \nu(n) \end{array}\right\} \Leftrightarrow \left\{\begin{array}{c} y(0) = \mathbf{u}^H(0)\mathbf{x}(0) + \nu(0) \\ \lambda^{1/2}y(1) = \mathbf{u}^H(1)\mathbf{x}(0) + \lambda^{1/2}\nu(1) \\ \vdots \quad\quad \vdots \quad\quad \vdots \\ \lambda^{n/2}y(n) = \mathbf{u}^H(n)\mathbf{x}(0) + \lambda^{n/2}\nu(n) \end{array}\right.$$

which represents a stochastic system of linear equations.

c)

Both the stochastic and the deterministic system of linear simultaneous equations describe the same problem, and therefore should have a common solution. Comparing the two systems, we set

$$\left\{\begin{array}{c} \mathbf{x}(0) = \mathbf{w}_o \\ y(n) = \lambda^{-n/2}d^*(n) \\ \nu(n) = \lambda^{-n/2}e_0^*(n) \end{array}\right.$$

Problem 14.11

The reason for the RLS operating satisfactorily is the fact that the minimum mean-square error follows the recursion (see Equation (10.39) in the textbook)

$$\mathscr{E}_{\text{min}}(n) = \lambda\mathscr{E}_{\text{min}}(n-1) + \xi^*(n)e(n)$$

Hence, with λ in the interval $0 < \lambda \leq 1$, the stored value of the minimum mean-square error, $\mathscr{E}_{\text{min}}(n-1)$, is reduced by the fact λ as the minimum mean-square error is recursively updated.

Problem 14.12

Consider first the correspondence between $\hat{\mathbf{x}}(1|\mathbf{Y}_0)$ and $\hat{\mathbf{w}}(0)$. For the one-step prediction, we have

$$\hat{\mathbf{x}}(n+1|\mathbf{Y}_n) : \lambda^{-n/2}\hat{\mathbf{w}}(n)$$

Therefore, setting $n = 0$, we see that $\hat{\mathbf{x}}(1|\mathbf{Y}_0)$ in the Kalman filter corresponds to $\hat{\mathbf{w}}(0)$ in the RLS algorithm.

Consider next the correlation matrix of the error in state prediction, which is defined by

$$\mathbf{K}(n) : \lambda^{-1}\mathbf{P}(n)$$

Putting $n = 0$, we readily see that $\mathbf{K}(0)$ in the Kalman filter corresponds to $\lambda^{-1}\mathbf{P}(0)$ in the RLS algorithm.

Problem 14.13

Let

$$\mathbf{K}(n) = \mathbf{K}^{1/2}(n)\mathbf{K}^{H/2}(n)$$

Hence,

$$\chi(\mathbf{K}) \leq \chi(\mathbf{K}^{1/2})\chi(\mathbf{K}^{H/2}) = \left(\chi(\mathbf{K}^{1/2})\right)^2$$

The implication of this result is that the condition number of the square root $\mathbf{K}^{1/2}(n)$ is the square root of the condition number of the original matrix $\mathbf{K}(n)$.

Problem 14.14

The condition number of $\mathbf{K}(n)$ is

$$\chi(\mathbf{K}) = \frac{\lambda_{\text{max}}}{\lambda_{\text{min}}}$$

Given that

$$\begin{aligned}\chi(\mathbf{K}) =& \chi(\mathbf{U}\mathbf{D}\mathbf{U}^H) \\ \leq& \chi(\mathbf{U})\chi(\mathbf{D})\chi(\mathbf{U}^H)\end{aligned} \tag{1}$$

The eigenvalues of an upper triangular matrix are the same as the diagonal elements of the matrix. For the situation at hand, the upper triangular matrix has 1's for all its diagonal elements. Hence, the λ_{max} and λ_{min} of $\mathbf{U}(n)$ are both equal to one, and so

$$\chi(\mathbf{U}) = \chi(\mathbf{U}^H) = 1$$

Accordingly, Equation (1) simplifies to

$$\chi(\mathbf{D}) \geq \chi(\mathbf{K})$$

Problem 14.15

The square-root implementation of a Kalman filter requires more computation than the conventional Kalman filter. The problem of computational efficiency led to the development of a modified version of the square-root filtering algorithm known as the UD-factorization algorithm. In this second approach, the filtered state-error correlation matrix $\mathbf{K}(n)$ is factored into an upper triangular matrix $\mathbf{U}(n)$ with 1's along its main diagonal and a real diagonal matrix $\mathbf{D}(n)$, as shown by:

$$\mathbf{K}(n) = \mathbf{U}(n)\mathbf{D}(n)\mathbf{U}^H(n)$$

Equivalently, the factorization may be written as

$$\mathbf{K}(n) = \left(\mathbf{U}(n)\mathbf{D}^{1/2}(n)\right)\left(\mathbf{U}(n)\mathbf{D}^{1/2}(n)\right)^H$$

where $\mathbf{D}^{1/2}(n)$ is the square root of $\mathbf{D}(n)$. The nonnegative definiteness of the computed matrix $\mathbf{K}(n)$ is guaranteed by updating the factors $\mathbf{U}(n)$ and $\mathbf{D}(n)$ instead of $\mathbf{K}(n)$ itself. However, a Kalman filter based on the UD-factorization does not possess the numerical

advantage of a standard square-root Kalman filter. Moreover, a Kalman filter using UD-factorization may suffer from serious overflow and underflow problems.

One final comment is in order: With the ever-increasing improvements in digital technology, the old argument that square roots are expensive and awkward to calculate is no longer as compelling as it used to be in yesteryears.

Chapter 15

Problem 15.1

To derive the square-root information filter, we proceed as follows. Let

$$\mathbf{A}(n) = \begin{bmatrix} \lambda^{1/2}\mathbf{K}^{-H/2}(n-1) & \lambda^{1/2}\mathbf{u}(n) \\ \hat{\mathbf{x}}^H(n|\mathbf{Y}_{n-1})\mathbf{K}^{-H/2}(n-1) & y^*(n) \\ \mathbf{0}^T & 1 \end{bmatrix}$$

Taking the Hermitian transpose:

$$\mathbf{A}^H(n) = \begin{bmatrix} \lambda^{1/2}\mathbf{K}^{-1/2}(n-1) & \mathbf{K}^{-1/2}(n-1)\hat{\mathbf{x}}(n|\mathbf{Y}_{n-1}) & \mathbf{0} \\ \lambda^{1/2}\mathbf{u}^H(n) & y(n) & 1 \end{bmatrix}$$

Hence, postmultiplying $\mathbf{A}(n)$ by $\mathbf{A}^H(n)$:

$$\mathbf{A}(n)\mathbf{A}^H(n) = \begin{bmatrix} \lambda\mathbf{K}^{-1}(n-1) + \lambda\mathbf{u}(n)\mathbf{u}^H(n) & \lambda^{1/2}\mathbf{K}^{-1}(n-1)\hat{\mathbf{x}}(n|\mathbf{Y}_{n-1}) + \lambda^{1/2}\mathbf{u}(n)y(n) & \lambda^{1/2}\mathbf{u}(n) \\ \begin{pmatrix} \lambda^{1/2}\hat{\mathbf{x}}(n|\mathbf{Y}_{n-1})\mathbf{K}^{-1}(n-1) \\ +\lambda^{1/2}\mathbf{u}^H(n)y^*(n) \end{pmatrix} & \lambda\hat{\mathbf{x}}^H(n|\mathbf{Y}_{n-1})\mathbf{K}^{-1}(n-1)\hat{\mathbf{x}}(n-1|\mathbf{Y}_n) & y^*(n) \\ \lambda^{1/2}\mathbf{u}^H(n) & y(n) & 1 \end{bmatrix}$$

This 2-by-2 matrix pertains to the pre-array of the square-root information filter.

Consider next the post-array of the square-root information filter. Let

$$\mathbf{B}(n) = \begin{bmatrix} \mathbf{B}_{11}(n) & \mathbf{b}_{21}(n) & \mathbf{b}_{31}(n) \\ \mathbf{0}^T & \mathbf{b}_{22}(n) & \mathbf{b}_{32}(n) \end{bmatrix}$$

Taking the Hermitian transpose:

$$\mathbf{B}^H(n) = \begin{bmatrix} \mathbf{B}_{11}^H(n) & \mathbf{0} \\ \mathbf{b}_{21}^H(n) & \mathbf{b}_{22}^*(n) \\ \mathbf{b}_{31}^H(n) & \mathbf{b}_{32}^*(n) \end{bmatrix}$$

Hence, pre-multiplying $\mathbf{B}(n)$ by $\mathbf{B}^H(n)$:

$$\mathbf{B}^H(n)\mathbf{B}(n) = \begin{bmatrix} \mathbf{B}_{11}^H(n)\mathbf{B}_{11}(n) & \mathbf{B}_{11}^H(n)\mathbf{b}_{21}(n) & \mathbf{B}_{11}(n)\mathbf{b}_{31}^H(n) \\ \mathbf{b}_{21}^H(n)\mathbf{B}_{11}(n) & \mathbf{b}_{21}^H(n)\mathbf{b}_{21}(n) + |\mathbf{b}_{22}(n)|^2 & \mathbf{b}_{21}^H(n)\mathbf{b}_{31}(n) + \mathbf{b}_{22}^*(n)\mathbf{b}_{32}(n) \\ \mathbf{b}_{31}^H(n)\mathbf{B}_{11}(n) & \mathbf{b}_{31}^H(n)\mathbf{b}_{31}(n) + \mathbf{b}_{32}^*(n)\mathbf{b}_{22}(n) & \mathbf{b}_{31}^H(n)\mathbf{b}_{31}(n) + |\mathbf{b}_{32}(n)|^2 \end{bmatrix}$$

Equating terms in $\mathbf{A}(n)\mathbf{A}^H(n)$ to corresponding terms in $\mathbf{B}^H(n)\mathbf{B}(n)$, we get,

1.

$$\begin{aligned} \mathbf{B}_{11}^H(n)\mathbf{B}_{11}(n) =& \lambda\mathbf{K}^{-1}(n-1) + \lambda\mathbf{u}(n)\mathbf{u}^H(n) \\ =& \mathbf{K}^{-1}(n) \\ =& \mathbf{K}^{-H/2}(n)\mathbf{K}^{-1/2}(n) \end{aligned}$$

Hence,

$$\mathbf{B}_{11}^H = \mathbf{K}^{-H/2}(n)$$

2.

$$\begin{aligned} \mathbf{B}_{11}^H(n)\mathbf{b}_{21}(n) =& \lambda^{1/2}\mathbf{K}^{-1}(n-1)\hat{\mathbf{x}}(n|\mathbf{Y}_{n-1}) + \lambda^{1/2}\mathbf{u}(n)y(n) \\ =& \mathbf{K}^{-1}(n)\hat{\mathbf{x}}(n+1|\mathbf{Y}_n) \end{aligned}$$

Hence,

$$\mathbf{b}_{21}(n) = \mathbf{K}^{-H/2}(n)\hat{\mathbf{x}}(n+1|\mathbf{Y}_n)$$

3.

$$\mathbf{B}_{11}^H(n)\mathbf{b}_{31}(n) = \lambda^{1/2}\mathbf{u}(n)$$

Hence,

$$\mathbf{b}_{31}(n) = \lambda^{1/2}\mathbf{K}^{H/2}(n)\mathbf{u}(n)$$

4.

$$\mathbf{b}_{31}^H(n)\mathbf{b}_{31}(n) + |\mathbf{b}_{32}(n)|^2 = 1$$

Hence,

$$\begin{aligned}|\mathbf{b}_{32}(n)|^2 =& 1 - \lambda \mathbf{u}^H(n)\mathbf{K}(n)\mathbf{u}(n)\\ =& r^{-1}(n)\end{aligned}$$

That is

$$\mathbf{b}_{32}(n) = r^{-1/2}(n)$$

5.

$$\mathbf{b}_{21}^H(n)\mathbf{b}_{21}(n) + |\mathbf{b}_{22}(n)|^2 = \hat{\mathbf{x}}^H(n|\mathbf{Y}_{n-1})\mathbf{K}^{-1}(n-1)\hat{\mathbf{x}}(n-1|\mathbf{Y}_n) + |y(n)|^2$$

Hence,

$$\begin{aligned}|\mathbf{b}_{22}(n)|^2 =& \hat{\mathbf{x}}(n|\mathbf{Y}_{n-1})\mathbf{K}^{-1}(n-1)\hat{\mathbf{x}}(n|\mathbf{Y}_{n-1}) + |y(n)|^2\\ &- \hat{\mathbf{x}}(n+1|\mathbf{Y}_n)\mathbf{K}^{-1}(n)\hat{\mathbf{x}}(n+1|\mathbf{Y}_n)\end{aligned} \tag{1}$$

6.

$$\mathbf{b}_{31}^H(n)\mathbf{b}_{21}(n) + \mathbf{b}_{32}^*(n)\mathbf{b}_{22}(n) = y(n)$$

Hence,

$$r^{-1/2}(n)\mathbf{b}_{22}(n) = y(n) - \lambda^{1/2}\mathbf{u}^H(n)\mathbf{K}^{1/2}(n)\mathbf{K}^{-1/2}(n)\hat{\mathbf{x}}(n+1|\mathbf{Y}_n)$$

That is,

$$\mathbf{b}_{22}(n) = r^{1/2}(n)\left[y(n) - \lambda^{1/2}\mathbf{u}^H(n)\hat{\mathbf{x}}(n+1|\mathbf{Y}_n)\right]$$

But, we know that

$$\hat{\mathbf{x}}(n+1|\mathbf{Y}_n) = \lambda^{-1/2}\hat{\mathbf{x}}(n|\mathbf{Y}_{n-1}) + \alpha(n)\mathbf{g}(n)$$

where $\mathbf{g}(n)$ is the Kalman gain and $\alpha(n)$ is the innovation. Therefore,

$$\begin{aligned}\mathbf{b}_{22}(n) =& r^{1/2}(n)\left[y(n) - \mathbf{u}^H(n)\hat{\mathbf{x}}(n|\mathbf{Y}_{n-1}) - \lambda^{1/2}\alpha(n)\mathbf{u}^H(n)\mathbf{g}(n)\right]\\ =& r^{1/2}(n)\left[\alpha(n) - \lambda^{1/2}\alpha(n)\mathbf{u}^H(n)\mathbf{g}(n)\right]\\ =& r^{1/2}(n)\alpha(n)\left[2 - \lambda^{1/2}\mathbf{u}^H(n)\mathbf{g}(n)\right]\end{aligned}$$

But,

$$\lambda^{1/2}\mathbf{g}(n) = \mathbf{K}^{-1}(n)\mathbf{u}(n) \cdot r^{-1}(n)$$

Therefore,

$$\begin{aligned}\mathbf{b}_{22}(n) =& r^{1/2}(n)\alpha(n)\left[1 - \mathbf{u}^H(n)\mathbf{K}^{-1}(n)\mathbf{u}(n)r^{-1}(n)\right] \\ =& r^{-1/2}(n)\alpha(n)\left[r^{1/2}(n) + \mathbf{u}^H(n)\mathbf{K}^{-1}(n)\mathbf{u}(n)\right] \\ =& r^{-1/2}(n)\alpha(n) \qquad (2)\end{aligned}$$

where we have used $r(n) = 1 + \mathbf{u}^H(n)\mathbf{K}^{-1}(n)\mathbf{u}(n)$.

Final Check In a laborious way, it can be shown that Equation (1) is satisfied exactly by the value of $\mathbf{b}_{22}(n)$ defined by Equation (2).

Problem 15.2

Consider a linear dynamic system described by the state-space model

$$\mathbf{x}(n+1) = \lambda^{-1/2}\mathbf{x}(n) \qquad (1)$$

$$y(n) = \mathbf{u}^H(n)\mathbf{x}(n) + \nu(n) \qquad (2)$$

where $\nu(n)$ is a Gaussian variable of zero mean and variance $Q(n)$. The Kalman filtering algorithm for the model is described by:

$$\mathbf{g}(n) = \lambda^{-1/2}\mathbf{K}(n-1)\mathbf{u}(n)R^{-1}(n)$$

$$R(n) = \mathbf{u}^H(n)\mathbf{K}(n-1)\mathbf{u}(n) + Q(n)$$

$$\alpha(n) = y(n) - \mathbf{u}^H(n)\hat{\mathbf{x}}(n|\mathbf{Y}_{n-1})$$

$$\hat{\mathbf{x}}(n+1|\mathbf{Y}_n) = \lambda^{-1/2}\hat{\mathbf{x}}(n|\mathbf{Y}_{n-1}) + \mathbf{g}(n)\alpha(n)$$

$$\mathbf{K}(n) = \lambda^{-1}\mathbf{K}(n-1) - \lambda^{-1/2}\mathbf{g}(n)\mathbf{u}^H(n)\mathbf{K}(n-1)$$

Then proceeding in a manner similar to that described within the chaper, we may formulate the extended square-root information filter for the state-space model of Equations (1) and (2) as follows:

$$\mathbf{K}^{-1}(n) = \lambda\left[\mathbf{K}^{-1}(n-1) + Q^{-1}(n)\mathbf{u}(n)\mathbf{u}^H(n)\right] \qquad (3)$$

$$\mathbf{K}^{-1}(n)\hat{\mathbf{x}}(n+1|\mathbf{Y}_n) = \lambda^{-1/2}\left[\mathbf{K}^{-1}(n-1)\hat{\mathbf{x}}(n|\mathbf{Y}_{n-1}) + Q^{-1}(n)\mathbf{u}(n)y(n)\right] \qquad (4)$$

where $\mathbf{g}(n)$ is the Kalman gain (vector).

Thus, in light of Equations (3) and (4), we may formulate the following array structure for the square-root information filter:

$$\begin{bmatrix} \lambda^{1/2}\mathbf{K}^{-H/2}(n-1) & \lambda^{-1/2}Q^{-1/2}(n)\mathbf{u}(n) \\ \hat{\mathbf{x}}(n|\mathbf{Y}_{n-1})\mathbf{K}^{-H/2}(n-1) & Q^{-1/2}(n)y(n) \\ \mathbf{0}^T & Q^{-1/2}(n) \end{bmatrix} \Theta(n) =$$

$$\begin{bmatrix} \mathbf{K}^{-H/2}(n) & \mathbf{0} \\ \hat{\mathbf{x}}(n|\mathbf{Y}_{n-1})\mathbf{K}^{-H/2}(n-1) & R^{-1/2}(n)\alpha^*(n) \\ \lambda^{-1/2}Q^{-1}(n)\mathbf{u}^H(n)\mathbf{K}^{1/2}(n) & R^{-1/2}(n) \end{bmatrix}$$

The equation includes an ordinary square-root information filter as a special case. Specifically, putting $Q(n) = 0$ we get the pre-array-to-post-array transformation for that filter.

Problem 15.3

a)

Verify the expression for the extended square-root information filter given by

$$\begin{bmatrix} \lambda^{1/2}\mathbf{K}^{-H/2}(n-1) & \lambda^{-1/2}\mathbf{u}(n) \\ \hat{\mathbf{x}}(n|\mathbf{Y}_{n-1})\mathbf{K}^{-H/2}(n-1) & y^*(n) \\ \mathbf{0}^T & 1 \\ \lambda^{1/2}\mathbf{K}^{1/2}(n-1) & \mathbf{0} \end{bmatrix} \Theta(n) =$$

$$\begin{bmatrix} \mathbf{K}^{-H/2}(n-1) & \mathbf{0} \\ \hat{\mathbf{x}}(n+1|\mathbf{Y}_n)\mathbf{K}^{-H/2}(n-1) & r^{1/2}(n)\alpha^*(n) \\ \lambda^{1/2}\mathbf{u}^H(n)\mathbf{K}^{1/2}(n) & r^{1/2}(n) \\ \mathbf{K}^{1/2}(n) & -\mathbf{g}(n)r^{1/2}(n) \end{bmatrix}$$

Let

$$\mathbf{A}(n) = \begin{bmatrix} \lambda^{1/2}\mathbf{K}^{-H/2}(n-1) & \lambda^{-1/2}\mathbf{u}(n) \\ \hat{\mathbf{x}}(n|\mathbf{Y}_{n-1})\mathbf{K}^{-H/2}(n-1) & y^*(n) \\ \mathbf{0}^T & 1 \\ \lambda^{1/2}\mathbf{K}^{1/2}(n-1) & \mathbf{0} \end{bmatrix}$$

$$\mathbf{B}^H(n) = \begin{bmatrix} \mathbf{B}_{11}^H(n) & \mathbf{0} \\ \mathbf{b}_{21}^H(n) & \mathbf{b}_{22}^*(n) \\ \mathbf{b}_{31}^H(n) & b_{32}^*(n) \\ \mathbf{B}_{41}^H(n) & \mathbf{B}_{42}^H(n) \end{bmatrix}$$

Then, equating the matrix product $\mathbf{A}(n)\mathbf{A}^H(n)$ to the matrix product $\mathbf{B}^H(n)\mathbf{B}(n)$, and comparing their corresponding terms, we may say the following:

- Points 1 through 6 discussed in the solution to Problem 15.1 remain valid. Accordingly, the entries $\mathbf{B}_{11}(n), \mathbf{b}_{21}(n), \mathbf{b}_{31}(n), \mathbf{b}_{22}(n)$, and $\mathbf{b}_{32}(n)$ will all have the value determined there.
- We have the following additional equations to consider:

1.

$$\mathbf{B}_{41}^H(n)\mathbf{B}_{11}^H(n) = \mathbf{I}$$

since $\mathbf{B}_1 1(n) = \mathbf{K}^{-1/2}(n)$, it follows that

$$\mathbf{B}_{41}^H(n) = \mathbf{K}^{1/2}(n) \tag{1}$$

2.

$$\mathbf{B}_{42}^H(n) = \mathbf{B}_{41}^H(n)\mathbf{b}_{31}(n)/\mathbf{b}_3 2(n)$$

Since $\mathbf{b}_{32}(n) = r^{-1/2}(n)$ and $\mathbf{b}_{31}^H(n) = \lambda^{1/2}\mathbf{u}^H(n)\mathbf{K}^{1/2}(n)$, it follows that

$$\begin{aligned} \mathbf{B}_{42}^H(n) =& \lambda^{1/2} r^{1/2}(n)\mathbf{K}^{1/2}(n)\mathbf{K}^{-H/2}(n)\mathbf{u}(n) \\ =& -\lambda^{1/2} r^{1/2}(n)\mathbf{K}(n)\mathbf{u}(n) \end{aligned} \tag{2}$$

But from Kalman filter theory for the dynamical system being considered here, we know that

$$\lambda^{1/2}\mathbf{K}(n)\mathbf{u}(n) = \mathbf{g}(n)$$

Accordingly, we may simplify Equation (2) to

$$\mathbf{B}_{42}^H(n) = -r^{1/2}(n)\mathbf{g}(n)$$

where $\mathbf{g}(n)$ is the Kalman gain. This completes the evaluation of the post-array for the extended square-root information filter.

3. In addition tot he two equation described above, we have several other relations that follow from $\mathbf{A}(n)\mathbf{A}^H(n) = \mathbf{B}^H(n)\mathbf{B}(n)$; these additional relations merely provide a means to check the value already determined for the entries of the post-array.

b)
Using the results from part **a)**, derive the extended QR-RLS algorithm.

In order to derive the extended QR-RLS algorithm we make use of the following relationships between the Kalman and RLS variables:

$$\begin{aligned} &\mathbf{K}^{-1} \rightarrow \lambda\Phi(n) && \alpha(n) \rightarrow \lambda^{-n/2}\xi^*(n) \\ &r^{-1}(n) \rightarrow \gamma(n) && y(n) \rightarrow \lambda^{-n/2}d^*(n) \\ &\mathbf{g}(n) \rightarrow \lambda^{-1/2}\mathbf{k}(n) && \hat{\mathbf{x}}(n|\mathbf{Y}_{n-1}) \rightarrow \lambda^{-n/2}\hat{\mathbf{w}}(n-1) \end{aligned}$$

Therefore, the extended QR-RLS could be written as:

$$\begin{bmatrix} \lambda^{1/2}\Phi^{1/2}(n) & \mathbf{u}(n) \\ \lambda^{1/2}\mathbf{p}^H(n-1) & d(n) \\ \mathbf{0}^T & 1 \\ \lambda^{-1/2}\Phi^{H/2}(n-1) & \mathbf{0} \end{bmatrix} \Theta(n) = \begin{bmatrix} \Phi^{1/2}(n) & \mathbf{0} \\ \mathbf{p}^H(n) & \xi(n)\gamma^{1/2}(n) \\ \mathbf{u}^H(n)\Phi^{-H/2}(n) & \gamma^{1/2}(n) \\ \Phi^{-H/2}(n) & -\mathbf{k}(n)\gamma^{-1/2}(n) \end{bmatrix}$$

where we used the same reasoning as while obtaining Equations (15.40)-(15.47) in the textbook

$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1) + \left[\mathbf{k}(n)\gamma^{-1/2}(n)\right]\left[\xi(n)\gamma^{-1/2}(n)\right]^*$$

Problem 15.4

We start with the matrix relation:

$$\begin{bmatrix} \lambda^{1/2}\Phi^{1/2}(n) & \mathbf{u}(n) \\ \lambda^{1/2}\mathbf{p}^H(n-1) & d(n) \\ \mathbf{0}^T & 1 \\ \lambda^{-1/2}\Phi^{H/2}(n-1) & \mathbf{0} \end{bmatrix} \Theta(n) = \begin{bmatrix} \Phi^{1/2}(n) & \mathbf{0} \\ \mathbf{p}^H(n) & \xi(n)\gamma^{1/2}(n) \\ \mathbf{u}^H(n)\Phi^{-H/2}(n) & \gamma^{1/2}(n) \\ \Phi^{-H/2}(n) & -\mathbf{k}(n)\gamma^{-1/2}(n) \end{bmatrix} \tag{1}$$

where $\Theta(n)$ is unitary rotation.

Let

$$\mathbf{X}(n) = \Phi^{-H/2}(n) + \eta_x(n) \tag{2}$$

where $\eta_x(n)$ represents the effect of round-off error.

Assuming there are no additional local errors introduced at time n, the recuresion pertaining to the bottom parts of the prearray and postarray of Equation (1) takes on the following form:

$$\left[\lambda^{-1/2}\mathbf{X}(n-1), \quad \mathbf{0}\right]\Theta(n) = [\mathbf{X}(n), \quad -\mathbf{y}(n)] \tag{3}$$

The vector $\mathbf{y}(n)$ is the quantized version of $\mathbf{k}(n)\gamma^{-1/2}(n)$:

$$\mathbf{y}(n) = \mathbf{k}(n)\gamma^{-1/2}(n) + \eta_y(n) \tag{4}$$

where $\mathbf{k}(n)$ is the gain vector, $\gamma(n)$ is the conversion factor, and $\eta_y(n)$ is the round-off error.

Substituting Equations (2) and (4) into Equation (3), we may express the quantized version of the last rows in Equation (1) as follows:

$$\begin{aligned}\left[\lambda^{-1/2}\Phi^{-H/2}(n-1) + \lambda^{-1/2}\eta_x(n-1), \quad \mathbf{0}\right]\Theta(n) \\ = \left[\Phi^{-H/2}(n) + \eta_x(n), \ -\mathbf{k}(n)\gamma^{-1/2}(n) - \eta_y(n)\right]\end{aligned} \tag{5}$$

Under infinite-precision arithmetic, we have from the last rows of Equation (1):

$$\left[\lambda^{-1/2}\Phi^{-H/2}(n-1) \quad \mathbf{0}\right]\Theta(n) = \left[\Phi^{-H/2}(n) \quad -\mathbf{k}(n)\gamma^{-1/2}(n)\right] \tag{6}$$

Hence, comparing Equations (5) and (6), we infer

$$\left[\lambda^{-1/2}\eta_x(n-1) \quad \mathbf{0}\right]\theta(n) = [\eta_x(n) \quad -\eta_y(n)] \tag{7}$$

Equation (7) reveals that the error propagation due to $eta_x(n-1)$ is NOT necessarily stable, in that the local errors tend to grow unboundedly. The unlimited error growth is due to Equation (1) the amplification produced by the factor $\lambda^{-1/2}$ for $\lambda < 1$, and Equation (2) the fact that the unitary rotation Θ is independent of the error eta_x. Consequently, as the recursion progresses the stored values of $\Phi^{1/2}$ and $\Phi^{-H/2}$ deviate more and more from each other's Hermitian transpose, thereby contradicting the very premise on which the extended QR-RLS algorithm is based.

Problem 15.5

We start with

$$\mathbf{Q}(n)\mathbf{A}(n) = \begin{bmatrix}\mathbf{R}(n) \\ \mathbf{0}\end{bmatrix}$$

Let

$$\mathbf{Q}(n) = \begin{bmatrix} \mathbf{Q}_1(n) \\ \mathbf{Q}_2(n) \end{bmatrix}$$

Hence,

$$\begin{bmatrix} \mathbf{Q}_1(n) \\ \mathbf{Q}_2(n) \end{bmatrix} \mathbf{A}(n) = \begin{bmatrix} \mathbf{R}(n) \\ \mathbf{0} \end{bmatrix}$$

$$\mathbf{A}(n) = \begin{bmatrix} \mathbf{Q}_1^H(n) & \mathbf{Q}_2^H(n) \end{bmatrix} \begin{bmatrix} \mathbf{R}(n) \\ \mathbf{0} \end{bmatrix}$$

$$= \mathbf{Q}_1^H(n)\mathbf{R}(n)$$

$$\mathbf{A}^H(n) = \mathbf{R}^H(n)\mathbf{Q}_1(n)$$

The projection matrix is therefore

$$\mathbf{P}(n) = \mathbf{A}(n) \left(\mathbf{A}^H(n)\mathbf{A}(n)\right)^{-1} \mathbf{A}^H(n)$$

$$= \mathbf{Q}_1^H(n)\mathbf{R}(n) \left(\mathbf{R}^H(n)\mathbf{Q}_1(n)\mathbf{Q}_1^H(n)\mathbf{R}(n)\right)^{-1} \mathbf{R}^H(n)\mathbf{Q}_1^H(n)$$

Since $\mathbf{Q}_1(n)\mathbf{Q}_1^H(n) = \mathbf{I}$, we have

$$\mathbf{P}(n) = \mathbf{Q}_1^H(n)\mathbf{R}(n) \left(\mathbf{R}^H(n)\mathbf{R}(n)\right)^{-1} \mathbf{R}^H(n)\mathbf{Q}_1^H(n)$$

We also note that for an upper triangular matrix $\mathbf{R}(n)$,

$$\mathbf{R}(n) \left(\mathbf{R}^H(n)\mathbf{R}(n)\right)^{-1} \mathbf{R}^H(n) = \text{Identity matrix}$$

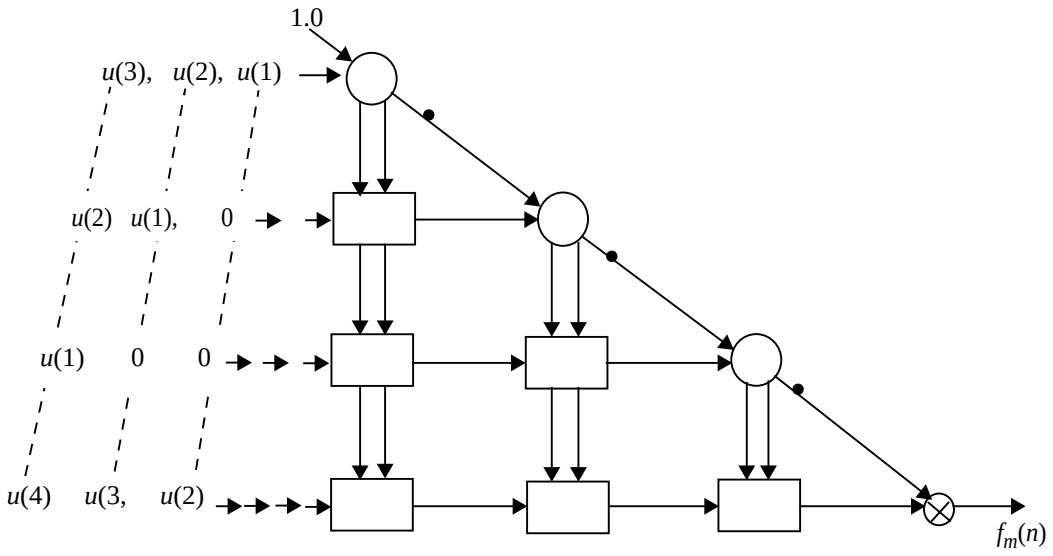
Hence,

$$\mathbf{P}(n) = \mathbf{Q}_1^H(n)\mathbf{Q}_1(n)$$

Problem 15.6

In a prediction-error filter, the input $u(n)$ represents the response and the tap input $u(n-1), \ldots, u(n-M)$ represent the variables used to estimate $u(n)$. Hence, we may restructure the inputs of the systolic array in Figure 14.4 of the text in the following

manner so that it operates as a prediction-error filter (illustrated here for order $M = 3$)



Prediction-error filter

Problem 15.7

The output of the last interval cell in the bottom row of the triangular section is given by (see Fig. 15.2(a) of the text):

$$u_{\text{out}} = cu_{\text{in}} - s^*\lambda^{1/2}x$$

where c and s are the Givens parameters, u_{in} is the input to the cell and x is the stored value to the cell. At time n, we have

$$u_{\text{in}}(n) = d(n)$$

$$s^*(n)\lambda^{1/2}x(n-1) = c(n)\hat{\mathbf{w}}^H(n-1)\mathbf{u}(n)$$

where $d(n)$ is the desired response, $\mathbf{u}(n)$ is the input vector, and $\hat{\mathbf{w}}(n-1)$ is the previous value of the least-squres weight vector. Hence,

$$\begin{aligned} u_{\text{out}} =& c(n)d(n) - c(n)\hat{\mathbf{w}}^H(n-1)\mathbf{u}(n) \\ =& c(n)\left[d(n) - \hat{\mathbf{w}}(n-1)\mathbf{u}(n)\right] \end{aligned}$$

Recognizing that

$$c(n) = \gamma^{1/2}(n)$$

and

$$d(n) - \hat{\mathbf{w}}^H(n-1)\mathbf{u}(n) = \xi(n)$$

we finally get

$$u_{\text{out}} = \gamma^{1/2}(n)\xi(n)$$

Problem 15.8

a) We note that

$$\mathbf{R}^H(n)\mathbf{a}(n) = \mathbf{s}(\phi)$$

Hence,

$$\mathbf{a}(n) = \mathbf{R}^{-H}(n)s(\phi)$$

Taking Hermitian transpose:

$$\mathbf{a}^H(n) = s^H(\phi)\mathbf{R}^{-1}(n)$$

b)
For an MVDR beamformer:

$$\mathbf{R}(n)\hat{\mathbf{w}}(n) = \frac{\mathbf{a}(n)}{\mathbf{a}^H(n)\mathbf{a}(n)}$$

Hence,

$$\hat{\mathbf{w}}(n) = \frac{\mathbf{R}^{-1}(n)\mathbf{a}(n)}{\mathbf{a}^H(n)\mathbf{a}(n)}$$

We note that $\mathbf{a}^H(n)\mathbf{a}(n)$ is a scalar. With $\mathbf{R}(n)$ being an upper triangular matrix, it follows that the linear section performs backward substitution. The resulting output of this section is the weight vector $\hat{\mathbf{w}}(n)$.

Problem 15.9

Reformulating the prearry of the extended QR-RLQ algorithm in Problem 2 of Chapter 15 by making use of the correspondences in Table 15.3, we may write

$$\begin{bmatrix} \lambda^{1/2}\Phi^{1/2}(n-1) & \mathbf{u}(n) \\ \lambda^{1/2}\mathbf{a}^H(n-1) & 0 \\ \mathbf{0}^T & 1 \end{bmatrix} \Theta(n) = \begin{bmatrix} \Phi^{1/2}(n) & \mathbf{0} \\ \mathbf{a}^H(n) & -e'(n)\gamma^{-1/2}(n) \\ \mathbf{u}^H(n)\Phi^{-H/2}(n) & \gamma^{1/2}(n) \end{bmatrix}$$

Squaring both sides of this equation, and retaining the terms on the second rows:

$$\lambda \|\mathbf{a}(n-1)\|^2 = \|\mathbf{a}(n)\|^2 + \gamma^{-1}(n)\,|e'(n)|^2$$

The term $e'(n)/\gamma^{1/2}(n)$ is recognized as an estimation error denoted by $\varepsilon(n)$. Hence,

$$\lambda \|\mathbf{a}(n-1)\|^2 = \|\mathbf{a}(n)\|^2 + |\varepsilon(n)|^2$$

Problem 15.10

The standard RLS filter is the covariance version of the Kalman filter. The inverse QR-RLS filter is the square-root version of the covariance Kalman filter. It follows therefore that the inverse QR-RLS filter is the square-root RLS filter.

Problem 15.11

```
% batch.m
make_rp;
rp.TNRdB = 10; rp.INRdB = 40; rp.Nsnaps= 200; rp.mu = 1e-10; rp.name = 'run1';
rp.sin_theta_2 = 0.05;run_qrd_rls_mvdr(rp);
rp.TNRdB = 10; rp.INRdB = 40; rp.Nsnaps= 200; rp.mu = 1e-10; rp.name = 'run2';
rp.sin_theta_2 = 0.15;run_qrd_rls_mvdr(rp);
rp.TNRdB = 10; rp.INRdB = 40; rp.Nsnaps= 200; rp.mu = 1e-10; rp.name = 'run3';
rp.sin_theta_2 = 0.2;run_qrd_rls_mvdr(rp);
plot_15_11;
```

```
rp.TNRdB = 10; % TNR in dB
rp.INRdB = 20; % INR in dB
rp.Nsnaps = 200;
```

```
rp.mu = 1e-8;
rp.p = 5;
rp.decay = 0;
rp.verbose = 0;
rp.mean_v = 0; % mean of complex-valued AWGN
rp.var_v = 1; % variance of complex-valued AWGN
rp.sin_theta_2 = -0.05;
rp.sin_theta_1 = 0;
rp.name = 'run1';


function [Wo, xp, gamma, e] = qrd_rls_AR_pred(Xi, Y, verbose, lambda, init, init_val)
% qrd_rls_AR_pred.m
% function [Wo, xp, gamma, e] = qrd_rls_AR_pred(Xi, Y, verbose, lambda, init, init_val)
%
% qrd_rls_AR_pred.m - use the QR decomposition-based RLS algorithm
% to predict complex-valued AR process
% written for MATLAB 4.0
%
% Reference: Haykin, _Adaptive Filter Theory_, 2nd (corr.) ed., 1991
%
%
% Input Parameters
% Xi : matrix of training/test points - each row is
% considered a datum
% Y : vector of corresponding desired outputs for
% predictor
% verbose : set to 1 for interactive processing
% lambda : the initial value of the forgetting factor
% init,
% init_val: initialization method and value
% mvdr' - init_val is steering vector
%
%
% Output Parameters:
% Wo : row-wise matrix of Hermitian transposed weights
% at each iteration
% xp : row vector of predicted outputs
% gamma : row vector of a priori to a posteriori conversion factors
% e : row vector of a posteriori prediction errors Y - xp
Nout = size(Xi, 2);
% length of maximum number of timesteps that can be predicted
N = size(Xi, 1);
% order of predictor
M = size(Xi, 2);
% initialize weight matrix and associated parameters
```

```
% for QRD RLS predictor
W = zeros(Nout, 1);
Wo = [];
gamma = [];
lambda_root = sqrt(lambda);
Phi_root = zeros(M, M);
p_H = zeros(1, M);
W_H = zeros(1, M);
z = zeros(1, M);
% initialize predictor until Phi_root is full-rank '
for n = 1:M,
    % compute post-array from pre-array via QRD. Note that computation
    % of p_H, eta, and W_H are omitted in the initialization period.
    u = Xi(n, :).';
    pre_array = [ [lambda_root*Phi_root u] ; [lambda_root*p_H Y(n)] ; [z 1] ];
    post_array = triu(qr(pre_array'))';
    Phi_root = post_array(1:M, 1:M);
    gamma_root = post_array(M+2, M+1);
    % save results
    Wo = [ Wo; W_H ];
    gamma = [gamma gamma_root^2 ];
    % predict next sample and compute error
    xp(n) = W_H * u;
    e(n) = Y(n) - xp(n);
    if (verbose ~= 0)
        disp(['time step ', int2str(n), ': mag. pred. err. = ' , num2str(abs(e(n)))]);
    end;
end % for n
% if initializing for MVDR adaptive beamforming, set the
% auxiliary vector p after Phi_root is full-rank
if (strcmp(init, 'mvdr')),
    p_H = (Phi_root \ init_val)';
end;
% run normally after initialization is complete
for n = (M+1):N,
    % compute post-array from pre-array via QRD
    u = Xi(n, :).';
    pre_array = [ [lambda_root*Phi_root u] ; [lambda_root*p_H Y(n)] ; [z 1] ];
    post_array = triu(qr(pre_array'))';
    Phi_root = post_array(1:M, 1:M);
    p_H = post_array(M+1, 1:M);
    ganuna_root = post_array(M+2, M+1);
    eta = post_array(M+1, M+1) / gamma_root;
    W_H = (Phi_root' \ p_H')';
    if (strcmp(init, 'mvdr')), W_H = W_H / (p_H * p_H'); end;
    % save results
```

```
    Wo = [Wo; W_H];
    gamma = [gamma gamma_root^2 ];
    % predict next sample and compute error
    xp(n) = W_H * u;
    e(n) = Y(n) - xp(n);
    if (verbose ~= 0)
        disp(['time step ', int2str(n), ': mag. pred. err. = ', num2str(abs(e(n)))]);
    end;
end % for n


function run_qrd_rls_mvdr(rp)
% run_qrd_rls_mvdr.m
% Computer Experiment
% Section 14.5, Adaptive Filter Theory, 3rd edition
% MVDR adaptive beamforming
% modify path to suit
% path(path, '/home/yee/aft/qrd_rls');
Ninit = rp.p;
Ndata = Ninit + rp.Nsnaps;
seed = 1;
lambda = 1;
% enter mean and variance of complex-valued AWGN
rp.mean_v = 0;
rp.var_v = 1;
% A_1, phi_1 are target signal amplitude/elec. angle
% A_2, phi_2 are interference signal amplitude/elec. angle
% s is steering vector along elec. angle of look direction of interest
A_1 = sqrt (rp.var_v) * 10^ (rp.TNRdB/20);
phi_1 = pi*rp.sin_theta_1;
A_2 = sqrt(rp.var_v) * 10^ (rp.INRdB/20);
phi_2 = pi*rp.sin_theta_2;
s = exp(-j*[0:(rp.p-1)]'*phi_1);
% setup input/output sequences
for i = 1:Ndata,
    % setup random disturbances
    randn('seed', i);
    vr = sqrt(rp.var_v/2) * randn(1, rp.p) + rp.mean_v;
    vi = sqrt(rp.var_v/2) * randn(1, rp.p) + rp.mean_v;
    v = vr + j*vi;
    rand('seed', i);
    %Psi = 2*pi*rand(1); % For Q 11.11
    Psi=0.2; %For Q 11.12
    Xi(i, :) = A_1*exp(j*[1:rp.p]*phi_1) + A_2*exp(j*[1:rp.p]*phi_2 + Psi) + v;
end;
Y = zeros(1, Ndata);
```

```
% run bea/nformer for indicated number of snapshots
[Wo, xp, gamma, e] = qrd_rls_AR_pred(Xi, Y, rp.verbose, lambda, 'mvdr', s);
% test vectors for spatially sampled response
W_H = Wo(Ndata, :);
st = -1:0.025:1;
est = exp(j*pi*[0:(rp.p-1)]'*st);
W=Wo; % simple renaming so that the format jives with thtat expected by the
% plot_mvdr.m routine
eval(['save ' rp.name])


%plot out the different weights found through simulations
figure
plot_mvdr('run1'); hold on
plot_mvdr('run2');
plot_mvdr('run3');

hold off
title('figure 15.12')
print -dpsc 15_11


function [] = plot_mvdr( name )
load(name)
    numst = 1000; % resolution in spatial response
%    W_H = conj(W(Ndata, :)); % Hemitian transpose of last one
    st = linspace(-1,1,numst); % sine(theta) space
    est = exp(-1j*pi*[0:(5-1)]'*st); % steering matrix
    % amplitude response
    P = 20*log10(abs(W_H*est).^2);
    plot(st,P)
end
```

Uncorrelated target signal and interference

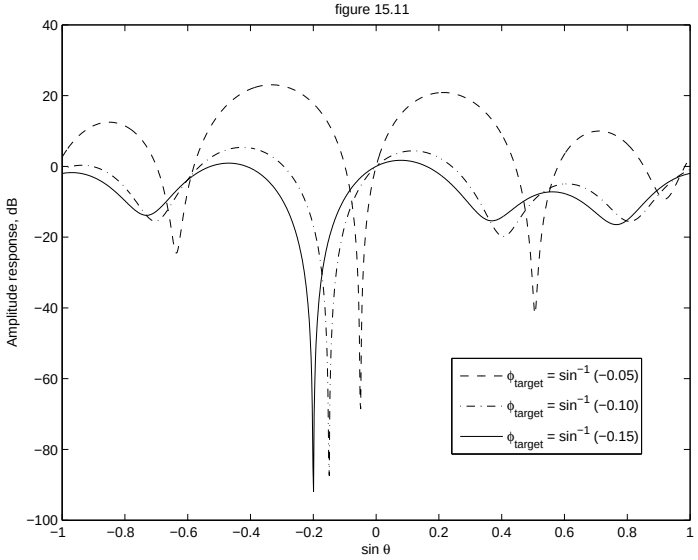


Figure 1

Problem 15.12

Correlated target signal and interference

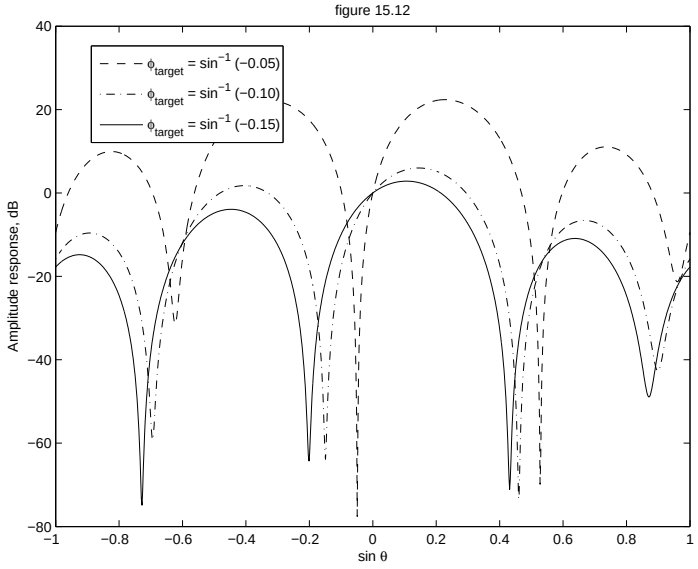


Figure 2

Chapter 16

Problem 16.1

For the transversal filter of Fig. 16.3 in the text we have:

$$\text{tap-weight vector} = \mathbf{k}_M(n)$$

$$\text{tap-input vector} = \mathbf{u}_M(i), \qquad i = 1, 2, \ldots, n$$

$$\text{desired response, } d(i) = \begin{cases} 1 & i = n \\ 0 & i = n-1, \ldots, 1 \end{cases}$$

The a posteriori estimation error equals

$$e(i) = d(i) - \mathbf{k}_M^H(n)\mathbf{u}_M(i), \qquad i = 1, 2, \ldots, n$$

The deterministic cross-correlation vector $\phi(n)$ equals

$$\begin{aligned}\phi(n) =& \sum_{i=1}^{n} \lambda^{n-i}\mathbf{u}_M d^*(i) \\ =& \mathbf{u}_M(n)\end{aligned}$$

We also note that

$$E_d(n) = \sum_{i=1}^{n} \lambda^{n-1} |d(i)|^2 = 1$$

Hence, the sum of the weighted error squares equals

$$\begin{aligned}E_{\text{min}}(n) =& E_d(n) - \phi^H(n)\hat{\mathbf{w}}(n) \\ =& 1 - \mathbf{u}_M^H(n)\mathbf{k}_m(n)\end{aligned}$$

We note that the inner product $\mathbf{u}_M^H(n)\mathbf{k}_m(n)$ is a real-valued scalar. Hence,

$$\begin{aligned} E_{\text{min}}(n) =& 1 - \mathbf{k}_m^H(n)\mathbf{u}_m(n) \\ =& \gamma_m(n) \end{aligned}$$

Problem 16.2

We start with

$$\Phi_m(n) = \lambda\Phi_m(n-1) + \mathbf{u}_m(n)\mathbf{u}_m^H(n)$$

where $\mathbf{u}_m(n)$ is the tap-input vector, $\Phi_m(n-1)$ is the past value of the deterministic correlation matrix, and $\Phi_m(n)$ is its present value. Hence,

$$\begin{aligned} \lambda\Phi_m(n-1) =& \Phi_m(n) - \mathbf{u}_m(n)\mathbf{u}_m^H(n) \\ =& \Phi_m(n)\left[\mathbf{I} - \mathbf{u}_m(n)\mathbf{u}_m^H(n)\Phi_m^{-1}(n)\right] \end{aligned}$$

where $\mathbf{I}$ is the identity matrix. Hence, taking the determinants of both sides:

$$\lambda \det\left[\Phi_m(n-1)\right] = \det\left[\Phi_m(n)\right] \det\left[\mathbf{I} - \mathbf{u}_m(n)\mathbf{u}_m^H(n)\Phi_m^{-1}(n)\right] \tag{1}$$

But,

$$\begin{aligned} \det\left[\mathbf{I} - \mathbf{u}_m(n)\mathbf{u}_m^H(n)\Phi_m^{-1}(n)\right] =& \det\left[\mathbf{I} - \mathbf{u}_m^H(n)\Phi_m^{-1}(n)\mathbf{u}_m(n)\right] \\ =& 1 - \mathbf{u}_m^H(n)\Phi_m^{-1}(n)\mathbf{u}_m(n) \\ =& \gamma_m(n) \end{aligned}$$

We may therefore rewrite Equation (1) as

$$\lambda \det\left[\Phi_m(n-1)\right] = \det\left[\Phi_m(n)\right]\gamma_m(n)$$

Hence, we may express the conversion factor $\gamma_m(n)$ as

$$\gamma_m(n) = \lambda\left(\frac{\det\left[\Phi_m(n-1)\right]}{\det\left[\Phi_m(n)\right]}\right)$$

Problem 16.3

a) The $(m+1)$-by-$(m+1)$ correlation matrix Φ_{m+1} may be expressed in the form

$$\Phi_{m+1} = \begin{bmatrix} U(n) & \phi_1^H(n) \\ \phi_1(n) & \Phi_m(n-1) \end{bmatrix} \tag{1}$$

Define the inverse of the matrix as

$$\Phi_{m+1}^{-1} = \begin{bmatrix} \alpha_1 & \beta_1^H \\ \beta_1 & \Gamma_1 \end{bmatrix} \tag{2}$$

Hence, from Equations (1) and (2):

$$\begin{aligned} \mathbf{I}_{m+1} =& \Phi_{m+1}(n)\Phi_{m+1}^{-1}(n) \\ =& \begin{bmatrix} U(n) & \phi_1^H(n) \\ \phi_1(n) & \Phi_m(n-1) \end{bmatrix} \begin{bmatrix} \alpha_1 & \beta_1^H \\ \beta_1 & \Gamma_1 \end{bmatrix} \\ =& \begin{bmatrix} U(n)\alpha_1 + \phi_1^H(n)\beta_1 & U(n)\beta_1^H + \phi_1^H(n)\Gamma_1 \\ \phi_1(n)\alpha_1 + \Phi_m(n-1)\beta_1 & \phi_1(n)\beta_1^H + \Phi_m(n-1)\Gamma_1 \end{bmatrix} \end{aligned}$$

From this relation we deduce the following four equations:

$$U(n)\alpha_1 + \phi_1^H(n)\beta_1 = 1 \tag{3}$$

$$U(n)\beta_1^H + \phi_1^H(n)\Gamma_1 = \mathbf{0} \tag{4}$$

$$\phi_1(n)\alpha_1 + \Phi_m(n-1)\beta_1 = \mathbf{0} \tag{5}$$

$$\phi_1(n)\beta_1^H + \Phi_m(n-1)\Gamma_1 = \mathbf{I}_m \tag{6}$$

Eliminate β_1 between Equations (3) and (5):

$$U(n)\alpha_1 - \phi_1^H(n)\Phi_m^{-1}(n-1)\phi_1(n)\alpha_1 = 1$$

Hence,

$$\alpha_1 = \frac{1}{U(n) - \phi_1^H(n)\Phi_m^{-1}(n-1)\phi_1(n)} \tag{7}$$

which is real-valued. Correspondingly,

$$\begin{aligned} \beta_1 =& -\Phi_m^{-1}(n-1)\phi_1(n)\alpha_1 \\ =& \frac{-\Phi_m^{-1}(n-1)\phi_1(n)}{U(n) - \phi_1^H(n)\Phi_m^{-1}(n-1)\phi_1(n)} \end{aligned}$$

From Equation (6):

$$\begin{aligned}\Gamma_1 =&\Phi_m^{-1}(n-1) - \Phi_m^{-1}(n-1)\phi_1(n)\beta_1^H \\ =&\Phi_m^{-1}(n-1) + \frac{\Phi_m^{-1}(n-1)\phi_1(n)\phi_1^H(n)\Phi_m^{-1}(n-1)}{U(n) - \phi_1^H(n)\Phi_m^{-1}(n-1)\phi_1(n)}\end{aligned} \tag{9}$$

Check:

Substitute Equations (8) and (9) into the left hand side of Equation (4):

$$\begin{aligned}U(n)\beta_1^H + \phi_1^H(n)\Gamma_1 =&u(n)\left(\frac{-\phi_1^H(n)\Phi_m^{-1}(n-1)}{U(n) - \phi_1^H(n)\Phi_m^{-1}(n-1)\phi_1(n)}\right) \\ &+ \phi_1^H(n)\Phi_m^{-1}(n-1) \\ &+ \frac{\phi_1^H(n)\Phi_m^{-1}(n-1)\phi_1(n)\phi_1^H(n)\Phi_m^{-1}(n-1)}{U(n) - \phi_1^H(n)\Phi_m^{-1}(n-1)\phi_1(n)} \\ =&\mathbf{0}\end{aligned}$$

This agrees with the right-hand side of Equation (4).

From Equation (7) we note that

$$\alpha_1 = \frac{1}{\mathscr{F}_m(n)}$$

From Equation (8) we note that

$$\beta_1 = -\frac{\hat{\mathbf{w}}_f(n)}{\mathscr{F}_m(n)}$$

From Equation (9) we note that

$$\Gamma_1 = \begin{bmatrix} 0 & \mathbf{0}_m^T \\ \mathbf{0}_m & \Phi_m^{-1}(n-1) \end{bmatrix} + \frac{\hat{\mathbf{w}}_f(n)\hat{\mathbf{w}}_f^H(n)}{\mathscr{F}_m(n)}$$

Hence, we may express the inverse matrix of Equation (1) as follows:

$$\begin{aligned}\Phi_{m+1}^{-1}(n) &= \begin{bmatrix} 0 & \mathbf{0}_m^T \\ \mathbf{0}_m & \Phi_m^{-1}(n-1) \end{bmatrix} + \frac{1}{\mathscr{F}_m(n)}\begin{bmatrix} 1 & -\hat{\mathbf{w}}_f^H(n) \\ -\hat{\mathbf{w}}_f(n) & \hat{\mathbf{w}}_f(n)\hat{\mathbf{w}}_f^H(n) \end{bmatrix} \\ &= \begin{bmatrix} 0 & \mathbf{0}_m^T \\ \mathbf{0}_m & \Phi_m^{-1}(n-1) \end{bmatrix} + \frac{1}{\mathscr{F}_m(n)}\begin{bmatrix} 1 \\ -\hat{\mathbf{w}}_f(n) \end{bmatrix}\begin{bmatrix} 1 & -\hat{\mathbf{w}}_f^H(n) \end{bmatrix} \\ &= \begin{bmatrix} 0 & \mathbf{0}_m^T \\ \mathbf{0}_m & \Phi_m^{-1}(n-1) \end{bmatrix} + \frac{1}{\mathscr{F}_m(n)}\mathbf{a}_m(n)\mathbf{a}_m^H(n)\end{aligned}$$

b)

Consider next the second form of the $(m+1)$-by$(m+1)$ correlation matrix $\Phi_{m+1}(n)$ given by

$$\Phi_{m+1}(n) = \begin{bmatrix} \Phi_m(n) & \phi_2(n) \\ \phi_2^H(n) & U(n-m) \end{bmatrix} \tag{10}$$

Define the inverse of this matrix as

$$\Phi_{m+1}^{-1}(n) = \begin{bmatrix} \Gamma_2 & \beta_2 \\ \beta_2^H & \alpha_2 \end{bmatrix} \tag{11}$$

Using Equations (10) and (11):

$$\begin{aligned} \mathbf{I}_{m+1} =& \Phi_{m+1}(n)\Phi_{m+1}^{-1}(n) \\ =& \begin{bmatrix} \Phi_m(n) & \phi_2(n) \\ \phi_2^H(n) & U(n-m) \end{bmatrix} \begin{bmatrix} \Gamma_2 & \beta_2 \\ \beta_2^H & \alpha_2 \end{bmatrix} \\ =& \begin{bmatrix} \Phi_m(n)\Gamma_2 + \phi_2(n)\beta_2^H & \Phi_m(n)\beta_2 + \phi_2(n)\alpha_2 \\ \phi_2^H(n)\Gamma_2 + U(n-m)\beta_2^H & \phi_2^H(n)\beta_2 + U(n-m)\alpha_2 \end{bmatrix} \end{aligned}$$

We thus deduce the following four relations:

$$\Phi_m(n)\Gamma_2 + \phi_2(n)\beta_2^H = \mathbf{I}_m \tag{12}$$

$$\Phi_m(n)\beta_2 + \phi_2(n)\alpha_2 = \mathbf{0} \tag{13}$$

$$\phi_2^H(n)\Gamma_2 + U(n-m)\beta_2^H = \mathbf{0} \tag{14}$$

$$\phi_2^H(n)\beta_2 + U(n-m)\alpha_2 = 1 \tag{15}$$

Eliminate β_2 between Equations (13) and (15):

$$-\phi_2^H(n)\Phi_m^{-1}(n)\phi_2(n)\alpha_2 + U(n-m)\alpha_2 = 1$$

Hence,

$$\alpha_2 = \frac{1}{U(n-m) - \phi_2^H(n)\Phi_m^{-1}(n)\phi_2(n)} \tag{16}$$

Correspondingly, β_2 equals

$$\begin{aligned} \beta_2 =& -\Phi_m^{-1}(n)\phi_2(n)\alpha_2 \\ =& \frac{\Phi_m^{-1}(n)\phi_2(n)}{U(n-m) - \phi_2^H(n)\Phi_m^{-1}(n)\phi_2(n)} \end{aligned} \tag{17}$$

Substitute Equation (17) into Equation (12) and solve for Γ_2:

$$\begin{aligned}\Gamma_2 =&\Phi_m^{-1}(n) - \Phi_m^{-1}(n)\phi_2(n)\beta_2^H \\ =&\Phi_m^{-1}(n) + \frac{\Phi_m^{-1}(n)\phi_2(n)\phi_2^H(n)\Phi_m^{-1}(n)}{U(n-m) - \phi_2^H(n)\Phi_m^{-1}(n)\phi_2(n)}\end{aligned} \tag{18}$$

Check:

Substitute Equations (17) and (18) into the left-hand side of Equation (14)

$$\begin{aligned}&\phi_2^H(n)\Phi_m^{-1}(n) + \frac{\phi_2^H(n)\Phi_m^{-1}(n)\phi_2(n)\Phi_m^{-1}(n)}{U(n-m) - \phi_2^H(n)\Phi_m^{-1}(n)\phi_2(n)} \\ &\qquad -U(n-m)\frac{\phi_2^H(n)\Phi_m^{-1}(n)}{U(n-m) - \phi_2^H(n)\Phi_m^{-1}(n)\phi_2(n)} =\mathbf{0}_m\end{aligned}$$

which agrees with the right-hand side of Equation (14).

Next, we note that

$$\alpha_2 = \frac{1}{\mathscr{B}_m(n)}$$

$$\beta_2 = -\frac{\mathbf{g}(n)}{\mathscr{B}_m(n)}$$

$$\Gamma_2 = \begin{bmatrix}\Phi_m^{-1}(n) & \mathbf{0}_m \\ \mathbf{0}_m^T & 0\end{bmatrix} + \frac{1}{\mathscr{B}_m(n)}\mathbf{w}_b(n)\mathbf{w}_b^H(n)$$

Hence, we may express the inverse matrix $\Phi_{m+1}^{-1}(n)$ in the alternative form:

$$\begin{aligned}\Phi_{m+1}^{-1}(n) &= \begin{bmatrix}\Phi_m^{-1}(n) & \mathbf{0}_m \\ \mathbf{0}_m^T & 0\end{bmatrix} + \frac{1}{\mathscr{B}_m(n)}\begin{bmatrix}\mathbf{w}_b(n)\mathbf{w}_b^H(n) & -\mathbf{w}_b(n) \\ -\mathbf{w}_b(n) & 1\end{bmatrix} \\ &= \begin{bmatrix}\Phi_m^{-1}(n) & \mathbf{0}_m \\ \mathbf{0}_m^T & 0\end{bmatrix} + \frac{1}{\mathscr{B}_m(n)}\begin{bmatrix}-\mathbf{w}_b(n) \\ 1\end{bmatrix}\begin{bmatrix}-\mathbf{w}_b(n) & 1\end{bmatrix} \\ &= \begin{bmatrix}\Phi_m^{-1}(n) & \mathbf{0}_m \\ \mathbf{0}_m^T & 0\end{bmatrix} + \frac{1}{\mathscr{B}_m(n)}\mathbf{c}_m(n)\mathbf{c}_m^H(n)\end{aligned}$$

Problem 16.4

a)

From the solution to part (a) of problem 16.3, we have

$$\Phi_{m+1}^{-1}(n) = \begin{bmatrix} 0 & \mathbf{0}_m^T \\ \mathbf{0}_m & \Phi_m^{-1}(n-1) \end{bmatrix} + \frac{1}{\mathscr{F}_m(n)} \mathbf{a}_m(n)\mathbf{a}_m^H(n) \tag{1}$$

Correspondingly, the input vector $\mathbf{u}_{m+1}(n)$ is partitioned as follows:

$$\mathbf{u}_{m+1}(n) = \begin{bmatrix} u(n) \\ \mathbf{u}_{m+1}(n-1) \end{bmatrix} \tag{2}$$

From the definition of the conversion factor, we have

$$\gamma_{m+1}(n) = 1 - \mathbf{u}_{m+1}^H(n-1)\Phi_{m+1}^{-1}(n)\mathbf{u}_{m+1}(n) \tag{3}$$

Therefore, substituting Equations (1) and (2) into Equation (3):

$$\begin{aligned} \gamma_{m+1}(n) =& 1 - \mathbf{u}_m^H(n-1)\Phi_m^{-1}(n-1)\mathbf{u}_m(n-1) \\ & - \frac{1}{\mathscr{F}_m(n)} \mathbf{u}_{m+1}^H(n)\mathbf{a}_m(n)\mathbf{a}_m^H(n)\mathbf{u}_{m+1}(n) \\ =& \gamma_{m+1}(n-1) - \frac{|f_m(n)|^2}{\mathscr{F}_m(n)} \end{aligned} \tag{4}$$

where we have used the fact that

$$f_m(n) = \mathbf{a}_m^H(n)\mathbf{u}_{m+1}(n)$$

b)

From the solution to part **b)** of problem 16.3, we have

$$\Phi_{m+1}^{-1}(n) = \begin{bmatrix} \Phi_m^{-1}(n) & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} + \frac{1}{\mathscr{B}_m(n)} \mathbf{c}_m(n)\mathbf{c}_m^H(n) \tag{5}$$

This time we partition the input vector $\mathbf{u}_{m+1}(n)$ as follows:

$$\mathbf{u}_{m+1}(n) = \begin{bmatrix} \mathbf{u}_m(n) \\ u(n-m) \end{bmatrix} \tag{6}$$

Therefore, substituting Equations (5) and (6) into Equation (3):

$$\begin{aligned}\gamma_{m+1}(n) =& 1 - \mathbf{u}_m^H(n)\Phi_m^{-1}(n)\mathbf{u}_m(n) \\ & - \frac{1}{\mathscr{B}_m(n)}\mathbf{u}_{m+1}^H(n)\mathbf{c}_m(n)\mathbf{c}_m^H(n)\mathbf{u}_{m+1}(n) \\ =& \gamma_m(n) - \frac{|b_m(n)|^2}{\mathscr{B}_m(n)} \end{aligned} \tag{7}$$

where we have made use of the fact that

$$b_m(n) = \mathbf{c}_m^H(n)\mathbf{u}_{m+1}(n)$$

c)

We invoke the following property of the conversion factor:

$$\gamma_m(n-1) = \frac{f_m(n)}{\eta_m(n)} \tag{8}$$

Also, we note that

$$\mathscr{F}_m(n) = \lambda\mathscr{F}_m(n-1) + \eta_m(n)f_m^*(n) \tag{9}$$

Therefore eliminating $\eta_m(n)$ between Equations (8) and (9):

$$\mathscr{F}_m(n) = \lambda\mathscr{F}_m(n-1) + \frac{|f_m(n)|^2}{\gamma_m(n-1)} \tag{10}$$

Finally, eliminating $|f_m(n)|^2$ between Equations (4) and (10):

$$\begin{aligned}\gamma_{m+1}(n) =& \gamma_m(n-1) - \frac{\gamma_m(n-1)}{\mathscr{F}_m(n)}\left[\mathscr{F}_m(n) - \lambda\mathscr{F}_m(n-1)\right] \\ =& \lambda\frac{\mathscr{F}_m(n-1)}{\mathscr{F}_m(n)}\gamma_m(n-1)\end{aligned}$$

d)

Next, we invoke another property of the conversion factor:

$$\gamma_m(n) = \frac{b_m(n)}{\beta_m(n)} \tag{11}$$

We also note that

$$\mathscr{B}_m(n) = \lambda \mathscr{B}_m(n-1) + b_m^*(n)\beta_m(n) \tag{12}$$

Therefore, eliminating $\beta_m(n)$ between Equations (11) and (12):

$$\mathscr{B}_m(n) = \lambda \mathscr{B}_m(n-1) + \frac{|b_m(n)|^2}{\gamma_m(n)} \tag{13}$$

Eliminating $|b_m(n)|^2$ between Equations (7) and (13):

$$\begin{aligned}\gamma_{m+1}(n) =& \gamma_m(n) - \frac{\gamma_m(n)}{\mathscr{B}_m(n)}\left[\mathscr{B}_m(n) - \lambda \mathscr{B}_m(n-1)\right] \\ =& \lambda \frac{\mathscr{B}_m(n-1)}{\mathscr{B}_m(n)}\gamma(n)\end{aligned}$$

Problem 16.5

a)

$$
\begin{aligned}
\mathscr{F}_m(n) &= \sum_{i=1}^{n} \lambda^{n-i} \left| f_m(i) \right|^2 \\
&= \sum_{i=1}^{n} \lambda^{n-i} f_m(i) f_m^*(i) \\
&= \sum_{i=1}^{n} \lambda^{n-i} \left[u(i) - \hat{\mathbf{w}}_{f,m}^H(n) \mathbf{u}_m(i-1) \right] f_m^*(i) \\
&= \sum_{i=1}^{n} \lambda^{n-i} u(i) f_m^*(i) - \cancel{\left(\hat{\mathbf{w}}_{f,m}^H(n) \sum_{i=1}^{n} \lambda^{n-i} \mathbf{u}_m(i-1) f_m^*(i) = \mathbf{0} \right)} \\
&= \sum_{i=1}^{n} \lambda^{n-i} u(i) f_m^*(i) \\
&= \sum_{i=1}^{n} \lambda^{n-i} \left[\eta_m(i) + \hat{\mathbf{w}}_{f,m}^H(n-1) \mathbf{u}_m(i-1) \right] f_m^*(i) \\
&= \sum_{i=1}^{n} \lambda^{n-i} \eta_m(i) f_m^*(i) + \cancel{\left(\hat{\mathbf{w}}_{f,m}^H(n-1) \sum_{i=1}^{n} \mathbf{u}_m(i-1) f_m^*(i) = \mathbf{0} \right)} \\
&= \sum_{i=1}^{n} \lambda^{n-i} \eta_m(i) f_m^*(i) \\
&= \sum_{i=1}^{n} \lambda^{n-i} \eta_m(i) f_m^*(i) + \eta_m(n) f_m^*(n) \\
&= \lambda \mathscr{F}_m(n-1) + \eta_m(n) f_m^*(n)
\end{aligned}
$$

b)

Following a similar procedure, we may use the relations

$$
\mathscr{B}_m(n) = \sum_{i=1}^{n} \lambda^{n-i} \left| b_m(i) \right|^2
$$

and

$$\sum_{i=1}^{n} \lambda^{n-i} \mathbf{u}_m(i) b_m^*(i) = \mathbf{0}$$

To derive the recursion

$$\mathscr{B}_m(n) = \lambda \mathscr{B}_m(n-1) + \beta_m(n) b_m^*(n)$$

Problem 16.6

Referring to the subsection on page 617 on the three kinds of estimation error for the conversion factor, Section 16.4, we may make the following comparative observations:

a)

Equation (16.27) teaches us the following: The estimation error, $\gamma_m(n)$, involves both a priori estimation error, $\xi_m(n)$, and a posteriori estimation error, $e_m(n)$. These two errors are basic to the RLS algorithm; see (10.26) for the a priori estimation error, and Equation (10.27) for the a posteriori estimation error

b)

Turning next to Equation (16.28), by definition, this second equation applies to forward linear prediction because the a posteriori prediction error $f_m(n)$ and a priori prediction error $\eta_m(n)$, they both relate to forward prediction, hence the justification for Equation (16.28).

c)

Finally, referring to Equation (16.29), by definition, this equation applies to backward linear prediction because the a priori prediction error $b_m(n)$ and the a posteriori prediction error $\beta_m(n)$, both of them are intended for backward prediction.

Problem 16.7

a)

To prove the statement in part **a)** of the Problem, we start with

$$\begin{aligned}
\Delta_{m-1}(n) &= \sum_{i=1}^{n} \lambda^{n-i} b_{m-1}(i-1) f_{m-1}^{*}(i) \\
&= \sum_{i=1}^{n} \lambda^{n-i} \left[u(i-m) - \hat{\mathbf{w}}_{b,m-1}^{H}(n-2)\mathbf{u}_{m-1}(i-1) \right] f_{m-1}^{*}(i) \\
&= \sum_{i=1}^{n} \lambda^{n-i} u(i-m) f_{m-1}^{*}(i) - \cancel{\left(\hat{\mathbf{w}}_{b,m-1}^{H}(n-2) \sum_{i=1}^{n} \mathbf{u}_{m-1}(i-1) f_{m-1}^{*}(i) = \mathbf{0} \right)} \\
&= \sum_{i=1}^{n} \lambda^{n-i} u(i-m) f_{m-1}^{*}(i) \\
&= \sum_{i=1}^{n} \lambda^{n-i} \left[\beta_{m-1}(i-1) + \hat{\mathbf{w}}_{b,m-1}^{H}(n-1)\mathbf{u}_{m-1}(i) \right] f_{m-1}^{*}(i) \\
&= \sum_{i=1}^{n} \lambda^{n-i} \beta_{m-1}(i-1) f_{m-1}^{*}(i) + \cancel{\left(\hat{\mathbf{w}}_{b,m-1}^{H}(n-1) \sum_{i=1}^{n} \mathbf{u}_{m-1}(i) f_{m-1}^{*}(i) = \mathbf{0} \right)} \\
&= \sum_{i=1}^{n} \lambda^{n-i} \beta_{m-1}(i-1) f_{m-1}^{*}(i) \\
&= \sum_{i=1}^{n} \lambda^{n-i} \beta_{m-1}(i-1) f_{m-1}^{*}(i) + \beta_{m-1}(n-1) f_{m-1}^{*}(n) \\
&= \lambda \Delta_{m-1}(n-1) + \beta_{m-1}(n-1) f_{m-1}^{*}(n) \qquad (1)
\end{aligned}$$

Comparing Equation (1) with Equation (16.49) of the text, we deduce the equivalence

$$\eta_{m-1}^{*}(n) b_{m-1}(n-1) = \beta_{m-1}(n-1) f_{m-1}^{*}(n) \qquad (2)$$

b)

Applying Equations (16.28) and (16.29) of the text, we may write

$$\eta_{m-1}^{*}(n) = \frac{f_{m-1}^{*}(n)}{\gamma_{m-1}(n-1)} \qquad (3)$$

$$b_{m-1}(n-1) = \gamma_{m-1}(n-1) \beta_{m-1}(n-1) \qquad (4)$$

Multiplying Equations (3) and (4)

$$\begin{aligned}\eta_{m-1}^{*}(n)b_{m-1}(n-1) =& \frac{f_{m-1}^{*}(n)}{\gamma_{m-1}(n-1)}\gamma_{m-1}(n-1)\beta_{m-1}(n-1)\\ =& f_{m-1}^{*}(n)\beta_{m-1}(n-1)\end{aligned}$$

which proves Equation (2)

Problem 16.8

a)

Let $\Phi_{m+1}(n)$ denote the $(m+1)$-by-$(m+1)$ correlation matrix of the tap-input vector $\mathbf{u}_{m+1}(i)$ applied to the forward prediction error filter of order m, where $1 \leq i \leq n$. Let $\mathbf{a}_m(n)$ denote the tap-weight vector of this filter, and $\mathscr{F}_m(n)$ denote the corresponding sum of weighted prediction-error squares. We may characterize this filter by the augmented normal equations

$$\Phi_{m+1}(n)\mathbf{a}_m(n) = \begin{bmatrix}\mathscr{F}_m(n)\\ \mathbf{0}_m\end{bmatrix} \tag{1}$$

where $\mathbf{0}_m$ is the m-by-1 null vector. The correlation matrix $\Phi_{m+1}(n)$ may be partitioned in two different ways, depending on how we interpret the first or last element of the tap-input vector $\mathbf{u}_{m+1}(i)$. The form of partitioning that we like to use first is the one that enables us to relate the tap-weight vector $\mathbf{a}_m(n)$, pertaining to prediction order m, to the tap-weight vector $\mathbf{a}_{m-1}(n)$, pertaining to prediction order $m-1$. This aim is realized by using

$$\Phi_{m+1}(n) = \begin{bmatrix}\Phi_m(n) & \phi_2(n)\\ \phi_2^H(n) & U_2(n)\end{bmatrix} \tag{2}$$

where $\Phi_m(n)$ is the m-by-m correlation matrix of the tap-input vector $\mathbf{u}_m(i)$, $\phi_2(n)$ is the m-by-1 cross-correlation vector between $\mathbf{u}_m(i)$ and $u(i-m)$, and $U_2(n)$ is zero for $n-m \leq 0$. We postmultiply both sides of Equation (2) by an $(m+1)$-by-1 vector whose first m elements are defined by the vector $\mathbf{a}_{m-1}(n)$ and whose last element equals zero. We may thus write

$$\begin{aligned}\Phi_{m+1}(n)\begin{bmatrix}\mathbf{a}_{m-1}(n)\\ 0\end{bmatrix} &= \begin{bmatrix}\Phi_m(n) & \phi_2(n)\\ \phi_2^H(n) & U_2(n)\end{bmatrix}\begin{bmatrix}\mathbf{a}_{m-1}(n)\\ 0\end{bmatrix}\\ &= \begin{bmatrix}\Phi_m(n)\mathbf{a}_{m-1}(n)\\ \phi_2^H(n)\mathbf{a}_{m-1}(n)\end{bmatrix}\end{aligned} \tag{3}$$

Both $\Phi_m(n)$ and $\mathbf{a}_{m-1}(n)$ have the same time argument n. Furthermore, in the first line of Equation (3), they are both positioned in such a way that when the matrix multiplication is performed $\Phi_m(n)$ becomes postmulitpled by $\mathbf{a}_{m-1}(n)$. For a forward prediction-error filter of order m-1, evaluated at time n, the set of augmented normal equations defined in Equation (1) takes the form

$$\Phi_m(n)\mathbf{a}_{m-1}(n) = \begin{bmatrix} \mathscr{F}_{m-1}(n) \\ \mathbf{0}_{m-1} \end{bmatrix}$$

Define the scalar

$$\Delta_{m-1}(n) = \phi_2^H(n)\mathbf{a}_{m-1}(n) \tag{4}$$

Accordingly, we may rewrite Equation (3) as

$$\Phi_{m+1}(n) \begin{bmatrix} \mathbf{a}_{m-1}(n) \\ 0 \end{bmatrix} = \begin{bmatrix} \mathscr{F}_{m-1}(n) \\ \mathbf{0}_{m-1} \\ \Delta_{m-1}(n) \end{bmatrix} \tag{5}$$

b)

For a definition of $\Delta_{m-1}(n)$, we have

$$\Delta_{m-1}(n) = \sum_{i=1}^{n} \lambda^{n-i} f_{m-1}^*(i) b_{m-1}(i-1) \tag{6}$$

For another definition of this same quantity, we have

$$\Delta_{m-1}(n) = \phi_2^H(n)\mathbf{a}_{m-1}(n) \tag{7}$$

where

$$\phi_2(n) = \sum_{i=1}^{n} \lambda^{n-i} \mathbf{u}_m(i) u^*(i-1) \tag{8}$$

To show that these two definitions are equivalent, we first substitute Equation (8) into Equation (7):

$$\Delta_{m-1}(n) = \sum_{i=1}^{n} \lambda^{n-i} \mathbf{u}_m^H(i)\mathbf{a}_{m-1}(n) u(i-m) \tag{9}$$

From the definition of the forward prediction error, we have

$$f_{m-1}(i) = \mathbf{a}_{m-1}^{H}(n)\mathbf{u}_m(i), \qquad 1 \le i \le n$$

We may therefore rewrite Equation (9) as

$$\Delta_{m-1}(n) = \sum_{i=1}^{n} \lambda^{n-i} f_{m-1}^{*}(i) u(i-m) \tag{10}$$

Next, from the definition of the backward prediction error, we have

$$b_{m-1}(i) = u(i-m) - \sum_{k=1}^{m-1} w_{b,m-1,k}^{*}(n) u(i-k) \tag{11}$$

where $w_{b,m-1,k}(n)$ is the kth element of the backward predictor's coefficient vector. Therefore, eliminating $u(i-m)$ between Equations (10) and (11):

$$\Delta_{m-1}(n) = \sum_{i=1}^{n} \lambda^{n-i} f_{m-1}^{*}(i) b_{m-1}(i) + \sum_{i=1}^{n} \sum_{k=1}^{n} \lambda^{n-i} w_{b,m-1,k}^{*}(n) f_{m-1}^{*}(i) u(i-k) \tag{12}$$

But the tap inputs $u(i-1), u(i-2), \ldots, u(i-m+1)$ are the very inputs involved in computing the forward prediction error. From the principle or orthogonality, we therefore have

$$\sum_{k=1}^{m-1} \lambda^{n-1} f_{m-1}^{*}(i) u(i-k) = 0 \qquad \text{for all } i$$

That is,

$$\Delta_{m-1}(n) = \sum_{i=1}^{n} \lambda^{n-i} f_{m-1}^{*}(i) b_{m-1}(i)$$

which is the desired result.

c)

Consider next the backward prediction-error filter of order m. Let $\mathbf{c}_m(n)$ denote its tap-weight vector, and $\mathscr{B}_m(n)$ denote the corresponding sum of weighted prediction-error

squares. This filter is characterized by the augmented normal equations written in the matrix form:

$$\Phi_{m+1}(n)\mathbf{c}_m(n) = \begin{bmatrix} \mathbf{0}_m \\ \mathscr{B}_m(n) \end{bmatrix} \tag{13}$$

where $\Phi_{m+1}(n)$ is as defined previously, and $\mathbf{0}_m$ is the m-by-1 null vector. This time we use the other partitioned form of the correlation matrix $\Phi_m(n)$, as shown by

$$\Phi_{m+1}(n) = \begin{bmatrix} U_1(n) & \phi_1^H(n) \\ \phi_1(n) & \Phi_m(n-1) \end{bmatrix} \tag{14}$$

where $U_1(n)$ is the sum of weighted squared values of the input $u(i)$ for the time interval $1 \leq i \leq n$, $\phi_1(n)$ is the m-by-1 cross-correlation vector between $u(i)$ and the tap-input vector $\mathbf{u}_m(i-1)$, and $\Phi_m(n-1)$ is the m-by-m correlation matrix of $\mathbf{u}_m(i-1)$. Correspondingly, we postmultiply $\Phi_{m+1}(n)$ by an $(m+1)$-by-1 vector whose first element is zero and whose m remaining elements are defined by the tap-weight vector $\mathbf{c}_{m-1}(n-1)$ that pertains to a backward prediction-error filter of order m-1. We may thus write

$$\begin{aligned} \Phi_{m+1}(n) \begin{bmatrix} 0 \\ \mathbf{c}_{m-1}(n-1) \end{bmatrix} &= \begin{bmatrix} U_1(n) & \phi_1^H(n) \\ \phi_1(n) & \Phi_m(n-1) \end{bmatrix} \begin{bmatrix} 0 \\ \mathbf{c}_{m-1}(n-1) \end{bmatrix} \\ &= \begin{bmatrix} \phi_1^H(n)\mathbf{c}_{m-1}(n-1) \\ \Phi_m(n-1)\mathbf{c}_{m-1}(n-1) \end{bmatrix} \end{aligned} \tag{15}$$

Both $\Phi_m(n-1)$ and $\mathbf{c}_{m-1}(n-1)$ have the same time argument, $n-1$. Also, they are both position in the first line of Equation (15) in such a way that , when the matrix multiplication is performed, $\Phi_{m-1}(n-1)$ becomes postmultiplied by $\mathbf{c}_{m-1}(n-1)$. For a backward prediction-error filter of order $m-1$, evaluated at time $n-1$, the set of augmented normal equations in Equation (13) takes the form

$$\Phi_m(n-1)\mathbf{c}_{m-1}(n-1) = \begin{bmatrix} \mathbf{0}_{m-1} \\ \mathscr{B}_{m-1}(n-1) \end{bmatrix}$$

Define the second scalar

$$\Delta'_{m-1}(n-1) = \phi_1^H(n)\mathbf{c}_{m-1}(n-1) \tag{16}$$

where the prime is intended to distinguish this new parameter from $\Delta_{m-1}(n-1)$. Accordingly, we may rewrite Equation (15) as

$$\Phi_{m+1}(n) \begin{bmatrix} 0 \\ \mathbf{c}_{m-1}(n-1) \end{bmatrix} = \begin{bmatrix} \Delta'_{m-1}(n) \\ \mathbf{0}_{m-1} \\ \mathscr{B}_{m-1}(n-1) \end{bmatrix} \tag{17}$$

d)

The parameters $\Delta_{m-1}(n-1)$ and $\Delta'_{m-1}(n-1)$, defined by Equations (4) and (16), respectively, are in actual fact the complex conjugate of one another; that is,

$$\Delta'_{m-1}(n) = \Delta^*_{m-1}(n) \tag{18}$$

where $\Delta^*_{m-1}(n)$ is the complex conjugate of $\Delta_{m-1}(n)$. We prove this relation in three stages:

1. We premultiply both sides of Equation (5) by the row vector

   $$\begin{bmatrix} 0 & \mathbf{c}^H_{m-1}(n-1) \end{bmatrix}$$

   where the superscript H denotes the Hermitian transposition. The result of this matrix multiplication is the scalar

   $$\begin{aligned} \begin{bmatrix} 0 & \mathbf{c}^H_{m-1}(n-1) \end{bmatrix} \Phi_{m+1}(n) \begin{bmatrix} \mathbf{a}_{m-1}(n) \\ 0 \end{bmatrix} &= \begin{bmatrix} 0 & \mathbf{c}^H_{m-1}(n-1) \end{bmatrix} \begin{bmatrix} \mathscr{F}_{m-1}(n-1) \\ \mathbf{0}_{m-1} \\ \Delta_{m-1}(n) \end{bmatrix} \\ &= \Delta_{m-1}(n) \end{aligned} \tag{19}$$

   where we have used the fact that the last element of $\mathbf{c}_{m-1}(n-1)$ equals unity.

2. We apply Hermitian transposition to both sides of Equation (17), and use the Hermitian property of the correlation matrix $\Phi_{m+1}(n)$, thereby obtaining

   $$\begin{bmatrix} 0 & \mathbf{c}^H_{m-1}(n-1) \end{bmatrix} \Phi_{m+1}(n) = \begin{bmatrix} \Delta'^*_{m-1}(n) & \mathbf{0}^T_{m-1} & \mathscr{B}_{m-1}(n-1) \end{bmatrix}$$

   where $\Delta'^*_{m-1}(n)$ is the complex conjugate of $\Delta'_{m-1}(n)$ and $\mathscr{B}_{m-1}(n-1)$ is real valued. Next we use this relation to evaluate the scalar

   $$\begin{aligned} &\begin{bmatrix} 0 & \mathbf{c}^H_{m-1}(n-1) \end{bmatrix} \Phi_{m+1}(n) \begin{bmatrix} \mathbf{a}_{m-1}(n) \\ 0 \end{bmatrix} \\ &\qquad = \begin{bmatrix} \Delta'^*_{m-1}(n) & \mathbf{0}^T_{m-1} & \mathscr{B}_{m-1}(n-1) \end{bmatrix} \begin{bmatrix} \mathbf{a}_{m-1}(n) \\ 0 \end{bmatrix} \\ &\qquad = \Delta'^*_{m-1}(n) \end{aligned} \tag{20}$$

   where we have used the fact that the first element fo $\mathbf{a}_{m-1}(n)$ equals unity.

3. Comparison of Equations (19) and (20) immediately yields the relation of Equation (18) between the parameters $\Delta_{m-1}(n)$ and $\Delta'_{m-1}(n)$.

e)

We are now equipped with the relations needed to derive the desired time-update for recursive computation of the parameter $\Delta_{m-1}(n)$.

Consider the m-by-1 tap-weight vector $\mathbf{a}_{m-1}(n-1)$ that pertains to a forward prediction-error filter of order m-1, evaluated at time n-1. The reason for considering time n-1 will become apparent presently. Since the leading element of the vector $\mathbf{a}_{m-1}(n-1)$ equals unity, we may express $\Delta_{m-1}(n)$ as follows [see Equations (18) and (20)]

$$\Delta_{m-1}(n) = \begin{bmatrix}\Delta_{m-1}(n) & \mathbf{0}_{m-1}^T & \mathscr{B}_{m-1}(n-1)\end{bmatrix} \begin{bmatrix}\mathbf{a}_{m-1}(n) \\ 0\end{bmatrix} \tag{21}$$

Taking the Hermitian transpose of both sides of Equation (17), and recognizing the Hermitian property of $\Phi_{m+1}(n)$ and using the relation of Equation (19), we get

$$\begin{bmatrix}0 & \mathbf{c}_{m-1}(n-1)\end{bmatrix} \Phi_{m+1}(n) = \begin{bmatrix}\Delta_{m-1}(n) & \mathbf{0}_{m-1}^T & \mathscr{B}_{m-1}(n-1)\end{bmatrix} \tag{22}$$

Hence, substitution of Equations (22) into (21) yields

$$\Delta_{m-1}(n) = \begin{bmatrix}0 & \mathbf{c}_{m-1}^H(n-1)\end{bmatrix} \Phi_{m+1}(n) \begin{bmatrix}\mathbf{a}_{m-1}(n) \\ 0\end{bmatrix} \tag{23}$$

But the correlation matrix $\Phi_{m+1}(n)$ may be time-updated as follows:

$$\Phi_{m+1}(n) = \lambda\Phi_{m+1}(n-1) + \mathbf{u}_{m+1}(n)\mathbf{u}_{m+1}^H(n) \tag{24}$$

Accordingly, we may use this relation fro $\Phi_{m+1}(n)$ to rewrite Equation (23) as

$$\begin{aligned}\Delta_{m-1}(n) =& \lambda \begin{bmatrix}0 & \mathbf{c}_{m-1}^H(n-1)\end{bmatrix} \Phi_{m+1}(n-1) \begin{bmatrix}\mathbf{a}_{m-1}(n) \\ 0\end{bmatrix} \\ &+ \begin{bmatrix}0 & \mathbf{c}_{m-1}^H(n-1)\end{bmatrix} \mathbf{u}_{m+1}(n)\mathbf{u}_{m+1}^H(n) \begin{bmatrix}\mathbf{a}_{m-1}(n) \\ 0\end{bmatrix}\end{aligned} \tag{25}$$

f)

Next, we recognize from the definition of forward a priori prediction error that

$$\begin{aligned}\mathbf{u}_{m+1}^H(n) \begin{bmatrix}\mathbf{a}_{m-1}(n-1) \\ 0\end{bmatrix} &= \begin{bmatrix}\mathbf{u}_m^H(n) & u^*(n-1)\end{bmatrix} \begin{bmatrix}\mathbf{a}_{m-1}(n-1) \\ 0\end{bmatrix} \\ &= \mathbf{u}_m^H(n)\mathbf{a}_{m-1}(n-1) \\ &= \eta_{m-1}^*(n)\end{aligned} \tag{26}$$

and from the definition of the backward a posteriori prediction error that

$$\begin{aligned}\begin{bmatrix}0 & \mathbf{c}_{m-1}^H(n-1)\end{bmatrix}\mathbf{u}_{m+1}(n) &= \begin{bmatrix}0 & \mathbf{c}_{m-1}^H(n-1)\end{bmatrix}\begin{bmatrix}u(n)\\ \mathbf{u}_m(n-1)\end{bmatrix}\\ &= \mathbf{c}_{m-1}^H(n-1)\mathbf{u}_m(n-1)\\ &= b_{m-1}(n-1)\end{aligned} \tag{27}$$

Also, by substituting $n-1$ for n into Equation (5), we have

$$\Phi_{m+1}(n-1)\begin{bmatrix}\mathbf{a}_{m-1}(n-1)\\ 0\end{bmatrix} = \begin{bmatrix}\mathscr{F}_{m-1}(n-1)\\ \mathbf{0}_{m-1}\\ \Delta_{m-1}(n-1)\end{bmatrix}$$

Hence, using this relation and the fact that the last element of the tap-weight vector $\mathbf{c}_{m-1}(n-1)$, pertaining to the backward prediction-error filter, equals unity, we may write the first term on the right-hand side of Equation (25), expect for λ, as

$$\begin{aligned}\begin{bmatrix}0 & \mathbf{c}_{m-1}(n-1)\end{bmatrix}\Phi_{m+1}(n-1)\begin{bmatrix}\mathbf{a}_{m-1}(n)\\ 0\end{bmatrix} &= \begin{bmatrix}0 & \mathbf{c}_{m-1}^H(n-1)\end{bmatrix}\begin{bmatrix}\mathscr{F}_{m-1}(n-1)\\ \mathbf{0}_{m-1}\\ \Delta_{m-1}(n-1)\end{bmatrix}\\ &= \Delta_{m-1}(n-1)\end{aligned} \tag{28}$$

Finally, substituting Equations (26), (27), and (28) into Equation (25), we may express the time-update recursion for $\Delta_{m-1}(n)$ as

$$\Delta_{m-1}(n) = \lambda\Delta_{m-1}(n-1) + b_{m-1}(n-1)\eta_{m-1}^*(n) \tag{29}$$

which is the desired result.

Problem 16.9

Foir convenience of presentation the two parts of the solution are labeled **a)** and **b)**

a)

We start with the relations

$$\Phi_{m+1}(n)\begin{bmatrix}\mathbf{a}_{m-1}(n)\\ 0\end{bmatrix} = \begin{bmatrix}\mathscr{F}_{m-1}(n)\\ \mathbf{0}_{m-1}\\ \Delta_{m-1}(n)\end{bmatrix} \tag{1}$$

and

$$\Phi_{m+1}(n)\begin{bmatrix}0\\ \mathbf{c}_{m-1}(n-1)\end{bmatrix}=\begin{bmatrix}\Delta_{m-1}^{*}(n)\\ \mathbf{0}_{m-1}\\ \mathscr{B}_{m-1}(n-1)\end{bmatrix} \tag{2}$$

Multiplying Equation (2) by the ratio $\Delta_{m-1}(n)/\mathscr{B}_{m-1}(n-1)$ and subtracting the result from Equation (1):

$$\Phi_{m+1}(n)\left[\begin{bmatrix}\mathbf{a}_{m-1}(n)\\ 0\end{bmatrix}-\frac{\Delta_{m-1}(n)}{\mathscr{B}_{m-1}(n-1)}\begin{bmatrix}0\\ \mathbf{c}_{m-1}(n-1)\end{bmatrix}\right]=\begin{bmatrix}\mathscr{F}_{m-1}(n)-\dfrac{|\Delta_{m-1}(n)|^2}{\mathscr{B}_{m-1}(n-1)}\\ \mathbf{0}_m\end{bmatrix} \tag{3}$$

Equation (3) represents the augmented normal equations for forward prediction with order m, as shown by

$$\Phi_{m+1}(n)\mathbf{a}_{m-1}(n)=\begin{bmatrix}\mathscr{F}_m(n)\\ \mathbf{0}_m\end{bmatrix} \tag{4}$$

on the basis of which we may immediately write

$$\mathscr{F}_m(n)=\mathscr{F}_{m-1}(n)-\frac{|\Delta_{m-1}(n)|^2}{\mathscr{B}_{m-1}(n-1)}$$

Which is the desired equation.

b)

Multiplying Equation (1) by $\Delta_{m-1}^{*}(n)/F_{m-1}(n)$ and subtracting the result from Equation (2):

$$\Phi_{m+1}\left[\begin{bmatrix}0\\ \mathbf{c}_{m-1}(n-1)\end{bmatrix}-\frac{\Delta_{m-1}^{*}(n)}{\mathscr{F}_{m-1}(n)}\begin{bmatrix}\mathbf{a}_{m-1}(n)\\ 0\end{bmatrix}\right]=\begin{bmatrix}\mathbf{0}_m\\ \mathscr{B}_{m-1}(n-1)-\dfrac{|\Delta_{m-1}(n)|^2}{\mathscr{F}_{m-1}(n)}\end{bmatrix} \tag{5}$$

Equation (5) represents the augmented normal equation for backward prediction with order m, as shown by

$$\Phi_{m+1}(n)\mathbf{c}_{m-1}(n)=\begin{bmatrix}\mathbf{0}_m\\ \mathscr{B}_{m-1}(n)\end{bmatrix} \tag{6}$$

On the basis of which we may immediately write

$$\mathscr{B}_m(n)=\mathscr{B}_{m-1}(n-1)-\frac{|\Delta_{m-1}(n)|^2}{\mathscr{F}_{m-1}(n)}$$

Which is the desired result.

Problem 16.10

a)

From part **a)** of Problem 16.3 we have

$$\Phi_{M+1}^{-1}(n) = \begin{bmatrix} 0 & \mathbf{0}_M^T \\ \mathbf{0}_M & \Phi_M^{-1}(n-1) \end{bmatrix} + \frac{1}{\mathscr{F}_M(n)} \mathbf{a}_M(n)\mathbf{a}_M^H(n).$$

Similarly, from part **b)** of the same problem, we have

$$\Phi_{M+1}^{-1}(n) = \begin{bmatrix} \Phi_M^{-1}(n) & \mathbf{0}_M \\ \mathbf{0}_M^T & 0 \end{bmatrix} + \frac{1}{\mathscr{B}_M(n)} \mathbf{c}_M(n)\mathbf{c}_M^H(n).$$

Subtracting these two equations gives $\Phi_{M+1}^{-1}(n) - \Phi_{M+1}^{-1}(n) = \mathbf{0}$, or

$$\begin{bmatrix} 0 & \mathbf{0}_M^T \\ \mathbf{0}_M & \Phi_M^{-1}(n-1) \end{bmatrix} + \frac{1}{\mathscr{F}_M(n)} \mathbf{a}_M(n)\mathbf{a}_M^H(n) - \begin{bmatrix} \Phi_M^{-1}(n) & \mathbf{0}_M \\ \mathbf{0}_M^T & 0 \end{bmatrix} - \frac{1}{\mathscr{B}_M(n)} \mathbf{c}_M(n)\mathbf{c}_M^H(n) = \mathbf{0}$$

This is easily rearranged as

$$\begin{bmatrix} \Phi_M^{-1}(n) & \mathbf{0}_M \\ \mathbf{0}_M^T & 0 \end{bmatrix} - \begin{bmatrix} 0 & \mathbf{0}_M^T \\ \mathbf{0}_M & \Phi_M^{-1}(n-1) \end{bmatrix} = \frac{1}{\mathscr{F}_M(n)} \mathbf{a}_M(n)\mathbf{a}_M^H(n) - \frac{1}{\mathscr{B}_M(n)} \mathbf{c}_M(n)\mathbf{c}_M^H(n) \tag{1}$$

which is the desired results for part **a)** of the problem.

b)

From page 436, Equation (10.16) we have the basic matrix recursion

$$\Phi_M^{-1}(n) = \lambda^{-1}\Phi_M^{-1}(n-1) - \lambda^{-1} \left(\frac{\lambda^{-1}\Phi_M^{-1}(n-1)\mathbf{u}(n)\mathbf{u}^H(n)\Phi_M^{-1}(n-1)}{1 + \lambda^{-1}\mathbf{u}^H(n)\Phi_M^{-1}(n-1)\mathbf{u}(n)} \right)$$

From Equation (10.18) of the textbook, we may introduce the gain vector

$$\mathbf{k}_M(n) = \frac{\lambda^{-1}\Phi_M^{-1}(n-1)\mathbf{u}(n)}{1 + \lambda^{-1}\mathbf{u}^H(n)\Phi_M^{-1}(n-1)\mathbf{u}(n)}$$

which yields the expression

$$\Phi_M^{-1}(n) = \lambda^{-1}\Phi_M^{-1}(n-1) - \lambda^{-1}\mathbf{k}_M(n)\mathbf{u}^H(n)\Phi_M^{-1}(n-1)\mathbf{u}(n)$$

Now, again from Equation (10.18) of the textbook, we have

$$\lambda^{-1}\mathbf{u}^H(n)\Phi_M^{-1}(n-1) = \mathbf{k}_M^H(n)\left(1+\lambda^{-1}\mathbf{u}^H(n)\Phi_M^{-1}(n-1)\mathbf{u}(n)\right)$$

But from Equation (14.100) of the textbook, we may introduce the conversion factor as

$$\gamma_M(n) = \frac{1}{1+\lambda^{-1}\mathbf{u}^H(n)\Phi_M^{-1}(n-1)\mathbf{u}(n)}$$

Hence, the basic matrix inversion lemma becomes

$$\Phi_M^{-1}(n) = \lambda^{-1}\Phi_M^{-1}(n-1) - \frac{\mathbf{k}_M(n)\mathbf{k}_M^H(n)}{\gamma_M(n)} \tag{2}$$

which verifies the desired formula.

c)

The result of part **b)** can be rearranged as

$$\Phi_M^{-1}(n-1) = \lambda\Phi_M^{-1}(n) + \lambda\frac{\mathbf{k}_M(n)\mathbf{k}_M^H(n)}{\gamma_M(n)}$$

Inserting this recursion into Equation (1), we have

$$\begin{bmatrix}\Phi_M^{-1}(n) & \mathbf{0}_M\\ \mathbf{0}_M^T & 0\end{bmatrix} - \begin{bmatrix}0 & \mathbf{0}_M^T\\ \mathbf{0}_M & \lambda\Phi_M^{-1}(n)+\lambda\dfrac{\mathbf{k}_M(n)\mathbf{k}_M^H(n)}{\gamma_M(n)}\end{bmatrix} = \frac{1}{\mathscr{F}_M(n)}\mathbf{a}_M(n)\mathbf{a}_M^H(n) - \frac{1}{\mathscr{B}_M(n)}\mathbf{c}_M\mathbf{c}_M^H(n)$$

This, in turn, can be rearranged as

$$\begin{bmatrix}\Phi_M^{-1}(n) & \mathbf{0}_M\\ \mathbf{0}_M^T & 0\end{bmatrix} - \lambda\begin{bmatrix}0 & \mathbf{0}_M^T\\ \mathbf{0}_M & \Phi_M^{-1}(n)\end{bmatrix} = \frac{\mathbf{a}_M(n)\mathbf{a}_M^H(n)}{\mathscr{F}_M(n)} + \lambda\begin{bmatrix}0\\ \mathbf{k}_M(n)\end{bmatrix}\frac{\begin{bmatrix}0 & \mathbf{k}_M(n)\end{bmatrix}}{\gamma_M(n)} - \frac{\mathbf{c}_M(n)\mathbf{c}_M^H(n)}{\mathscr{B}_M(n)} \tag{3}$$

which verifies the desired results of part **c)**

d)

Now we multiply Equation (3) from the left by

$$\begin{bmatrix}1 & \left(z/\sqrt{\lambda}\right) & \left(z/\sqrt{\lambda}\right)^2 & \dots & \left(z/\sqrt{\lambda}\right)^M\end{bmatrix}$$

and from the right by

$$\left[1 \quad \left(z/\sqrt{\lambda}\right) \quad \left(z/\sqrt{\lambda}\right)^2 \quad \ldots \quad \left(z/\sqrt{\lambda}\right)^M\right]^H$$

First we note that, if we set

$$P(z,w^*) = [1(z/\sqrt{\lambda})\ldots(z/\sqrt{\lambda})^M] \begin{bmatrix} \Phi_M^{-1}(n) & \mathbf{0}_M \\ \mathbf{0}_M^T & 0 \end{bmatrix} \begin{bmatrix} 1 \\ w^*/\sqrt{\lambda} \\ \vdots \\ \left(w^*/\sqrt{\lambda}\right)^M \end{bmatrix}$$

Then, upon displacing this matrix one position along the main diagonal, we obtain

$$[1(z/\sqrt{\lambda})\ldots(z/\sqrt{\lambda})^M] \begin{bmatrix} 0 & \mathbf{0}_M \\ \mathbf{0}_M^T & \Phi_M^{-1}(n) \end{bmatrix} \begin{bmatrix} 1 \\ w^*/\sqrt{\lambda} \\ \vdots \\ \left(w^*/\sqrt{\lambda}\right)^M \end{bmatrix} = zw^*P(z,w^*)$$

Thus the left-hand side of part **c)** yields the two-variable polynomial

$$(1 - zw^*)\, P(z,w^*)$$

Similarly, rewrite the right-hand side as the sum and difference of dyads[†]:

$$\frac{\mathbf{a}_M(n)}{\sqrt{F_M(n)}} \times \frac{\mathbf{a}_M^H(n)}{\sqrt{F_M(n)}} + \sqrt{\frac{\lambda}{\gamma_M(n)}} \begin{bmatrix} 0 \\ \mathbf{k}_M(n) \end{bmatrix} \left(\sqrt{\frac{\lambda}{\gamma_M(n)}} \begin{bmatrix} 0 & \mathbf{k}_M^H(n) \end{bmatrix}\right) - \frac{\mathbf{c}_M(n)}{\sqrt{B_M(n)}} \times \frac{\mathbf{c}_M^H(n)}{\sqrt{B_M(n)}}$$

Accordingly, if we set

$$A(z) = [1(z/\sqrt{\lambda})\ldots(z/\sqrt{\lambda})^M] \frac{\mathbf{a}_M(n)}{\sqrt{\mathscr{F}_M(n)}}$$

$$K(z) = [1(z/\sqrt{\lambda})\ldots(z/\sqrt{\lambda})^M] \sqrt{\frac{\lambda}{\gamma_M(n)}} \begin{bmatrix} 0 \\ \mathbf{k}_M(n) \end{bmatrix}$$

$$C(z) = [1(z/\sqrt{\lambda})\ldots(z/\sqrt{\lambda})^M] \frac{\mathbf{c}_M(n)}{\sqrt{\mathscr{B}_M(n)}}$$

[†]A dyad is defined as two vectors with no components connecting them.

then we will have

$$[1(z/\sqrt{\lambda})\dots(z/\sqrt{\lambda})^M]\frac{\mathbf{a}_M(n)}{\sqrt{F_M(n)}}\frac{\mathbf{a}_M(n)}{\sqrt{F_M(n)}}\begin{bmatrix}1\\ w^*/\sqrt{\lambda}\\ \vdots\\ \left(w^*/\sqrt{\lambda}\right)^M\end{bmatrix} = A(z)A^*(\omega)$$

and likewise for the remaining terms. Putting all this together, we obtain the desired result:

$$(1-zw^*)\,P(z,w^*) = A(z)A^*(\omega) + K(z)K^*(\omega) - B(z)B^*(\omega) \tag{4}$$

e)

if we set $z = w = e^{\mathrm{j}\,\omega}$, in Equation (4) then

$$\underbrace{\left(1-e^{\mathrm{j}\,\omega}e^{(-\mathrm{j})\omega}\right)}_{=0}P\left(e^{\mathrm{j}\,\omega},e^{-\mathrm{j}\,\omega}\right) = A(e^{\mathrm{j}\,\omega})A^*(e^{\mathrm{j}\,\omega}) + K(e^{\mathrm{j}\,\omega})K^*(e^{\mathrm{j}\,\omega}) - C(e^{\mathrm{j}\,\omega})C^*(e^{\mathrm{j}\,\omega})$$

which defines the desired result as follows:

$$\left|A(e^{\mathrm{j}\,\omega})\right|^2 + \left|K(e^{\mathrm{j}\,\omega})\right|^2 = \left|C(e^{\mathrm{j}\,\omega})\right|^2, \quad \text{for all } \omega. \tag{5}$$

f)

If we set $w^* = z^*$ in Equation (4) of part **d)**, we have

$$\begin{aligned}\left(1-|z|^2\right)P(z,z^*) =& A(z)A^*(z) + K(z)K^*(z) - C(z)C^*(z)\\ =& |A(z)|^2 + |K(z)|^2 - |C(z)|^2\end{aligned} \tag{6}$$

Now, since $\Phi_M^{-1}(n)$ is a positive definite matrix, it is true that

$$P(z,z^*) = [1(z/\sqrt{\lambda})\dots(z/\sqrt{\lambda})^M]\Phi_M^{-1}(n)\begin{bmatrix}1\\ z^*/\sqrt{\lambda}\\ \vdots\\ \left(z^*/\sqrt{\lambda}\right)^M\end{bmatrix} \geq 0, \quad \text{for all } z$$

Accordingly, the right-hand side of Equation (6) above must take the same sign as $\left(1-|z|^2\right)$, which gives

$$|A(z)|^2 + |K(z)|^2 - |C(z)|^2 \begin{cases} <0, & |z|>1\\ =0, & |z|=1\\ >0, & |z|<1\end{cases} \tag{7}$$

g)
From the first inequality of Equation (7), we have

$$|A(z)|^2 + |K(z)|^2 - |C(z)|^2 < 0, \quad \text{if } |z| > 1 \tag{8}$$

Now, if $C(z)$ were to have a zero with modulus greater than one, that is,

$$C(z_0) = 0, \quad \text{with } |z_0| > 1$$

then, we would have

$$|A(z_0)|^2 + |K(z_0)|^2 - \underbrace{|C(z_0)|^2}_{0} = |A(z_0)|^2 + |K(z_0)|^2 \geq 0$$

With $|z_0| \geq 1$, we obtain a contradiction to Equation (7). Accordingly, $C(z)$ can have no zeros in the inequality $|z| > 1$, so that it may be determined without phase ambiguity from $A(z)$ and $K(z)$ by way of spectral factorization using the result from Equation (5). As stated in part **g)** of the Problem, it follows that once the forward prediction and gain quantities are known, the backward prediction variables contribute nothing further to the solution so are theoretically redundant.

Problem 16.11

a) Backward prediction

The first three lines of Table 16.3 follow from the state-space models of Equations (16.67) through (16.75), together with Table 14.3 of Chapter 14 in the textbook. For the remaining three entries, we may proceed as follows, bearing in mind that we are dealing with scalar quantities for the problem at hand:

(i)

$$\begin{aligned} k(n-1) \leftrightarrow \lambda^{-1}\Phi^{-1}(n-1) =& \lambda^{-1}\left(\sum_{i=1}^{n-1} \lambda^{n-i+1} |\varepsilon_{f,m-1}(i-1)|^2\right)^{-1} \\ =& \lambda^{-1}\mathscr{F}_{m-1}^{-1}(n-1) \end{aligned}$$

(ii)

$$\begin{aligned} g(n) \leftrightarrow \lambda^{-1/2}k(n) =& \lambda^{-1/2}\mathscr{F}_{m-1}^{-1}(n)\varepsilon_{f,m-1}(n) \\ =& \lambda^{-n/2}\gamma_{m-1}^{1/2}(n-1)\beta_{m-1}^{*}(n-1) + \eta_{m-1}^{*}(n)\kappa_{b,m}(n-1) \\ =& \lambda^{-n/2}\gamma_{m-1}^{1/2}(n-1)\beta_{m-1}^{*}(n) \end{aligned}$$

$$\begin{aligned}\beta(n) \leftrightarrow &\lambda^{-n/2}\left(\varepsilon^*_{b,m-1}(n-1) + \varepsilon^*_{f,m-1}(n-1)\kappa_{b,m}(n)\right)\\ =&\lambda^{-n/2}\gamma^{1/2}_{m-1}(n-1)\left(b^*_{m-1}(n-1) + f^*_{m-1}(n)\kappa_{b,m}(n)\right)\\ =&\lambda^{-n/2}\gamma^{1/2}_{m-1}(n-1)b^*_m(n)\end{aligned}$$

(iii) For the remaining entry in Table 16.3:

$$\begin{aligned}r(n) = \frac{\alpha(n)}{\beta(n)} =&\gamma_{m-1}(n-1)\left(\frac{\mathscr{B}_m(n)}{b_m(n)}\right), \qquad \gamma_m^{-1}(n) = \frac{\beta_m(n)}{b_m(n)}\\ =&\frac{\gamma_{m-1}(n-1)}{\gamma_m(n)}\end{aligned}$$

which is the desired result for backward prediction as stated in Table 16.3

b) The relations for joint-process estimation follow a similar procedure.

Problem 16.12

a) Joint-process estimate

The correction in the update equation for $E_{\min}(n)$ is defined by the product term $\alpha_m(n)e^*_m(n)$. This correction term is also equal to $|\varepsilon_{m+1}(n)|^2$. Hence

$$\varepsilon_{m+1}(n)\varepsilon^*_{m+1}(n) = \xi_m(n)e^*_m(n)$$

By definition

$$|\varepsilon_m(n)| = \sqrt{|e_m(n)| \cdot |\xi_m(n)|}$$

We know $\xi_m(n)e^*_m(n)$ is real. Hence, we must have

$$\arg\left[\varepsilon_m(n)\right] = \arg\left[e_m(n)\right]$$

Moreover, it is natural to have

$$\arg\left[e_m(n)\right] = \arg\left[\xi_m(n)\right]$$

which goes to show that

$$\arg\left[\varepsilon_m(n)\right] = \arg\left[e_m(n)\right] + \arg\left[\xi_m(n)\right]$$

b) Backward prediction

The correction term in the update equation for $B_{\min}(n)$ is defined by the product term $\beta_m(n)b_m^*(n)$. This correction term is also equal to $|\beta_{b,m}(n)|^2$. Hence,

$$\varepsilon_{b,m}(n)\varepsilon_{b,m}^*(n) = \beta_m(n)b_m^*(n)$$

By definition,

$$|\beta_{b,m}(n)| = \sqrt{|b_m(n)| \cdot |\Psi_m(n)|}$$

We know $\beta_m(n)b_m^*(n)$ is real. Hence, we must have

$$\arg\left[b_m(n)\right] = \arg\left[\beta_m(n)\right]$$

Moreover, it is natural to have

$$\arg\left[\beta_{b,m}(n)\right] = \arg\left[b_m(n)\right]$$

therefore,

$$\arg\left[\varepsilon_{b,m}(n)\right] = \arg\left[b_m(n)\right] + \arg\left[\beta_m(n)\right]$$

c) Forward prediction

The correction term in the update equation for $\mathscr{F}_m(n)$ is $\eta_m(n)f_m^*(n)$ which is also equal to $|\varepsilon_{f,m}(n)|^2$. Hence,

$$\varepsilon_{f,m}(n)\varepsilon_{f,m}^*(n) = \eta_m(n)f_m^*(n)$$

By definition,

$$|\varepsilon_{f,m}(n)| = \sqrt{|f_m(n)| \cdot |\eta_m(n)|}$$

We know $\eta_m(n)f_m^*(n)$ is real. Hence, we must have

$$\arg\left[f_m(n)\right] = \arg\left[\eta_m(n)\right]$$

Moreover, it is natural to have

$$\arg\left[\varepsilon_{f,m}(n)\right] = \arg\left[f_m(n)\right]$$

Therefore,

$$\arg\left[\varepsilon_{f,m}(n)\right] = \arg\left[f_m(n)\right] + \arg\left[\eta_m(n)\right]$$

Problem 16.13

The matrix product $\Phi_{m+1}(n)\mathbf{L}_m^H(n)$ equals

$$\Phi_{m+1}(n)\mathbf{L}_m^H(n) = \Phi_{m+1}(n)\begin{bmatrix} 1 & c_{1,1}(n) & \dots & c_{m-1,m-1}(n) & c_{m,m}(n) \\ 0 & 1 & \dots & c_{m-1,m-2}(n) & c_{m,m-1}(n) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & c_{m,1}(n) \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix} \tag{1}$$

(1) from the augmented system of normal equations, we have

$$\Phi_{m+1}(n)\begin{bmatrix} c_{m,m}(n) \\ c_{m,m-1} \\ \vdots \\ c_{m,1}(n) \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \mathscr{B}_m(n) \end{bmatrix}$$

(2) From the Equation (2) in Problem 16.8, we note that

$$\Phi_{m+1}(n) = \begin{bmatrix} \Phi_m(n) & \phi_2(n) \\ \phi_2^H(n) & U_2(n) \end{bmatrix}$$

Hence

$$\Phi_{m+1}(n)\begin{bmatrix} c_{m-1,m-1}(n) \\ c_{m-1,m-2}(n) \\ \vdots \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \Phi_m(n)\begin{bmatrix} c_{m-1,m-1}(n) \\ c_{m-1,m-2}(n) \\ \vdots \\ 1 \end{bmatrix} \\ \phi_2^H(n)\mathbf{c}_{m-1}(n) \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \mathscr{B}_{m-1}(n) \\ \phi_2^H(n)\mathbf{c}_{m-1}(n) \end{bmatrix}$$

(3) Thus far we have dealt with the last two columns of the matrix on the right-hand side of Equation (1). Similarly, we may go onto show that

$$\Phi_{m+1}(n)\begin{bmatrix} c_{1,1}(n) \\ 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \mathscr{B}_1(n) \\ \vdots \\ X \\ X \end{bmatrix}$$

where, herein and thereafter, the crosses refer to some nonzero elements. Finally, we arrive at

$$\Phi_{m+1}(n)\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \mathscr{B}_0(n) \\ X \\ \vdots \\ X \\ X \end{bmatrix}$$

Putting all of these pieces together, we conclude that

$$\Phi_{m+1}(n)\mathbf{L}_m^H(n) = \begin{bmatrix} \mathscr{B}_0(n) & 0 & \dots & 0 & 0 \\ X & \mathscr{B}_1(n) & \dots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ X & X & \dots & \mathscr{B}_{m-1}(n) & 0 \\ X & X & \dots & X & \mathscr{B}_m(n) \end{bmatrix} \tag{2}$$

Equation (2) shows that $\Phi_{m+1}(n)\mathbf{L}_m^H(n)$ is a lower triangular matrix. Since $\mathbf{L}_m(n)$ is itself a lower triangular matrix, all the elements of $\mathbf{L}_m(n)$ above the main diagonal are zero.

Moreover, since $\Phi_{m+1}(n)$ is Hermitian symmetric, $\mathbf{L}_m(n)\Phi_{m+1}(n)\mathbf{L}_m^H(n)$ is likewise hermitian symmetric. Accordingly, all the elements of this product below the main diagonal are also zero. Finally, since all the diagonal elements of $\mathbf{L}_m(n)$ equal unity, we conclude that $\mathbf{L}_m(n)\Phi_{m+1}(n)\mathbf{L}_m^H(n)$ is a diagonal matrix, as shown by

$$\begin{aligned} \mathbf{L}_m(n)\Phi_{m+1}(n)\mathbf{L}_m^H(n) =& D_{m+1}(n) \\ =& \text{diag}\left[\mathscr{B}_0(n), \mathscr{B}_1(n), \dots, \mathscr{B}_{m-1}(n), \mathscr{B}_m(n)\right] \end{aligned}$$

Problem 16.14

The joint probability density function of the time series $u(n), u(n-1), \ldots, u(n-M)$ is defined by

$$f_{\mathbf{U}}(\mathbf{u}) = \frac{1}{[2\pi \det(\mathbf{R})]^{1/2}} \exp\left(-\frac{1}{2}\mathbf{u}^H(n)\mathbf{R}^{-1}\mathbf{u}(n)\right)$$

where $\mathbf{u}^T(n) = \begin{bmatrix} u(n) & u(n-1) & \ldots & u(n-M) \end{bmatrix}$ and $\mathbf{R}$ is the $(M+1)$-by-$(M+1)$ ensemble-averaged correlation matrix of the input vector $\mathbf{u}(n)$. The log-likelihood function is correspondingly as follows:

$$\begin{aligned} L &= \ln f_{\mathbf{U}}(\mathbf{u}) \\ &= -\frac{1}{2}\ln[2\pi \det(\mathbf{R})] - \frac{1}{2}\mathbf{u}^H(n)\mathbf{R}^{-1}\mathbf{u}(n) \end{aligned} \tag{1}$$

For $n \geq M+1$, we may approximate the correlation matrix $\mathbf{R}$ in terms of the deterministic correlation matrix $\Phi(n)$ as

$$\mathbf{R} \approx \frac{1}{n}\Phi(n), \qquad n \geq M+1 \tag{2}$$

Hence, we may rewrite Equation (1) as the approximation

$$L \approx -\frac{1}{2}\ln[2\pi \det(\mathbf{R})] - \frac{1}{2n}\mathbf{u}^H(n)\Phi^{-1}\mathbf{u}(n) \tag{3}$$

The second term on the right side of Equation (3) is

$$\frac{1}{2n}\mathbf{u}^H(n)\Phi^{-1}(n)\mathbf{u}(n) = \frac{1}{2n}[1-\gamma(n)]$$

where $\gamma(n)$ is the conversion factor.

Problem 16.15

The a posteriori estimation $e_m(n)$ is order-updated by using the recursion

$$e_m(n) = e_{m-1}(n) - \kappa_m^*(n)b_m(n), \qquad m = 1, 2, \ldots, M \tag{1}$$

By definition, we have

$$e_m(n) = d(n) - \hat{d}(n|\mathscr{U}_{n-m}) \tag{2}$$

where $d(n)$ is the desired response and $\hat{d}(n|U_{n-m})$ is the least-squares estimate of $d(n)$ given the input samples $u(n), \ldots, u(n-m+1), u(n-m)$ that span the space $\mathscr{U}_{n-m}$. Similarly,

$$e_{m-1}(n) = d(n) - \hat{d}(n|\mathscr{U}_{n-m+1}) \tag{3}$$

where $\hat{d}(n|\mathscr{U}_{n-m+1})$ is the least-squares estimate of $d(n)$ given the input samples $u(n), \ldots, u(n-m+1)$. Substituting Equations (1) and (2) into (3), we get

$$\hat{d}(n|\mathscr{U}_{n-m}) = \hat{d}(n|\mathscr{U}_{n-m+1}) + h_m^*(n)b_m(n) \tag{4}$$

Equation (4) shows that given the regression coefficient h_m, we only need $b_m(n)$ as the new piece of information for updating the least-squares estimate of the desired response. Hence, $b_m(n)$ may be viewed as a form of innovation, which is in perfect agreement with the discussion presented in Section 14.1

Problem 16.16

From Equation (16.138) of the textbook:

$$\mathbf{D}_{m+1}(n) = \mathbf{L}_m(n)\Phi_{m+1}(n)\mathbf{L}_m^H(n)$$

Equivalently, we may write

$$\Phi_{m+1}^{-1}(n) = \mathbf{L}_m^H(n)\mathbf{D}_{m+1}^{-1}(n)\mathbf{L}_m(n)$$

Hence,

$$\begin{aligned}\mathbf{P}(n) =& \Phi_{m+1}^{-1}(n) \\ =& \left[\mathbf{D}^{-1/2}(n)\mathbf{L}(n)\right]^H \left[\mathbf{D}^{-1/2}(n)\mathbf{L}(n)\right]\end{aligned} \tag{1}$$

where the expression on the right-hand side of Equation (1) is exactly the Cholesky decomposition of the matrix $\mathbf{P}(n)$.

Problem 16.17

a)

From Equation (16.132)

$$\hat{x}(n+1|\mathscr{Y}_n) = \lambda^{1/2}\hat{x}(n|\mathscr{Y}_{n-1}) + g(n)\alpha(n) \tag{1}$$

From Equation (16.77) and the solution to problem 16.11:

$$g(n) \leftrightarrow \lambda^{-1/2}\mathscr{F}_{m-1}^{-1}(n)\varepsilon_{f,m-1}(n) \tag{2}$$

$$\alpha(n) \leftrightarrow \lambda^{-n/2}\gamma_{m-1}^{1/2}(n-1)\beta_m^*(n) \tag{3}$$

(It is important to note that the relation corresponding to $\alpha(n)$ must involve the error signal appropriate to the estimation problem of interest.)

Also from Table 16.3 (under backward prediction)

$$\hat{\mathbf{x}}(n+1|\mathscr{Y}_{n-1}) \leftrightarrow \left(-\lambda^{-n/2}\kappa_{b,m}(n-1)\right) \tag{4}$$

Hence, substituting Equations (2), (3) and (4) into Equation (1), and canceling the common term $\lambda^{(1/2)/2}$:

$$\kappa_{b,m}(n) = \kappa_{b,m}(n-1) - \left(\frac{\gamma_{m-1}^{1/2}(n-1)\varepsilon_{f,m-1}(n)}{\mathscr{F}_{m-1}(n)}\right)\beta_m^*(n) \tag{5}$$

But, from Equation (16.60) in the textbook:

$$\varepsilon_{f,m-1}(n-1) = \gamma_{m-1}^{1/2}(n-1)\eta_{m-1}(n) \tag{6}$$

Thus Equation (5) simplifies to

$$\kappa_{b,m}(n) = \kappa_{b,m}(n-1) - \left(\frac{\gamma_{m-1}(n-1)\eta_{m-1}(n)}{\mathscr{F}_{m-1}(n)}\right)\beta_m^*(n), \qquad m = 1, 2, \ldots, m \tag{7}$$

which is the desired result; see Equation (16.134)in the textbook.

b)

From Table 16.3 in the text; *under joint-process estimation*, and the solution to Problem 16.11:

$$\hat{x}(n|\mathscr{Y}_{n-1}) \leftrightarrow \lambda^{-n/2}h_{m-1}(n-1) \tag{10}$$

$$g \leftrightarrow \lambda^{-1/2}\beta_{m-1}^{-1}(n)\varepsilon_{b,m-1}(n) \tag{11}$$

$$\alpha(n) \leftrightarrow \lambda^{-n/2}\gamma_{m-1}^{1/2}(n-1)\xi_m^*(n) \tag{12}$$

Hence, using Equations (10) through (12) in Equation (1), we get

$$h_{m-1}(n) = h_{m-1}(n-1) + \left(\frac{\gamma_{m-1}^{1/2}(n-1)\varepsilon_{b,m-1}(n)}{\mathscr{B}_{m-1}(n)}\right)\xi_m^*(n) \tag{13}$$

From Equation (16.60), in the text replacing m with $m-1$ and n with $n-1$, we have

$$\varepsilon_{b,m-1}(n-1) = \gamma_{m-1}^{1/2}(n-1)\beta_{m-1}(n)$$

Hence,

$$h_{m-1}(n) = h_{m-1}(n-1) + \left(\frac{\gamma_{m-1}(n-1)\beta_{m-1}(n)}{\mathscr{B}_{m-1}(n)}\right)\xi_m^*(n) \tag{14}$$

which is exactly the same as Equation (16.135)

Problem 16.18

i)
Following the guidance provided in the Problem statement, we first reproduce the following equations from Table 16.6 in the text:

$$\kappa_{f,m}(n) = \kappa_{f,m}(n-1) - \frac{\gamma_{m-1}(n-1)\beta_{m-1}(n-1)}{\mathscr{B}_{m-1}(n-1)}\eta_m^*(n) \tag{1}$$

$$\kappa_{b,m} = \kappa_{f,m}^*(n) \tag{2}$$

$$\gamma_m(n) = 1 \tag{3}$$

To adapt the recursive LSL algorithm in Table 16.6 to the GAL algorithm summarized in Table 5.2, we now introduce a table that converts the first table to the second:

Approximate relation between LSL algorithm and the GAL

Recursive LSL algorithm	GAL algorithm
a) the pair of reflection coefficients, $\kappa_{f,m}(n)$, $\kappa_{b,m}(n)$	$\hat{\kappa}_m(n)$, $\hat{\kappa}_m^*(n)$
b) $\beta_{m-1}(n-1)\eta_m^*(n)$	$\tilde{\mu}\,(f_{m-1}(n)b_m^*(n) + b_{m-1}(n-1)f_m^*(n))$
c) $\frac{1}{2}\,(\mathscr{F}_{m-1}(n) + \mathscr{B}_m(n-1))$	$\varepsilon_{m-1}(n)$
d) $\gamma_{m-1}(n-1)$	1

We may justify the inclusion of the step-size parameter $\tilde{\mu}$ for entry (b) so as to "equalize" the product $\beta_{m-1}(n-1)\eta_m^*(n)$ with the summation $\frac{1}{2}\,(f_{m-1}(n)b_m^*(n) + b_{m-1}(n-1)f^*(n))$

ii) The table just described speaks for itself.

Problem 16.19

The forward reflection coefficient equals

$$\begin{aligned}\kappa_{f,m}(n) =& \frac{\Delta_{m-1}(n)}{\mathscr{B}_{m-1}(n-1)} \\ =& -\lambda\frac{\Delta_{m-1}(n-1)}{\mathscr{B}_{m-1}(n-1)} - \frac{b_{m-1}(n-1)f^*_{m-1}(n)}{\gamma_{m-1}(n-1)\mathscr{B}(n-1)} \\ =& -\lambda\frac{\mathscr{B}_{m-1}(n-2)}{\mathscr{B}_{m-1}(n-1)} \times \frac{\Delta_{m-1}(n-1)}{\mathscr{B}_{m-1}(n-2)} - \frac{b_{m-1}(n-1)f^*_{m-1}(n)}{\gamma_{m-1}(n-1)\mathscr{B}(n-1)}\xi^*_m(n)\end{aligned}$$

where, in the first term, we have multiplied and divided by $\mathscr{B}_{m-1}(n-1)$. But

$$\kappa_{f,m}(n-1) = -\frac{\Delta_{m-1}(n-1)}{\mathscr{B}_{m-1}(n-2)}$$

Hence,

$$\kappa_{f,m}(n) = \frac{\mathscr{B}_{m-1}(n-2)}{\mathscr{B}_{m-1}(n-1)}\left[\kappa_{f,m}(n-1) - \frac{b_{m-1}(n-1)f^*_{m-1}(n)}{\lambda\mathscr{B}_{m-1}(n-2)\gamma_{m-1}(n-1)}\right] \tag{1}$$

We now use the last line of part **d)** of Problem 16.4

$$\frac{\lambda\mathscr{B}_{m-1}(n-2)}{\mathscr{B}_{m-1}(n-1)} = \frac{\gamma_m(n-1)}{\gamma_{m-1}(n-1)}$$

Hence, we may rewrite Equation (1) as

$$\kappa_{f,m}(n) = \frac{\gamma_m(n-1)}{\gamma_{m-1}(n-1)}\left[\kappa_{f,m}(n-1) - \frac{b_{m-1}(n-1)f^*_{m-1}(n)}{\lambda\mathscr{B}_{m-1}(n-2)\gamma_{m-1}(n-1)}\right] \tag{2}$$

Next, the backward reflection coefficient equals

$$\begin{aligned}\kappa_{b,m}(n) =& \frac{\Delta^*_{m-1}(n)}{\mathscr{F}_{m-1}(n)} \\ =& -\lambda\frac{\Delta^*_{m-1}(n-1)}{\mathscr{F}_{m-1}(n)} - \frac{b^*_{m-1}(n-1)f_{m-1}(n)}{\mathscr{F}_{m-1}(n)\gamma_{m-1}(n-1)} \\ =& \lambda\kappa_{b,m}(n-1)\frac{\mathscr{F}_{m-1}(n-1)}{\mathscr{F}_{m-1}(n)} - \frac{b^*_{m-1}(n-1)f_{m-1}(n)}{\mathscr{F}_{m-1}(n)\gamma_{m-1}(n-1)} \\ =& \lambda\frac{\mathscr{F}_{m-1}(n-1)}{\mathscr{F}_{m-1}(n)}\left[\kappa_{b,m}(n-1) - \lambda^{-1}\frac{b^*_{m-1}(n-1)f_{m-1}(n)}{\mathscr{F}_{m-1}(n-1)\gamma_{m-1}(n-1)}\right]\end{aligned} \tag{3}$$

We now reintroduce the last line of part **c)** of Problem 16.4 in the equivalent form:

$$\lambda \frac{\mathscr{F}_{m-1}(n-1)}{\mathscr{F}_{m-1}(n)} = \frac{\gamma_m(n)}{\gamma_{m-1}(n-1)} \tag{3}$$

Hence, we may rewrite Equation (2) as follows

$$\kappa_{b,m}(n) = \frac{\gamma_m(n)}{\gamma_{m-1}(n-1)} \left[\kappa_{b,m}(n-1) - \lambda^{-1} \frac{b^*_{m-1}(n-1) f_{m-1}(n)}{\mathscr{F}_{m-1}(n-1)\gamma_{m-1}(n-1)} \right] \tag{4}$$

Equations (2) and (4) are the solution to the problem.

Problem 16.20

The forward a posteriori prediction error equals

$$\begin{aligned} f_m(n) =& f_{m-1}(n) + \kappa^*_{f,m}(n) b_{m-1}(n-1) \\ =& f_{m-1}(n) - \frac{\Delta^*_{m-1}(n)}{\mathscr{B}_{m-1}(n-1)} b_{m-1}(n-1) \end{aligned}$$

Hence, the normalized value of $f_m(n)$ equals

$$\begin{aligned} \bar{f}_m(n) =& \frac{f_m(n)}{\mathscr{F}^{1/2}_m(n)\gamma^{1/2}_m(n-1)} \\ =& \frac{f_{m-1}(n)}{\mathscr{F}^{1/2}_m(n)\gamma^{1/2}_m(n-1)} - \frac{\Delta^*_{m-1}(n)}{\mathscr{B}_{m-1}(n-1)\mathscr{F}^{1/2}_m(n)\gamma^{1/2}_m(n-1)} b_{m-1}(n-1) \end{aligned} \tag{1}$$

But

$$\begin{aligned} \mathscr{F}_m(n) =& \mathscr{F}_{m-1}(n) \left[1 - \frac{|\Delta_{m-1}(n)|^2}{\mathscr{F}_{m-1}(n)\mathscr{B}_{m-1}(n-1)} \right] \\ =& \mathscr{F}_{m-1}(n) \left[1 - \bar{\Delta}_{m-1}(n) \right] \end{aligned} \tag{2}$$

where

$$\bar{\Delta}_{m-1}(n) = \frac{\Delta_{m-1}(n)}{\mathscr{F}^{1/2}_{m-1}(n)\mathscr{B}^{1/2}_{m-1}(n-1)}$$

Similarly

$$\begin{aligned}\mathscr{B}_m(n) =&\mathscr{B}_{m-1}(n)\left[1-\frac{|\Delta_{m-1}(n)|^2}{\mathscr{B}_{m-1}(n-1)\mathscr{F}_{m-1}(n)}\right]\\ =&\mathscr{B}_{m-1}(n-1)\left[1-\left|\bar{\Delta}_{m-1}(n)\right|^2\right]\end{aligned} \tag{3}$$

Also, we may write

$$\begin{aligned}\gamma_m(n-1) =&\gamma_{m-1}(n-1)\left[1-\frac{|b_{m-1}(n-1)|^2}{\gamma_{m-1}(n-1)\mathscr{B}_{m-1}(n-1)}\right]\\ =&\gamma_{m-1}(n-1)\left[1-\left|\bar{b}_{m-1}(n-1)\right|^2\right]\end{aligned} \tag{4}$$

where

$$\bar{b}_{m-1}(n-1)=\frac{b_{m-1}(n-1)}{\gamma_{m-1}^{1/2}(n-1)\mathscr{B}_{m-1}^{1/2}(n-1)} \tag{5}$$

Hence, we may use Equations (2) and (4) to express the first term on the right side of Equation (1) as

$$\begin{aligned}\frac{f_{m-1}(n)}{\mathscr{F}_m^{1/2}(n)\gamma_m^{1/2}(n-1)} =&\frac{f_{m-1}(n)/\left[\mathscr{F}_{m-1}^{1/2}(n)\gamma_{m-1}^{1/2}(n-1)\right]}{\left[1-\left|\bar{\Delta}_{m-1}(n)\right|^2\right]^{1/2}\left[1-\left|\bar{b}_{m-1}(n-1)\right|^2\right]^{1/2}}\\ =&\frac{\bar{f}_{m-1}(n)}{\left[1-\left|\bar{\Delta}_{m-1}(n)\right|^2\right]^{1/2}\left[1-\left|\bar{b}_{m-1}(n-1)\right|^2\right]^{1/2}}\end{aligned} \tag{6}$$

Next, we use Equations (2) to (4) to express the second term on the right side of Equation (1) as

$$\begin{aligned}&\frac{\Delta_{m-1}^*(n)b_{m-1}(n-1)}{\mathscr{B}_{m-1}(n-1)\mathscr{F}_m^{1/2}(n)\gamma_m^{1/2}(n-1)}\\ &\quad=\frac{\Delta_{m-1}^*(n)b_{m-1}(n-1)}{\mathscr{B}_{m-1}(n-1)\mathscr{F}_m^{1/2}(n)\gamma_m^{1/2}(n-1)\left[1-\left|\bar{\Delta}_{m-1}(n)\right|^2\right]^{1/2}\left[1-\left|\bar{b}_{m-1}(n-1)\right|^2\right]^{1/2}}\\ &\quad=\frac{\bar{\Delta}_{m-1}^*(n)\bar{b}_{m-1}(n-1)}{\left[1-\left|\bar{\Delta}_{m-1}(n)\right|^2\right]^{1/2}\left[1-\left|\bar{b}_{m-1}(n-1)\right|^2\right]^{1/2}}\end{aligned} \tag{7}$$

Substituting Equations (6) and (7) into Equation (1), we may thus write

$$\bar{f}_m(n) = \frac{\bar{f}_{m-1}(n) - \bar{\Delta}^*_{m-1}(n)\bar{b}_{m-1}(n-1)}{\left[1 - \left|\bar{\Delta}_{m-1}(n)\right|^2\right]^{1/2}\left[1 - \left|\bar{b}_{m-1}(n-1)\right|^2\right]^{1/2}} \tag{8}$$

Next, the backward a posteriori prediction error is

$$b_m(n) = b_{m-1}(n-1) - \frac{\Delta_{m-1}(n)}{\mathscr{F}_{m-1}(n)} f_{m-1}(n)$$

Hence, the normalized value of $b_m(n)$ equals

$$\bar{b}_m(n) = \frac{b_{m-1}(n-1)}{\mathscr{B}^{1/2}_m(n)\gamma^{1/2}_m(n)} - \frac{\Delta_{m-1}(n) f_{m-1}(n)}{\mathscr{F}_{m-1}(n)\mathscr{B}^{1/2}_m(n)\gamma^{1/2}_m(n)} \tag{9}$$

The conversion factor may be updated by using the recursion:

$$\begin{aligned}
\gamma_m(n) =& \gamma_{m-1}(n-1) - \frac{\left|f_{m-1}(n)\right|^2}{\mathscr{F}_{m-1}(n)} \\
=& \gamma_{m-1}(n-1)\left[1 - \frac{\left|f_{m-1}(n)\right|^2}{\mathscr{F}_{m-1}(n)\gamma_{m-1}(n-1)}\right] \\
=& \gamma_{m-1}(n-1)\left[1 - \left|\bar{f}_{m-1}(n)\right|^2\right]
\end{aligned} \tag{10}$$

where

$$\bar{f}_{m-1}(n) = \frac{f_{m-1}(n)}{\mathscr{F}^{1/2}_{m-1}(n)\gamma^{1/2}_{m-1}(n-1)}$$

Next, using Equations (3) and (10), we may express the first term on the right side of (Equation 9) as

$$\begin{aligned}
\frac{b_{m-1}(n-1)}{\mathscr{B}^{1/2}_m(n)\gamma^{1/2}_m(n)} =& \frac{b_{m-1}(n-1)/\left[\mathscr{B}^{1/2}_{m-1}(n-1)\gamma^{1/2}_{m-1}(n-1)\right]}{\left[1 - \left|\bar{\Delta}_{m-1}(n)\right|^2\right]^{1/2}\left[1 - \left|\bar{f}_{m-1}(n)\right|^2\right]^{1/2}} \\
=& \frac{\bar{b}_{m-1}(n-1)}{\left[1 - \left|\bar{\Delta}_{m-1}(n)\right|^2\right]^{1/2}\left[1 - \left|\bar{f}_{m-1}(n)\right|^2\right]^{1/2}}
\end{aligned} \tag{11}$$

Using Equations (3) and (9), we may also rewrite the second term on the right side of Equation (9) as

$$\begin{aligned}
&\frac{\Delta_{m-1}(n)f_{m-1}(n)}{\mathscr{F}_{m-1}(n)\mathscr{B}_m^{1/2}(n)\gamma_m^{1/2}(n)} \\
&\quad = \frac{\Delta_{m-1}(n)f_{m-1}(n)}{\mathscr{F}_{m-1}(n)\mathscr{B}_{m-1}^{1/2}(n-1)\gamma_{m-1}^{1/2}(n-1)\left[1-\left|\bar{\Delta}_{m-1}(n)\right|^2\right]^{1/2}\left[1-\left|\bar{f}_{m-1}(n)\right|^2\right]^{1/2}} \\
&\quad = \frac{\bar{\Delta}_{m-1}(n)\bar{f}_{m-1}(n)}{\left[1-\left|\bar{\Delta}_{m-1}(n)\right|^2\right]^{1/2}\left[1-\left|\bar{f}_{m-1}(n)\right|^2\right]^{1/2}}
\end{aligned} \tag{12}$$

Hence, the use of Equations (11) and (12) in Equation (9) yields

$$\bar{b}_m(n) = \frac{\bar{b}_{m-1}(n-1) - \bar{\Delta}_{m-1}(n)\bar{f}_{m-1}(n)}{\left[1-\left|\bar{\Delta}_{m-1}(n)\right|^2\right]^{1/2}\left[1-\left|\bar{f}_{m-1}(n)\right|^2\right]^{1/2}} \tag{13}$$

Finally, we note that

$$\Delta_{m-1}(n) = \lambda\Delta_{m-1}(n-1) + \frac{b_{m-1}(n-1)f_{m-1}^*(n)}{\gamma_{m-1}(n-1)}$$

The normalized value of $\Delta_{m-1}(n)$ is therefore

$$\begin{aligned}
\bar{\Delta}_{m-1}(n) =& \frac{\Delta_{m-1}(n)}{\mathscr{F}_{m-1}^{1/2}(n)\mathscr{B}_{m-1}^{1/2}(n-1)} \\
=& \lambda\frac{\Delta_{m-1}(n)}{\mathscr{F}_{m-1}^{1/2}(n)\mathscr{B}_{m-1}^{1/2}(n-1)} + \frac{b_{m-1}(n-1)f_{m-1}^*(n)}{\mathscr{F}_{m-1}^{1/2}(n)\mathscr{B}_{m-1}^{1/2}(n-1)\gamma_{m-1}(n-1)}
\end{aligned} \tag{14}$$

Using Equations (2) and (3), we may rewrite the first term on the right side of Equation (14) as

$$\lambda\frac{\Delta_{m-1}(n-1)}{\mathscr{F}_{m-1}^{1/2}(n)\mathscr{B}_{m-1}^{1/2}(n-1)} = \frac{\Delta_{m-1}(n)}{\mathscr{F}_{m-1}^{1/2}(n)\mathscr{B}_{m-1}^{1/2}(n-1)} \times \frac{\gamma\mathscr{F}_{m-1}^{1/2}(n-1)\mathscr{B}_{m-1}^{1/2}(n-2)}{\mathscr{F}_{m-1}^{1/2}(n-1)\mathscr{B}_{m-1}^{1/2}(n-1)} \tag{15}$$

We next use the relations:

$$\frac{\lambda\mathscr{F}_{m-1}(n-1)}{\mathscr{F}_{m-1}(n)} = \frac{\gamma_m(n)}{\gamma_{m-1}(n-1)}$$

$$\frac{\lambda\mathscr{B}_{m-1}(n-2)}{\mathscr{B}_{m-1}(n-1)} = \frac{\gamma_m(n-1)}{\gamma_{m-1}(n-1)}$$

$$\bar{\Delta}_{m-1}(n-1) = \frac{\Delta_{m-1}(n-1)}{\mathscr{F}_{m-1}^{1/2}(n-1)\mathscr{B}_m^{1/2}(n-2)}$$

Hence, we may rewrite Equation (15) as

$$\lambda\frac{\Delta_{m-1}(n-1)}{\mathscr{F}_{m-1}^{1/2}(n)\mathscr{B}_{m-1}^{1/2}(n-1)} = \bar{\Delta}_{m-1}(n-1)\frac{\gamma^{1/2}(n)\gamma_m^{1/2}(n-1)}{\gamma_{m-1}(n-1)} \tag{16}$$

Next, multiplying Equation (4) by Equation (10), we have

$$\frac{\gamma_m(n)\gamma_m(n-1)}{\gamma_{m-1}^2(n-1)} = \left[1-\left|\bar{f}_{m-1}(n)\right|^2\right]\left[1-\left|\bar{b}_{m-1}(n-1)\right|^2\right]$$

Accordingly, we may rewrite Equation (16) as

$$\lambda\frac{\Delta_{m-1}(n-1)}{\mathscr{F}_{m-1}^{1/2}(n)\mathscr{B}_{m-1}^{1/2}(n-1)} = \bar{\Delta}_{m-1}(n-1)\left[1-\left|\bar{f}_{m-1}(n)\right|^2\right]^{1/2}\left[1-\left|\bar{b}_{m-1}(n-1)\right|^2\right]^{1/2} \tag{17}$$

Next, the second term on the right side of Equation (14) is

$$\frac{b_{m-1}(n-1)f_{m-1}^*(n)}{\mathscr{F}_{m-1}^{1/2}(n)\mathscr{B}_{m-1}^{1/2}(n-1)\gamma_{m-1}(n-1)} = \bar{b}_{m-1}(n-1)\bar{f}_{m-1}^*(n) \tag{18}$$

Thus, substituting Equations (17) and (18) into Equation (14), we get the desired time-update

$$\begin{aligned}\bar{\Delta}_{m-1}(n) =& \bar{\Delta}_{m-1}(n-1)\left[1-\left|\bar{f}_{m-1}(n)\right|^2\right]^{1/2}\left[1-\left|\bar{b}_{m-1}(n-1)\right|^2\right]^{1/2} \\ &+ \bar{b}_{m-1}(n-1)\bar{f}_{m-1}^*(n)\end{aligned} \tag{19}$$

Ordering Equations (19), (8) and (13), in this manner, we constitute the normalized LSL algorithm.

Chapter 17

Problem 17.1

a)

The received signal of a digital communication system in baseband form is given by

$$u(t) = \sum_{m=-\infty}^{\infty} x_m h(t - mT) + \nu(t)$$

where x_m is the transmitted symbol, $h(t)$ is the overall impulse response, T is the symbol period, and $\nu(t)$ is the channel noise. Evaluating $u(t)$ at time t_1 and t_2:

$$u(t) = \sum_{m=-\infty}^{\infty} x_m h(t_1 - mT) + \nu(t_1)$$

$$u(t) = \sum_{\ell=-\infty}^{\infty} x_\ell h(t_2 - \ell T) + \nu(t_2)$$

Hence, the autocorrelation function of $u(t)$ is

$$\begin{aligned} r_u(t_1, t_2) =& \mathbb{E}\left[u(t_1)u(t_2)\right] \\ =& \mathbb{E}\left[\sum_{m=-\infty}^{\infty}\sum_{\ell=-\infty}^{\infty} x_m x_\ell^* h(t_1 - mT)h(t_2 - \ell T)\right] + \mathbb{E}\left[\nu(t_1)\nu^*(t_2)\right] \\ =& \sum_{m=-\infty}^{\infty}\sum_{\ell=-\infty}^{\infty} r_x(mT - \ell T)h(t_1 - mT)h^*(t_2 - \ell T) + \sigma_\nu^2\delta(t_1 - t_2) \end{aligned} \quad (1)$$

where $r_x(mT - \ell T)$ is the autocorrelation function of the transmitted signal. From Equation (1) we immediately see that

$$r_u(t_1 + T, t_2 + T) = r_u(t_1, t_2)$$

which shows that $u(t)$ is indeed cyclostationary in the wide sense.

b)
Applying the following definitions

$$r_u^\alpha(\tau) = \frac{1}{\tau}\int_{-T/2}^{T/2} r_u\left(t+\frac{\tau}{2}, t-\frac{\tau}{2}\right)\exp\left(\mathrm{j}\,2\pi\alpha t\right)\mathrm{d}\,t$$

$$S_u^\alpha(\omega) = \int_{\text{-}\infty}^{\infty} r_u^\alpha(\tau)\exp\left(\mathrm{j}\,2\pi f\tau\right)\mathrm{d}\,\tau, \qquad \omega = 2\pi f$$

$$\alpha = \frac{k}{T}, \qquad k = 0, \pm 1, \pm 2, \ldots$$

to the result obtained in part **a)**, we may show that

$$S_u^{k/T} = \frac{1}{T}H(\exp(\mathrm{j}\,\omega + \mathrm{j}\,k\pi/T))H^*(\exp(\mathrm{j}\,\omega - \mathrm{j}\,k\pi/T))S_x\left(\omega + \frac{k\pi}{T}\right) + \sigma_\nu^2\delta(k), \qquad k = 0, \pm 1, \pm 2, \ldots \tag{2}$$

As a check, we see that for $k = 0$, Equation (2) reduces to the standard result:

$$S_u(\omega) = \frac{1}{T}\left|H(\exp(\mathrm{j}\,\omega))\right|^2 S_x(\omega) + \sigma_\nu^2$$

Let $\Psi_k(\omega)$ denote the phase response of $S_u^{k/T}$ and $\Phi(\omega)$ denote the phase response of the channel. Then recognizing that the power spectral density $S_x(\omega)$ of the transmitted signal is real valued, we readily find from Equation (2) that

$$\Psi_k(\omega) = \Phi\left(\omega + \frac{k\pi}{T}\right) - \Phi\left(\omega - \frac{k\pi}{T}\right), \qquad k = 0, \pm 1, \pm 2, \ldots \tag{3}$$

c)
From the formula for the inverse Fourier transform, we have

$$\psi_k(\tau) = \frac{1}{2\pi}\int_{\text{-}\infty}^{\infty}\Psi_k(\omega)\exp(\mathrm{j}\,\omega\tau)\,\mathrm{d}\,\omega$$

$$\phi_k(\tau) = \frac{1}{2\pi}\int_{\text{-}\infty}^{\infty}\Phi_k(\omega)\exp(\mathrm{j}\,\omega\tau)\,\mathrm{d}\,\omega$$

Applying these definitions to Equation (3):

$$\begin{aligned}\psi_k(\tau) =& \phi(\tau)\exp(-\,\mathrm{j}\,\pi k\tau/T) - \phi(\tau)\exp(\mathrm{j}\,\pi k\tau/T)\\ =& -2\,\mathrm{j}\,\phi(\tau)\sin\left(\frac{k\pi\tau}{T}\right), \qquad k = 0, \pm 1, \pm 2, \ldots\end{aligned} \tag{4}$$

On the basis of Equation (4), we may make two important observations:

1. For $k = 0$ and $k\tau/T$ equal to an integer, $\phi_k(\tau)$ is identically zero. For these values of k, $\phi(\tau)$ cannot be determined. This means that for an arbitrary channel, the unknown phase response $\Phi(f)$ cannot be identified for $k = 0$ or $k\tau/T =$ integer by using cyclostationary second-order statistics of the channel output.

2. For $k = \pm 2$ and higher in absolute value, the use of $\psi_k(\tau)$ does not reveal any more information abotu the channel phase response than what can be obtained with $k = \pm 1$. We may therefore just as well work with $k = 1$, for which $\phi_k(\tau)$ has the largest support, as shown by

   $$\psi_1(\tau) = -2\,\mathrm{j}\,\phi(\tau)\sin\left(\frac{\pi\tau}{T}\right)$$

   That is

   $$\phi(\tau) = \frac{\mathrm{j}\,\psi_1(\tau)}{2\sin(\pi\tau/T)}$$

   which shows that $\phi(\tau)$ is identifiable from $\psi_1(\tau)$ expect for $\tau = mT$, where m is an integer.

Problem 17.2

In the noise-free case, we have

$$\mathbf{U}_n = \mathcal{H}\mathbf{x}_n \tag{1}$$

where $\mathcal{H}$ is the LN-by-$(M + N)$ multichannel filtering matrix, $\mathbf{x}_n$ is the $(M + N)$-by-1 transmitted signal vector, and $\mathbf{u}_n$ is the LN-by-1 multichannel recieved signal vector. Let $\mathbf{u}_n$ be applied to a multichannel structre characterized by the (M_N)-by-LN filtering matrix $\mathbf{T}$ such as we have

$$\mathbf{T}\mathbf{U}_n = \mathbf{x}_n \tag{2}$$

The zero-forcing condition ensures the perfect recovery of $\mathbf{x}_n$ from $\mathbf{u}_n$. Substituting Equation (1) into Equation (2):

$$\mathbf{T}\mathcal{H} = \mathbf{I} \tag{3}$$

Where $\mathbf{I}$ is the identity matrix. From Equation (3) we immediately deduce that

$$\mathbf{T} = \mathcal{H}^{+}$$

where $\mathcal{H}^{+}$ is the pseudoinverse of $\mathcal{H}$.

Problem 17.3

The relationship between $\mathbf{g}$ and $\mathcal{H}$ on the one hand and $\mathbf{h}$ and $\mathcal{G}_k$ on the other hand, as described in Equation (17.27), namely

$$\mathbf{g}_k \mathcal{H}\mathcal{H}^H \mathbf{g}_k = \mathbf{h}^H \mathcal{G}_k \mathcal{G}_k \mathbf{h}$$

follows directly from the two sets of definitions

$$\mathcal{H} = \begin{bmatrix} \mathcal{H}^{(0)} \\ \mathcal{H}^{(1)} \\ \vdots \\ \mathcal{H}^{(L-1)} \end{bmatrix}, \qquad \mathcal{H}^{(\ell)} = \begin{bmatrix} h_0^{(\ell)} & \dots & h_M^{(\ell)} & \dots & 0 \\ 0 & \dots & h_M^{(\ell)} & \dots & 0 \\ \vdots & & \vdots & & \vdots \\ 0 & \dots & h_0^{(\ell)} & \dots & h_M^{(\ell)} \end{bmatrix}$$

$$\mathbf{g}_k = \begin{bmatrix} \mathbf{g}_k^{(0)} \\ \mathbf{g}_k^{(1)} \\ \vdots \\ \mathbf{g}_k^{(L-1)} \end{bmatrix}$$

$$\mathcal{G} = \begin{bmatrix} \mathcal{G}^{(0)} \\ \mathcal{G}^{(1)} \\ \vdots \\ \mathcal{G}^{(L-1)} \end{bmatrix}, \qquad \mathcal{G}^{(\ell)} = \begin{bmatrix} g_{k,0}^{(\ell)} & \dots & g_{k,N-1}^{(\ell)} & \dots & 0 \\ 0 & \dots & g_{k,N-2}^{(\ell)} & \dots & 0 \\ \vdots & & \vdots & & \vdots \\ 0 & \dots & g_{k,0}^{(\ell)} & \dots & g_{k,N-1}^{(\ell)} \end{bmatrix}$$

$$\mathbf{h} = \begin{bmatrix} \mathbf{h}^{(0)} \\ \mathbf{h}^{(1)} \\ \vdots \\ \mathbf{h}^{(L-1)} \end{bmatrix}$$

Problem 17.4

For a noiseless channel, the received signal is

$$u_n^{(\ell)} = \sum_{m=0}^{M} h_n^{(\ell)} x_{n-m}, \qquad \ell = 0, 1, \ldots, L-1 \tag{1}$$

By definition, we have

$$\mathbf{H}^{\ell}(z) = \sum_{m=0}^{M} h_m^{(\ell)} z^{-m}$$

where z is the unit-delay operator. We therefore rewrite Equation (1) in the equivalent from:

$$u_n^{(\ell)} = \mathbf{H}^{\ell}(z)\left[x_n\right] \tag{2}$$

where $\mathbf{H}^{\ell}(z)$ acts as an operator. Multiplying Equation (2) by $\mathbf{G}^{(\ell)}(z)$ and then summing over ℓ:

$$\sum_{\ell=0}^{L-1} \mathbf{G}^{\ell}(z)\left[u_n^{(\ell)}\right] = \sum_{\ell=0}^{L-1} \mathbf{G}^{\ell}(z)\mathbf{H}^{\ell}(z)\left[x_n\right] \tag{3}$$

According to the generalized Bezout identity:

$$\sum_{\ell=0}^{L-1} \mathbf{G}^{\ell}(z)\mathbf{H}^{\ell}(z) = 1$$

We may therefore simplify Equation (3) to

$$\sum_{\ell=0}^{L-1} \mathbf{G}^{\ell}(z)\left[u_n^{(\ell)}\right] = x_n$$

Let

$$y^{(\ell)}(n) = \mathbf{G}^{\ell}(z)\left[u_n^{(\ell)}\right], \qquad \ell = 0, 1, \ldots, L-1$$

Let $\mathbf{G}^{(\ell)}(z)$ be written in the expanded form:

$$\mathbf{G}^{(\ell)}(z) = \sum_{i=0}^{M} g_i^{(l)} z^{-i}$$

Hence,

$$\begin{aligned} y^{(\ell)}(n) &= \sum_{i=0}^{M} g_i^{(\ell)} z^{-i} \left[u_n^{(\ell)}\right] \\ &= \sum_{i=0}^{M} g_i^{(\ell)} u_{n-i}^{(\ell)} \end{aligned}$$

From linear prediction theory, we recognize that

$$\hat{u}_{n+1}^{(\ell)} = \sum_{i=0}^{M} g_i^{(\ell)} u_{n-i}^{(\ell)}$$

It follows therefore that

$$y^{(\ell)}(n) = z^{-i}\left[\hat{u}_{n+1}^{(\ell)}\right]$$

and so we may rewrite

$$x_n = \sum_{\ell=0}^{L-1} z^{-i}\left[\hat{u}_{n+1}^{(\ell)}\right]$$

Problem 17.5

We are given

$$\hat{x} = \frac{1}{f_Y(y)} \int_{-\infty}^{\infty} x f_V(y - c_0 x) f_X(x)\,\mathrm{d}\,x$$

where

$$f_X(x) = \begin{cases} \frac{1}{2\sqrt{3}} & -\sqrt{3} \leq x \leq \sqrt{3} \\ \\ 0, & \text{otherwise} \end{cases}$$

$$f_V(v) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{v^2}{2\sigma^2}\right),$$

and

$$f_Y(y) = \int_{-\infty}^{\infty} f_x(x) f_V(y - c_0 x)\,\mathrm{d}\,x$$

Hence,

$$\hat{x} = \frac{\displaystyle\int_{-\sqrt{3}}^{\sqrt{3}} x \exp\left(-\frac{(y - c_0 x)^2}{2\sigma^2}\right) \mathrm{d}\, x}{\displaystyle\int_{-\sqrt{3}}^{\sqrt{3}} \exp\left(-\frac{(y - c_0 x)^2}{2\sigma^2}\right) \mathrm{d}\, x} \tag{1}$$

Let

$$t = \frac{y - c_0 x}{\sigma}$$

$$\mathrm{d}\, t = \frac{-c_0 \,\mathrm{d}\, x}{\sigma}$$

Then, we may rewrite Equation (1) as

$$\begin{aligned}
\hat{x} =& \frac{\displaystyle\int_{\frac{y - \sqrt{3}c_0}{\sigma}}^{\frac{y + \sqrt{3}c_0}{\sigma}} \frac{\sigma}{c_0^2} (y - t\sigma) \exp\left(-\frac{t^2}{2}\right) \mathrm{d}\, t}{\displaystyle\int_{\frac{y - \sqrt{3}c_0}{\sigma}}^{\frac{y + \sqrt{3}c_0}{\sigma}} \frac{\sigma}{c_0} \exp\left(-\frac{t^2}{2}\right) \mathrm{d}\, t} \\
=& \frac{1}{c_0} y - \frac{\sigma}{c_0} \frac{\displaystyle\int_{\frac{y - \sqrt{3}c_0}{\sigma}}^{\frac{y + \sqrt{3}c_0}{\sigma}} t \exp\left(-\frac{t^2}{2}\right) \mathrm{d}\, t}{\displaystyle\int_{\frac{y - \sqrt{3}c_0}{\sigma}}^{\frac{y + \sqrt{3}c_0}{\sigma}} \exp\left(-\frac{t^2}{2}\right) \mathrm{d}\, t}
\end{aligned} \tag{2}$$

Next we recognize the following two results

$$\begin{aligned}
\int_{\frac{y - \sqrt{3}c_0}{\sigma}}^{\frac{y + \sqrt{3}c_0}{\sigma}} t \exp\left(-\frac{t^2}{2}\right) \mathrm{d}\, t &= \exp(-\frac{t^2}{2})\Big|_{(y - \sqrt{3}c_0)/\sigma}^{(y + \sqrt{3}c_0)/\sigma} \\
&= \sqrt{2\pi} \left\{ Z\left(\frac{y + \sqrt{3}c_0}{\sigma}\right) - Z\left(\frac{y - \sqrt{3}c_0}{\sigma}\right) \right\}
\end{aligned}$$

$$\int_{\frac{y-\sqrt{3}c_0}{\sigma}}^{\frac{y+\sqrt{3}c_0}{\sigma}} \exp\left(-\frac{t^2}{2}\right) \mathrm{d}\, t = \int_{\frac{y-\sqrt{3}c_0}{\sigma}}^{\infty} \exp\left(-\frac{t^2}{2}\right) \mathrm{d}\, t - \int_{\frac{y+\sqrt{3}c_0}{\sigma}}^{\infty} \exp\left(-\frac{t^2}{2}\right) \mathrm{d}\, t$$
$$= \sqrt{2\pi}\left\{Q\left(\left(y-\sqrt{3}c_0\right)/\sigma\right) - Q\left(\left(y+\sqrt{3}c_0\right)/\sigma\right)\right\}$$

We may therefore rewrite Equation (2) in the compact form:

$$\hat{x} = \frac{y}{c_0} + \frac{\sigma}{c_0}\frac{Z\left(\left(y+\sqrt{3}c_0\right)/\sigma\right) - Z\left(\left(y-\sqrt{3}c_0\right)/\sigma\right)}{Q\left(\left(y-\sqrt{3}c_0\right)/\sigma\right) - Q\left(\left(y+\sqrt{3}c_0\right)/\sigma\right)}$$

Problem 17.6

For convergence of a Bussgang algorithm:

$$\mathbb{E}\left[y(n)y(n+k)\right] = \mathbb{E}\left[y(n)g\left(y(n+k)\right)\right]$$

For large n:

$$\mathbb{E}\left[y(n)^2\right] = \mathbb{E}\left[y(n)g\left(y(n)\right)\right]$$

For $y(n) = x(n)$ to achieve perfect equalization:

$$\mathbb{E}\left[\hat{x}^2\right] = \mathbb{E}\left[xg\left(x\right)\right]$$

With $\hat{x}$ being of zero mean and unit variance, we thus have

$$\mathbb{E}\left[\hat{x}g(x)\right] = 1$$

Problem 17.7

We start (for real-valued data)

$$\begin{aligned} y(n) =& \sum_i \hat{w}_i(n)u(n-i) \\ =& \hat{\mathbf{w}}^T(n)\mathbf{u}(n) \\ =& \mathbf{u}^T(n)\hat{\mathbf{w}}(n) \end{aligned}$$

Let

$$\mathbf{y} = \begin{bmatrix} y(n_1) & y(n_2) & \dots & y(n_K) \end{bmatrix}^T$$

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}^T(n_1) \\ \mathbf{u}^T(n_2) \\ \vdots \\ \mathbf{u}^T(n_K) \end{bmatrix}$$

Assuming that $\hat{\mathbf{w}}(n)$ has a constant value $\hat{\mathbf{w}}$, averaged over the whole block of data, then

$$\mathbf{y} = \mathbf{U}\hat{\mathbf{w}}$$

The solution for $\hat{\mathbf{w}}$ is

$$\hat{\mathbf{w}} = \mathbf{U}^+\mathbf{y}$$

where $\mathbf{U}^+$ is the pseudoinverse of the matrix $\mathbf{U}$.

Problem 17.8

a)

For the binary PSK system, a plot of the error signal versus the equalizer output has the form shown by the continuous curve in Fig. 1:

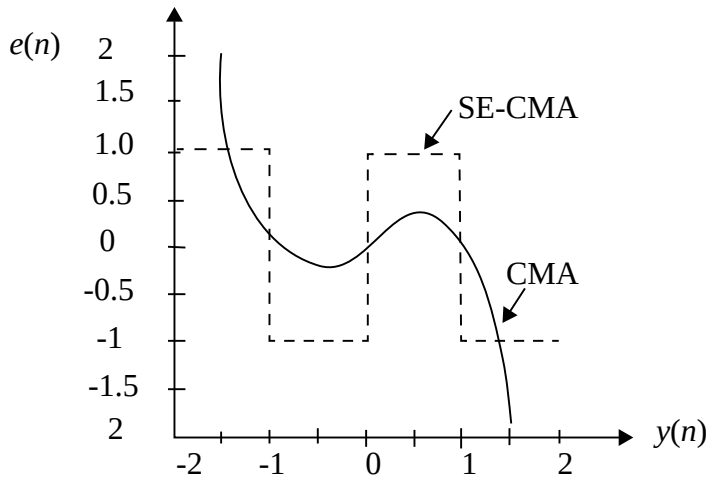


Figure 1

b)
The corresponding plot for the signed-error (SE) version of the CMA is shown by the dashed line in Figure 1.

c)
CMA is a stochastic algorithm minimizing the Godard criterion

$$J_{cm} = \frac{1}{4}\mathbb{E}\left[\left[|y_n|^2 - R_2\right]^2\right]$$

where the positive constant R_2 is the dispersion constant, which is chosen in accordance with the source statistics. For a fractionally spaced equalizer (FSE) update algorithm, the algorithm is described by

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\mathbf{u}(n)\underbrace{y^*(n)\gamma^2 - |y(n)|^2}_{\psi(y_n)}, \qquad \gamma^2 = 1 + R_2$$

where μ is small positive step size and $\psi(\cdot)$ is called the CMA error function.

The signed-error SE-CMA algorithm is described as follows:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\mathbf{u}(n)\,\mathrm{sgn}\,\psi(y(n))$$

where $\mathrm{sgn}(\cdot)$ denotes the signum function. Specifically,

$$\mathrm{sgn}(x) = \begin{cases} 1 & \text{for } x > 0 \\ -1 & \text{for } x < 0 \end{cases}$$

The SE-CMA is computationally more efficient than CMA.

Problem 17.9

The update function for the constant modulus algorithm (CMA) is described by

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu\mathbf{u}(n)e^*(n)$$

where

$$e(n) = y(n)\left(\gamma^2 - |y(n)|^2\right), \qquad \gamma^2 = 1 + R_2$$

In quadriphase-shift keying (QPSK), the output signal $y(n)$ is complex-valued, as shown by

$$y(n) = y_I(n) + \mathrm{j}\, y_Q(n), \qquad \begin{array}{l} y_I(n) = \text{in-phase comonent} \\ y_I(n) = \text{quadrature comonent} \end{array}$$

Hence,

$$e_I(n) = y_I(n)\left(R_2 - y_I^2(n) - y_Q^2(n)\right)$$

$$e_Q(n) = y_Q(n)\left(R_2 - y_I^2(n) - y_Q^2(n)\right)$$

For the signed CMA, we thus have

$$\begin{aligned} \hat{\mathbf{w}}(n+1) =& \hat{\mathbf{w}}(n) + \mu \mathbf{u}(n) \operatorname{sgn} e(n) \\ =& \hat{\mathbf{w}}(n) + \mu \mathbf{u} \operatorname{sgn}\left[e_I(n) + \mathrm{j}\, e_Q(n)\right] \end{aligned}$$

The weights are complex valued. Hence, following the analysis presented in Chapter 5 of the text, we may write

$$\hat{\mathbf{w}}_I(n+1) = \hat{\mathbf{w}}_I(n) + \mu\left(\mathbf{u}_I(n) \operatorname{sgn}\left[e_I(n)\right] - \mathbf{u}_Q(n) \operatorname{sgn}\left[e_Q(n)\right]\right) \tag{1}$$

$$\hat{\mathbf{w}}_Q(n+1) = \hat{\mathbf{w}}_Q(n) + \mu\left(\mathbf{u}_Q(n) \operatorname{sgn}\left[e_I(n)\right] - \mathbf{u}_I(n) \operatorname{sgn}\left[e_Q(n)\right]\right) \tag{2}$$

where

$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}_I(n) + \mathrm{j}\, \hat{\mathbf{w}}_Q(n)$$

$$\mathbf{u}(n) = \mathbf{u}_I(n) + \mathrm{j}\, \mathbf{u}_Q(n)$$

The standard version of the complex CMA is as follows:

$$\hat{\mathbf{w}}_I(n+1) = \hat{\mathbf{w}}_I(n) + \mu\left(\mathbf{u}_I(n) e_I(n) - \mathbf{u}_Q(n) e_Q(n)\right) \tag{3}$$

$$\hat{\mathbf{w}}_Q(n+1) = \hat{\mathbf{w}}_Q(n) + \mu\left(\mathbf{u}_Q(n) e_I(n) - \mathbf{u}_I(n) e_Q(n)\right) \tag{4}$$

Both versions of the CMA, the signed version of Eqs. (1) and (2) and the standard version of Equations (3) and (4), can now be treated in the same way as the real-valued CMA in Problem 17.8.

Problem 17.10

Dithered signed-error version of CMA, hereafter refereed to as DSE-CMA:

a)

According to quantization theory, the operator $\alpha\,\mathrm{sgn}(\nu(n))$ has an effect equivalent to that of the two-level quantizer:

$$Q(\nu(n)) = \begin{cases} \Delta/2 & \nu(n) \geq 0 \\ -\Delta/2 & \nu(n) < 0 \end{cases}$$

where $\nu(n) = e(n) + \alpha\varepsilon(n)$

Furthermore, since the sample of the dither $\varepsilon(n)$ are i.i.d over the interval $[-1, 1]$, $\{\alpha\varepsilon(n)\}$ satisfies the requirement for a valid dither process if the constant α satisfies the requirement:

$$\alpha \geq |e(n)|$$

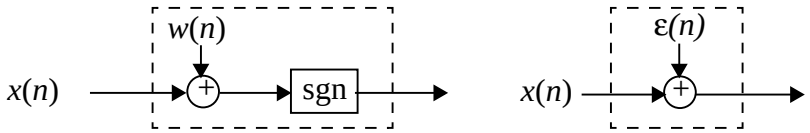


Figure 1

The equivalent model is illustrated in Figure 1. Hence, we may rewrite the DSE-CMA update formula:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\mathbf{u}(n)\,(e(n) + \varepsilon(n)) \tag{1}$$

Also since $\varepsilon(n)$ is an uncorrelated random process, its first moment is defined by

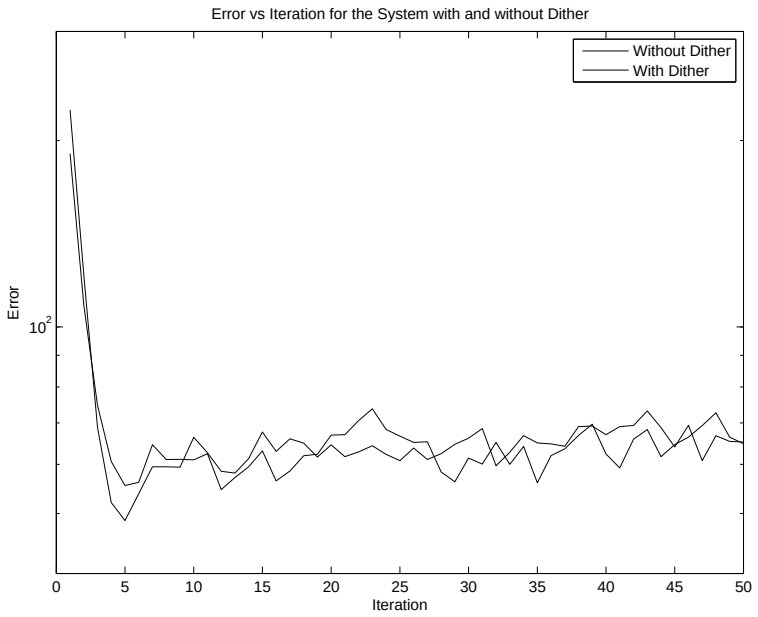
$$\mathbb{E}\,[\varepsilon(n)|e(n)] = \mathbb{E}\,[\varepsilon(n)] = 0 \tag{2}$$

Taking the expectation of the DSE-CMA error function, we find that it is a hard-limited version of the $e(n)$, as shown by

$$\mathbb{E}\,[\nu(n)|y(n)] = \begin{cases} \alpha, & \nu(n) > 0 \\ e(n), & |\nu(n)| \leq \alpha \\ -\alpha, & \nu(n) < -\alpha \end{cases}$$

which follows from Equations (1) and (2).

b)



Problem 17.11

a)

The Shalvi-Weinstein equalizer is based on the cost function

$$J(n) = \mathbb{E}\left[|y(n)|^4\right] \text{ subject to the constraint } \mathbb{E}\left[|y(n)|^2\right] = \sigma_x^2$$

where $y(n)$ is the equalizer output and σ_x^2 is the variance of the original data sequence applied to the channel input. Applying the method of Lagrange multipliers, we may define a cost function for the Shalvi-Weinstein equalizer that incorporates the constraint as follows:

$$J_{SW}(n) = \mathbb{E}\left[|y(n)|^4\right] + \lambda\left(\mathbb{E}\left[|y(n)|^2\right] - \sigma_x^2\right) \tag{1}$$

where λ is the Lagrange multiplier.

From Equation (17.105) of the text, we find that the cost function of the Godard algorithm takes the following form for $p = 2$:

$$\begin{aligned} J_G =& \mathbb{E}\left[\left(|y(n)|^2 - R_2\right)^2\right] \\ =& \mathbb{E}\left[|y(n)|^4\right] - 2R_2\mathbb{E}\left[|y(n)|^2\right] + R_2^2 \end{aligned} \tag{2}$$

where R_2 is a positive real constant. Comparing Equations (1) and (2), we see that these two cost functions have the same mathematical form. Hence, we may infer that these two equalization algorithms share the same optimization criterion.

b)

For a more detailed discussion of the equivalence between the Godard and Shalvi-Weinstein algorithms, we may proceed as follows:

First, rewrite the tap-weight vector of the equalizer in polar form (i.e., a unit-norm vector times a radial scale factor), and then optimize the Godard cost function with respect to the radial factor. The "reduced" cost function that results from this transformation is then recognized as a monotonic transformation of the corresponding Shalvi-Weinstein cost function. Since the transformation relating these two criteria is monotonic, their stationary points and local/global minima coincide to within a radial factor. By taking this approach, the equivalence between the Godard and Shalvi-Weinstein equalizers seems to hold under general conditions (linear or nonlinear channels, *iid* or correlated data sequences applied to the channel input, Gaussian or non-Gaussian channel noise, ect).[1]

Problem 17.12

For the derivation of Equation (17.116), see the Appendix of the paper by Johnson et al[2]. Note, however, the CMA cost function for binary PSK given in Equation (63) of that paper is four times that of Equation (17.116)

[1] For further details on the issues raised herein, see P.A. Regalia, "On the equivalence between the Godard and Shalvi-Weinstein schemes of blind equalization", Signal Processing, vol. 73, pp. 185-190, 1999.

[2] C.R. Johnson, et al., "Blind equalization using the constant modulus criterion: A review", Proc. IEEE, vol. 86, pp. 1927-1950, October 1998.

Appendix:
Computer Experiment on Adaptive Equalization, involving the LMS, RLS, IDBD algorithm and the Autostep Method

The appendix of the solution manual is an additional computer experiment that was not included in the original textbook, but it is included here as a comparative simulation between multiple methods of adaptive filtering. It should be viewed as an insightful extension of the computer experiment in Section 6.8, however, this project is extended across multiple algorithms learning performance.

The experiment, report, and associated source codes were completed by Erkan Baser during his studies of a graduate course in Adaptive Fitler Theory at McMaster University, and they are being reproduced here with his permission. Most importantly, the material presented herein is a "verbatim" reproduction of Erkan Baser's Report.

One last comment from Simon Haykin is in order:

> "The Autostep Method, in its current form, is heuristic in its formulation. It distinguishes itself from the other adaptive filtering algorithms in bypassing the need for a prescribed step-size paramter. The challenge for a researcher (e.g., Ph.D. student) to develop the underlying mathematics of the Autostep method in a rigorous manner."

1. Project Specification

Unlike the digital time-invariant filters, the adaptive filters have time-variant coefficients in order to adjust itself to unknown or time varying environments. The objective of this project is to implement adaptive filter algorithms for equalization of a distortion produced by the communication channel. The Fig.-1 shows the block diagram of the entire system. The transmitter sends a Bernoulli sequence of $+1$ and -1 with the probability 0.5 i.e., $x(n)$ is a random variable having zero-mean with the unit variance. The channel introduces distortion according to its impulse response:

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

where W denotes the parameter controlling the amplitude of the distortion throughout the channel. In addition, at the output of the channel additive white Gaussian noise $v(n)$ with variance σ_v^2, i.e., $v \sim N(0,\sigma_v^2)$. is added to the signal.

The aim of the adaptive equalizer is to correct the distortion produced by the channel and additive white Gaussian noise. The adaptive equalizer can be considered as an $M-tap$ finite impulse response (FIR) filter. However, the tap-weights $\hat{w}_0(n),\hat{w}_1(n),...,\hat{w}_{M-1}(n)$ approximate the optimum Wiener filter solution. That is, the weights are adjusted by comparing the output signal of the adaptive equalizer with the time-delayed desired signal. To do this, the the mean square error (MSE) cost function is minimized. In this work, we implement the following three linear adaptive filtering algorithms to perform the adaptive equalization: ***i)*** least mean square (LMS), ***ii)*** incremental delta-bar-delta (IDBD). and ***iii)*** autostep.

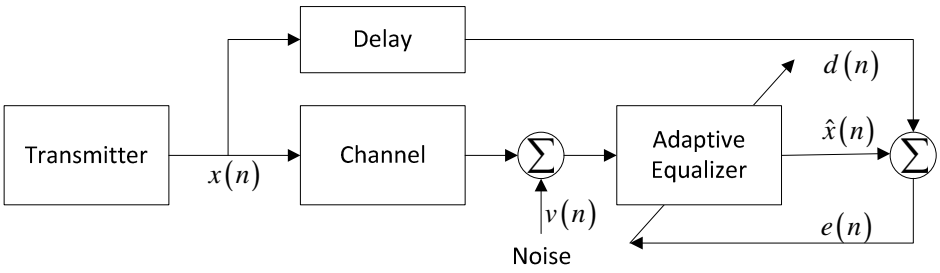


Fig.-1 Block diagram of the entire system.

2. Adaptive Equalizers

i) LMS Algorithm

As shown in Fig. 2, the LMS algorithm consists of two processes:

1. The transversal filter estimates the desired signal $\hat{x}(n)$ according to the current values of the tap-weights:

$$\hat{x}(n) = \sum_{i=0}^{M-1} \hat{w}_i(n)\, u(n-i). \tag{i.1}$$

Then, the estimation error $e(n)$ is computed by comparing the output signal to the desired response:

$$e(n) = d(n) - \hat{x}(n).$$

2. Automatic adjustment of the tap-weights are performed in accordance with the estimation error:

$$\underline{\hat{w}}(n+1) = \underline{\hat{w}}(n) + \mu \underline{u}(n) e^{*}(n), \tag{i.2}$$

where $*$ denotes the conjugate transpose of the error signal and μ is the stepsize parameter.

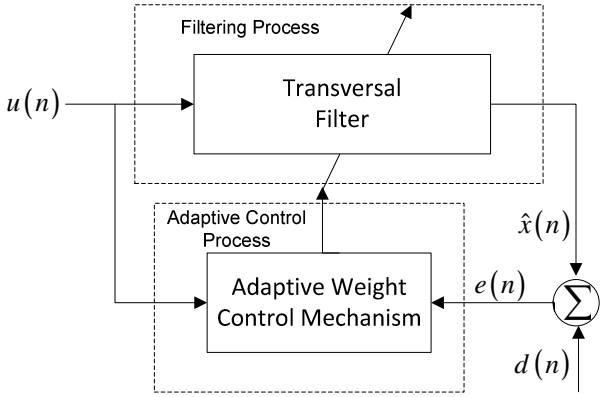


Fig.-2 Block diagram of the LMS filter.

The stepsize parameter μ is critical for the update of the tap-weights. That is, the LMS algorithm is convergent if the step size is selected according to [*S. Haykin, Adaptive Filter Theory, 5th edition, Prentice Hall, 2013*]

$$0 < \mu < \frac{2}{\lambda_{\max}}, \tag{i.3}$$

where $\lambda_{\max}$ denotes the largest eigenvalue of the correlation matrix of the input signal. However, the convergence rate become too slow if the stepsize is set to so small values. In typical applications, the computation of eigenvalues may not be possible. Therefore the max-stepsize is practically computed as

$$\mu_{\max} = \frac{1}{tr[R]},$$

where $tr[R]$ is the trace of the correlation matrix. For the adaptive equalization problem, the correlation matrix R of the input sequence $u(n),...,u(n-M+1)$ is symmetric and since the chancel coefficients are zero for $n>3$, its elements $r(\tau)=0$ for $\tau>2$. That is,

$$R=\begin{bmatrix} r(0) & r(1) & r(2) & 0 & \dots & 0 \\ r(1) & r(0) & r(1) & r(2) & \dots & 0 \\ r(2) & r(1) & r(0) & r(1) & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & r(0) \end{bmatrix},$$

where

$$\begin{aligned} r(0) &= \sum_{n=1}^{3} h(n)^2 + \sigma_v^2, \\ r(1) &= h(1)h(2) + h(2)h(3), \\ r(2) &= h(1)h(3). \end{aligned}$$

Thus, the maximum value for the stepsize parameter is practically computed as

$$\mu_{\max} = \frac{1}{tr[R]} = \frac{1}{M\sum_{n=1}^{3} h(n)^2 + \sigma_v^2}.$$

The convergence speed of the LMS algorithm hinges on the eigenvalue spread of R:

$$\chi(R) = \frac{\lambda_{\max}}{\lambda_{\min}},$$

where $\lambda_{\min}$ is the minimum eigenvalue of R. For example, in the 2D case, i.e., $M=2$ the plot of the MSE function is elliptic. Hence, the fastest convergence is observed along the shortest axis of the ellipse, while the convergence is the slowest along the longest axis.

Experiment-I: To see the effect of eigenvalue spread in the performance of the Adaptive equalization the LMS algorithm with the $\mu=0.075$ is invoked for $W=2.9,3.1,3.3,$ and 3.5. Table-1 shows the various parameters related to the adaptive equalization performed using the LMS algorithm against the channel distortion with $W=2.9,3.1,3.3,$ and 3.5.

Fig.-3 shows the learning curves of the LMS algorithm with $\mu=0.075$ averaged over 200 Monte Carlo runs for $W=2.9,3.1,3.3,$ and 3.5. Since the noise power has negligible effect on the distortion, compared to the channel distortion, it can be seen that the performance in the convergence of the LMS algorithm degrades as the amplitude of the distortion introduced by the channel increases. In other words, Table-2 indicates that the eigenvalue spread $\chi(R)$ increases as the channel distortion increases. Hence, the convergence rate slows down and the steady state value of the MSE error increases.

Parameters\Channel Distortion W	2.9	3.1	3.3	3.5
Number of Monte Carlo runs	200	200	200	200
Number of transmitted symbols	500	500	500	500
The noise power σ_v^2	0.001	0.001	0.001	0.001
Stepsize μ	0.075	0.075	0.075	0.075
Channel coefficients $h(1),h(2),h(3)$	0.2194 1.0000 0.2194	0.2798 1.0000 0.2798	0.3365 1.0000 0.3365	0.3887 1.0000 0.3887

Table-1 Parameters of the LMS Algorithm for each value of W .

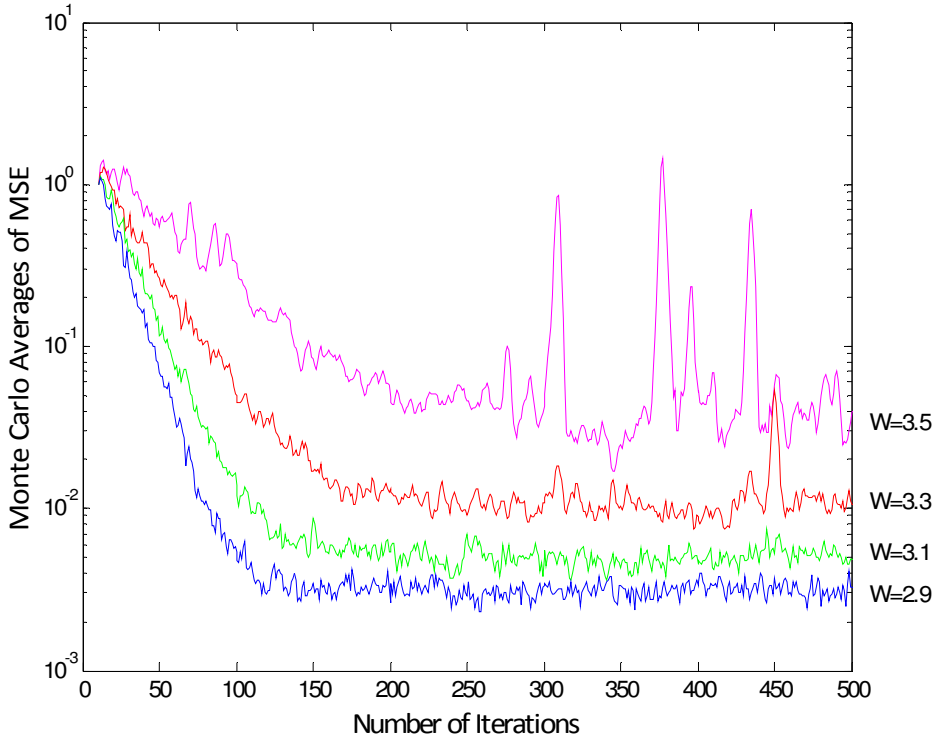


Fig.-3 Learning curves of the LMS algorithm with $\mu = 0.075$ for different level of channel distortions.

The Wiener filter provides the optimum solution, i.e., the minimum value of the MSE using the exact correlation R and covariance p matrices. However, the LMS algorithm uses the instantaneous estimates of these matrices. As expected, the instantaneous estimates have large variance, especially at the transient period of the adaptive equalizer and thus introduce gradient noise into the tap-weight estimates. Therefore, the LMS algorithm wanders around the minimum point of the MSE, i.e., it performs Brownian motion after a number of iterations.

Table-2 shows the minimum value of the MSE attained by the Wiener filter for $W = 2.9, 3.1, 3.3,$ and 3.5. In addition, the final value of the MSE for the LMS algorithm, i.e., J_∞ is computed by time averaging over the Monte Carlo runs. It is important note that the initial transient period is neglected to obtain the converged part of the LMS algorithms. Otherwise, we will end up with a too large value of the

MSE. As can be seen from the Table-2, the excess mean squared error (J_{ex}) increases as the channel distortion increases. The misadjustment factor, i.e., the ratio of $J_{ex}/J_{\min}$ indicates that the steady-state value of the MSE for the LMS algorithm divergences much more than the optimum MSE provided by the Wiener filter as W increases.

Results\Channel Distortion W	2.9	3.1	3.3	3.5
$\lambda_{\min}$	0.3339	0.2136	0.1255	0.0655
$\lambda_{\max}$	2.0295	2.3761	2.7263	3.0707
Eigenvalue Spread $\chi(R)$	6.0782	11.124	21.713	46.822
Final MSE J_{∞}	0.0032	0.0050	0.0114	0.0416
Wiener Filter's MSE $J_{\min}$	0.0014	0.0018	0.0025	0.0042
Excess MSE $J_{ex} = J_{\infty} - J_{\min}$	0.0018	0.0032	0.0089	0.0374
Misadjustment $J_{ex}/J_{\min}$	1.2857	1.7778	3.5600	8.9048

Table-2 Performance parameters of the LMS algorithm for the different levels of channel distortion.

Experiment-II: In this experiment the aim is to see the effect of the stepsize parameter on the performance of the LMS algorithm. With this aim, the LMS algorithm is invoked using different stepsize parameters $\mu = 0.0075, 0.075,$ and $\mu = 0.025$ for the same signal observed through the channel with $W = 3.1$. We know that the stepsize parameter affects the convergence rate. More explicitly, the convergence rate becomes too slow if the stepsize is set to small values. However, setting the stepsize to large values prevents the convergence of the LMS algorithm [*S. Haykin, Adaptive Filter Theory, 5th edition, Prentice Hall, 2013*]. The equation in (i.3) gives the range of the stepsize parameter. In addition, Table-2 provides the maximum eigenvalue of the correlation matrix for $W = 3.1$. Thus, the upper limit for the stepsize is computed as

$$\mu_{\max} = \frac{2}{\lambda_{\max}},$$
$$= 0.8417.$$

Therefore, the LMS algorithms for $\mu = 0.0075, 0.075,$ and $\mu = 0.025$ are convergent.

Results\Stepsize μ	0.0075	0.075	0.025
Final MSE J_{∞}	0.0033	0.0050	0.0022
Wiener Filter's MSE $J_{\min}$	0.0018	0.0018	0.0018
Excess MSE $J_{ex} = J_{\infty} - J_{\min}$	0.0015	0.0032	0.0004
Misadjustment $J_{ex}/J_{\min}$	0.8333	1.7778	0.2222
Convergence Rate (iteration No)	Not Converged	~200	~600

Table-3 Performance parameters of the LMS algorithm for the different stepsize parameter values.

Fig.-4 shows the learning curves of the LMS algorithm averaged over 200 Monte Carlo runs. It can be seen that the LMS algorithm with $\mu = 0.025,$ and 0.075 achieves their steady-state conditions. However, for $\mu = 0.0075$ the LMS algorithm needs more data to converge. In addition, a tradeoff arises between convergence rate and the misadjustment. Table-3 shows the misadjustments and approximate convergence rates of the LMS algorithms for the given stepsize parameter values. As can be seen in

Table-2, the faster convergence yields the more misadjustment. In addition, the LSM algorithm with $\mu = 0.0075$ achieves better performance in the misadjustment sense, compared to the LMS algorithm with $\mu = 0.075$ even though it did not converge to its steady-state condition. To deal with the adjustment of the stepsize parameter, the incremental Delta-Bar Delta algorithm (IDBD) was proposed.

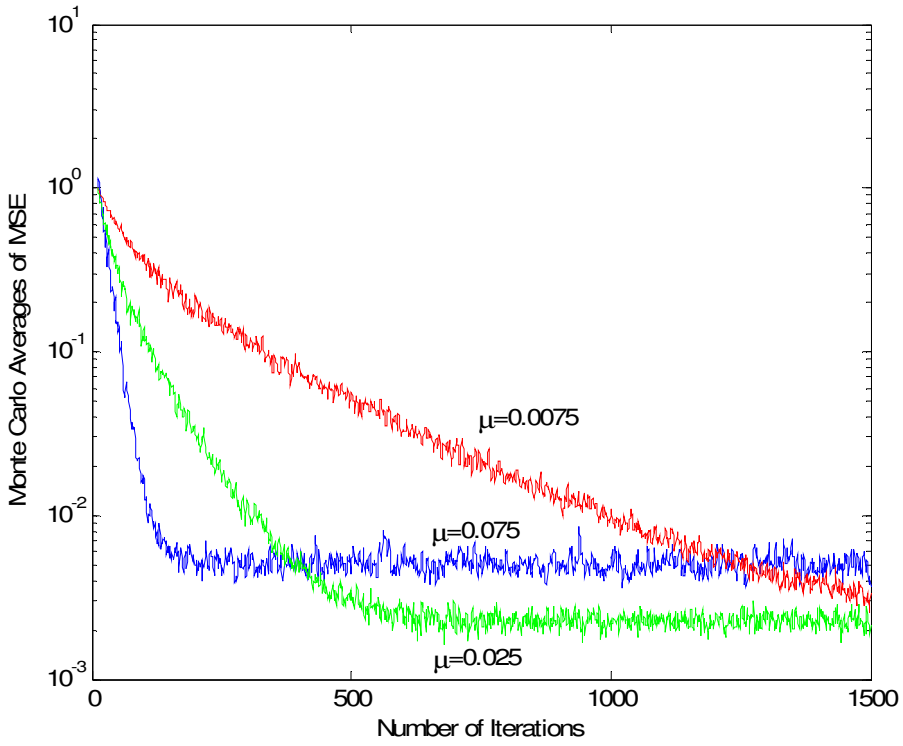


Fig.-4 Learning curves of the LMS algorithm for the different stepsize parameter values.

ii) IDBD Algorithm

The incremental Delta-Bar-Delta (IDBD) algorithm finds the optimal learning rate for a linear based learning system like the LMS learning rule given by (i.2). The aim of the LMS algorithm is to minimize the MSE. To do this, it updates the current tap-weights at each time-step using a fixed stepsize parameter. The stepsize parameter is to be prespecified according to (i.3). This parameter can be seen as the learning rate from each input. In the IDBD algorithm, the learning rate is adjusted for each input separately, i.e., $\mu_i(n)$ is time-varying for $i = 0,...,M-1$ to achieve faster convergence and smaller MSE, compared to the LMS algorithm.

The prespecified learning rate is the main reason of the steady state bias observed in the learning systems from the nonstationary signals. This is because that irrelevant inputs disturb the learning process with relevant inputs [*R. S. Sutton, "Adaptive Bias by Gradient Descent: An incremental version of delta-bar-delta," Proc. of 10th National Conf. of Artificial Intelligence, 1992*]. Therefore, for the inputs that are probably to be irrelevant the learning rate should be set to small values, whereas for inputs that are likely to be relevant the learning rate should be set to large values.

In the IDBD algorithm the learning rates (i.e., time-varying stepsize parameters) are given by

$$\mu_i(n) = e^{\beta_i(n)}, \text{ for } i = 0,...,M-1,$$

where $\beta_i(n)$ is the first memory parameter and is updated in a fashion like the LMS update-rule:

$$\beta_i(n+1) = \beta_i(n) + \kappa u(n-i) h_i(n) e(n)^*, \text{ for } i = 0,...,M-1,$$

where κ is a fixed constant known as the meta-learning rate and $h_i(n)$ is the second memory parameter updated according to

$$h_i(n+1) = h_i(n)\left[1-\mu_i(n+1)u(n-i)^2\right]^+ + \mu_i(n+1)u(n-i)e(n)^*, \quad \text{for } i = 0,...,M-1, \tag{ii.1}$$

where $[x]^+$ is x if $x > 0$, otherwise 0. The first term in (ii.1) is called the decay term while the second one increments $h_i(n)$ by the last weight change (i.e., $e(n)u(n-1)$). The intuition behind the IDBD algorithm can be summarized as follows: The change in the first memory parameter is proportional to the correlation between the current and a trace of past changes in the tap-weights. In other words, if the current change in the tap-weights is positively correlated with the past changes, this means that the learning-rates should have been set to larger values, otherwise, they were too large, i.e., the algorithm is wandering around the best tap-weights [*R. S. Sutton, "Adaptive Bias by Gradient Descent: An incremental version of delta-bar-delta," Proc. of 10th National Conf. of Artificial Intelligence, 199*2].

Fig.5 shows the block diagram of the IDBD filter. The first transversal filter performs the adaptive equalization with the previously updated learning rates (i.e., stepsizes) while the second transversal filter updates the learning rates for the next iteration using the current error and input.

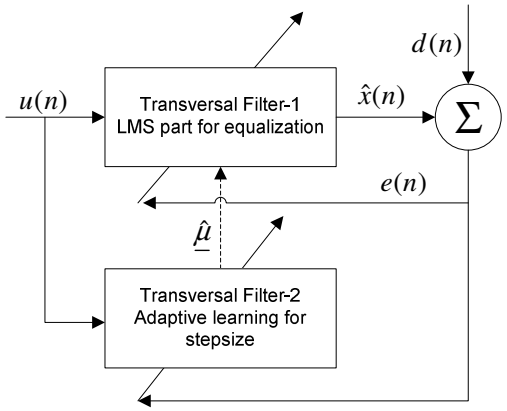


Fig.5 Block diagram of the IDBD filter.

Experiment-I: The aim of this experiment is to see the effect of eigenvalue spread in the performance of the IDBD algorithm. To do this, the IDBD algorithm with the $\kappa = 0.01$ is invoked for $W = 2.9, 3.1, 3.3,$ and 3.5. Fig.-6 shows the learning curves of the IDBD algorithms averaged over 200 Monte Carlo runs. It can be seen that the IDBD algorithm converges its steady-state condition for

$W = 2.9$, and 3.1. However, the number of data samples is not enough for the high channel distortions (i.e., $W = 3.3$, and 3.5) to observe the steady state conditions. In addition, the learning curves indicate that the MSE asymptotically decreases for all channel distortions. Consequently, Fig.-6 demonstrates that the IDBD outperforms the LMS algorithm.

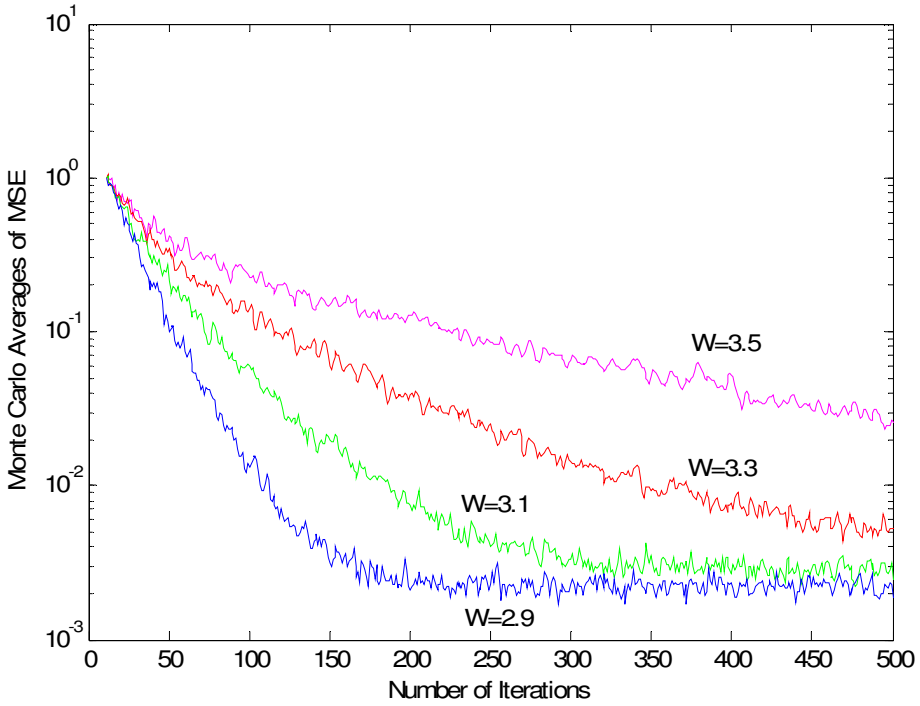


Fig.-6 Learning curves of the IDBD algorithm with $\kappa = 0.01$ for different level of channel distortions.

Results\Channel Distortion W	2.9	3.1	3.3	3.5
$\lambda_{\min}$	0.3339	0.2136	0.1255	0.0655
$\lambda_{\max}$	2.0295	2.3761	2.7263	3.0707
Eigenvalue Spread $\chi(R)$	6.0782	11.124	21.713	46.822
Final MSE J_{∞}	0.0022	0.0029	0.0053	0.0259
Wiener Filter's MSE $J_{\min}$	0.0014	0.0018	0.0025	0.0042
Excess MSE $J_{ex} = J_{\infty} - J_{\min}$	0.0008	0.0011	0.0028	0.0217
Misadjustment $J_{ex}/J_{\min}$	0.5714	0.6111	1.1200	5.1667

Table-4 Performance parameters of the IDBD algorithm for the different levels of channel distortion.

Table-4 shows the performance parameters of the IDBD algorithm. The same parameters were also computed for the LMS algorithm and were presented in Table-2. Note that J_{∞} is computed by time averaging over the Monte Carlo runs for the converged part of the IDBD algorithm by discarding the initial transient period. The comparison between the misadjustments in Table-2 and Table-4

corroborates the results about the performance of the IDBD algorithm, i.e., the IDBD algorithm outperforms the LMS algorithm by its small misadjustments for the same level of channel distortions.

Experiment-II: The aim of this experiment is to examine the effect of the meta-learning parameter (i.e., the free parameter in the IDBD algorithm) on the learning performance of the IDBD algorithm. To do this, the IDBD algorithm is invoked using different meta-learning rates $\kappa = 0.01, 0.10,$ and 0.30 for the same signal observed through the channel with $W = 3.1$.

Fig.-7 shows the learning curves of the IDBD algorithm averaged over 200 Monte Carlo runs. It can be seen that the IDBD algorithm's learning speed increases for the large values of the meta-learning rate. However, the IDBD algorithm pays the price for the fast learning by high misadjustment. This is the expected result, since the meta-learning rate κ can be thought of as a second stepsize parameter. Therefore, it has a range and for values outside this range the IDBD algorithm does not become convergent.

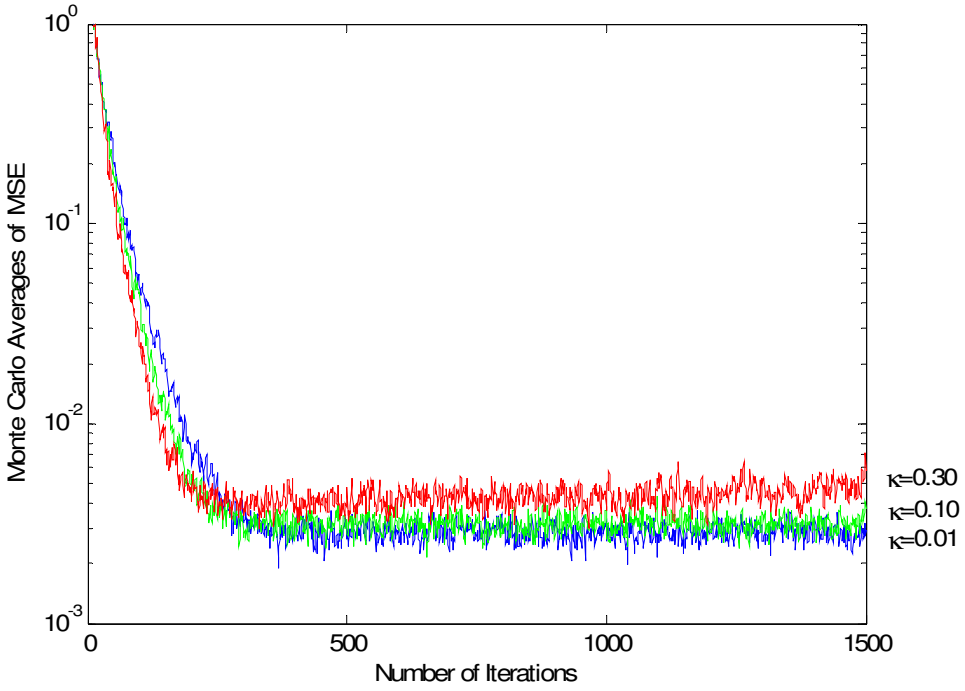


Fig.-7 Learning curves of the IDBD algorithm for the different meta-learning rate values.

Table-5 shows the misadjustments and approximate convergence rates in iteration number for the IDBD algorithms. It can be seen that the faster convergence yields the more misadjustment. In addition, the approximate convergence rate in iteration number corroborates the results inferred from Fig,-7 that IDBD learns faster with the larger κ but it suffer from the misadjustment. *The increase in the misadjustment observed in the IDBD algorithm with the higher values of the meta-learning rate κ can be expressed as the excessively gained confidence through the positive correlation between the current and the past changes as time proceeds.*

Results\Meta-learning κ	0.01	0.10	0.30
Final MSE J_{∞}	0.0029	0.0031	0.0041
Wiener Filter's MSE $J_{\min}$	0.0018	0.0018	0.0018
Excess MSE $J_{ex} = J_{\infty} - J_{\min}$	0.0011	0.0013	0.0023
Misadjustment $J_{ex}/J_{\min}$	0.6111	0.7222	1.2778
Convergence Rate (iteration No)	~340	~300	~230

Table-5 Performance parameters of the IDBD algorithm for the different meta-learning parameter values.

Experiment-III: The aim of the last experiment in this section is to examine the change in the learning rates (i.e., stepsize parameters) for different tap-weights in the IDBD algorithm as time proceeds. To do this, the IDBD algorithm is invoked using a small meta-learning parameter $\kappa = 0.01$ for the same signal observed through the channel with $W = 3.1$. The IDBD algorithm learns with learning. That is, it adjusts the tap-weights according to the relevance of the input. Thus, the irrelevant inputs make less contribution to the estimate, compared to relevant inputs associated with the high learning rates [*R. S. Sutton, "Adaptive Bias by Gradient Descent: An incremental version of delta-bar-delta," Proc. of 10th National Conf. of Artificial Intelligence, 1992*].

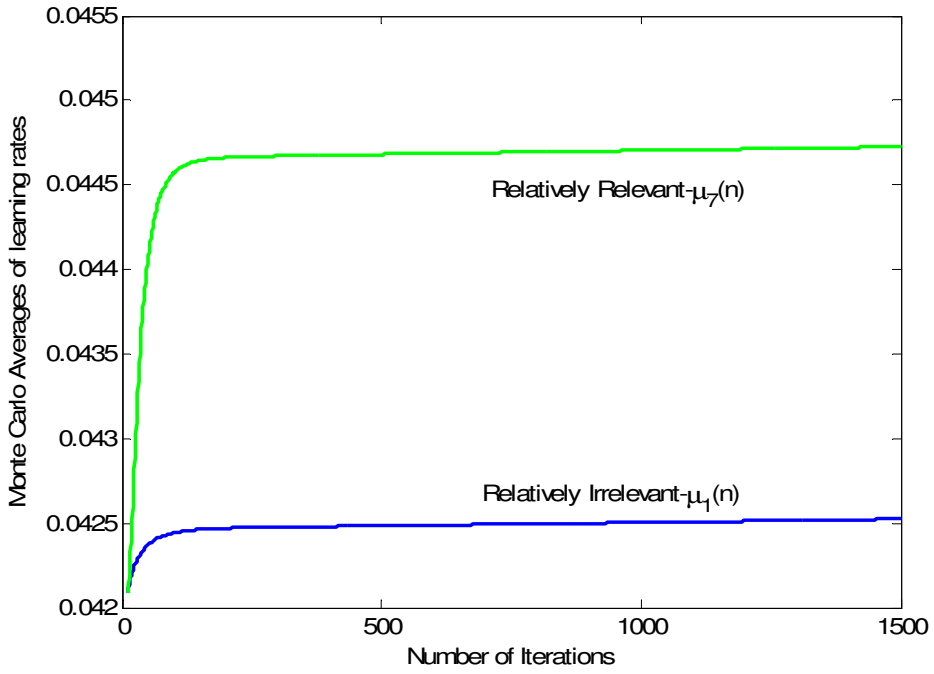


Fig.-8 Learning rates of the IDBD algorithm for relatively relevant and irrelevant inputs.

Fig.-8 shows the average behaviors of the two learning rates $\mu_1(n)$ and, $\mu_7(n)$ over 200 Monte Carlo runs. It can be seen that the learning rate $\mu_7(n)$ gets higher values to emphasize the contribution of the corresponding input to the estimate with higher associated tap-weight and after a while, it reaches its steady state value. However, the learning rate $\mu_1(n)$ gets relatively small values during its

transition period and is set to a smaller steady state value, compared to the value of $\mu_7(n)$. This means that inputs for the tap-weight $\hat{w}_7(n)$ is relatively more relevant than the inputs for the tap-weight $\hat{w}_1(n)$ in order to estimate the desired signal.

iii) AutoStep Method

The most obvious problem for the gradient descent type adaptive filters is the adjustment of the stepsize parameter. In the previous section, the IDBD algorithm was analyzed and the experiments demonstrated that its performance in terms of adaptation and stability hinges on the selection of the meta-learning rate (κ).Hence, the IDBD algorithm cannot be classified as tuning-free stepsize adaptive filtering utterly.

In the literature, experiments with huge data revealed three facts about the meta-learning rate κ: first , it has a unit, which is commensurable with orders of magnitude in different problems. Thus, the normalization of κ would make the IDBD algorithm less sensitive to the selection of κ. Second, κ is to be more flexible against abrupt changes observed in the first memory parameter $\beta(n)$ through $u(n-i)h_i(n)e(n)$. Third, the large stepsizes are detected and they are to be scaled down in order to prevent overshoots from the value attained by the Wiener filter. Considering all these facts about κ the following equations were developed respectively [*S. Haykin, Adaptive Filter Theory, 5th edition, Prentice Hall, 2013*]:

1. *Normalization:* The meta-learning parameter is normalized by taking running average of the product term $u(n-i)h_i(n)e(n)$, i.e.,

$$v_i(n+1) = v_i(n) + \gamma\mu_i(n)u_i(n)^2\left(\left|u(n-i)h_i(n)e(n)\right| - v_i(n)\right),$$

where γ is the forgetting factor. "The meta-learning rate κ has the same unit as the inverse of $u(n-i)h_i(n)e(n)$ thus the normalization with $v_i(n+1)$ makes the meta-learning rate free of units" [*A. Mahmood, M.Sc. Thesis, Dept. of Computer Science, University of Alberta, 2010*].

$$\beta_i(n+1) = \beta_i(n) + \kappa\frac{u(n-i)h_i(n)e(n)}{v_i(n+1)}$$

2. *Modification of the Normalizer:* As indicated above, the meta-learning rate is to be made more flexible against abrupt changes in $u(n-i)h_i(n)e(n)$. To do this, the normalizer is upper bounded as

$$v_i(n+1) = \max\left(\left|u(n-i)h_i(n)e(n)\right|, v_i(n) + \gamma\mu_i(n)u(n-i)^2\left(\left|u(n-i)h_i(n)e(n)\right| - v_i(n)\right)\right).$$

3. *High Step Detection:* To detect and prevent large step sizes the following two steps are carried out

$$B = \max\left(\sum_{i=0}^{M-1}\mu_i(n+1)u(n-i)^2, 1\right),$$

$$\mu_i(n+1) = \frac{\mu_i(n+1)}{B}.$$

Consequently, using these three equations together in the IDBD algorithm, the autostep filter was developed.

Experiment-I: The aim of this test is to see the effect of eigenvalue spread in the performance of the the autostep algorithm. To do this, the autostep algorithm with the optimum meta-learning rate $\kappa = 0.01$ is invoked for $W = 2.9, 3.1, 3.3,$ and 3.5. Fig.-9 shows the learning curves of the autostep algorithm averaged over 200 Monte Carlo runs.

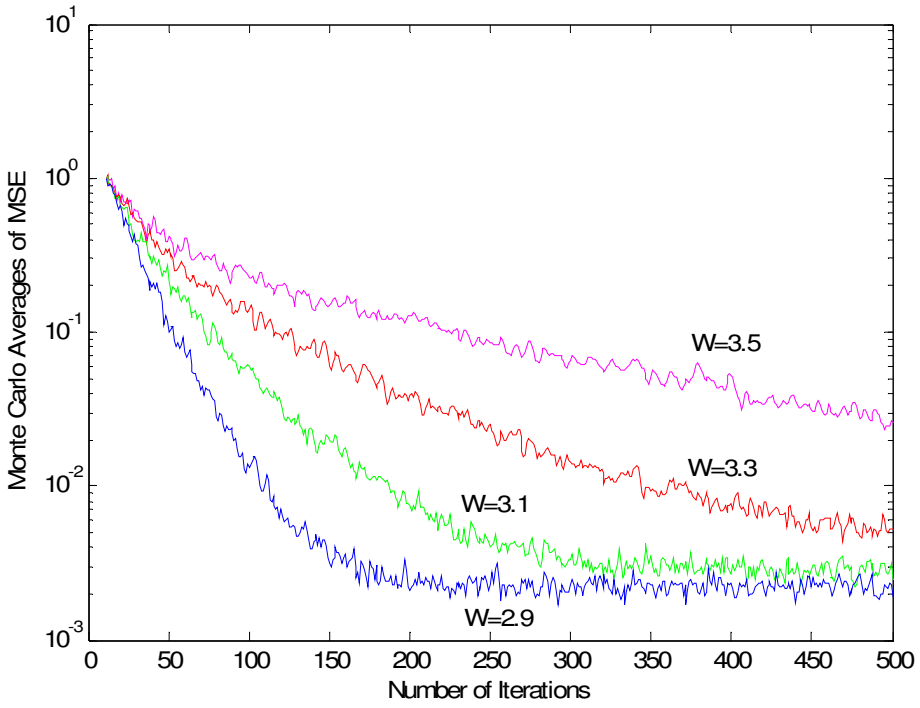


Fig.-9 Learning curves of the autostep algorithm with $\kappa = 0.01$ for different level of channel distortions.

As can be seen from Fig.-9, the performance of the autostep algorithm is quite similar to the IDBD algorithm. This is the expected result since the experiments for the IDBD algorithm demonstrated that the stepsize parameters are adjusted successfully if the meta-learning parameter is manually tuned to $\kappa = 0.01$. In addition, similar results were obtained for the Max-Normalized IDBD algorithm in [*A. Mahmood, M.Sc. Thesis, Dept. of Computer Science, University of Alberta, 2010*]: "There is a common range of meta-stepsize values around 10^{-2} that achieves near-best performance in all the problems that we have demonstrated."

Table-6 shows the performance parameters of the autostep algorithm. The same parameters were also computed for both the LMS algorithm and the IDBD algorithm. Note that J_{∞} is computed by time averaging over the Monte Carlo runs for the converged part of the autostep algorithm by discarding the initial transient period. The misadjustments for the IDBD algorithm in Table-4 and the autostep algorithm in Table-5 show the performance improvement in the autostep algorithm.

Results\Channel Distortion W	2.9	3.1	3.3	3.5
$\lambda_{\min}$	0.3339	0.2136	0.1255	0.0655
$\lambda_{\max}$	2.0295	2.3761	2.7263	3.0707
Eigenvalue Spread $\chi(R)$	6.0782	11.124	21.713	46.822
Final MSE J_{∞}	0.0022	0.0029	0.0053	0.0245
Wiener Filter's MSE $J_{\min}$	0.0014	0.0018	0.0025	0.0042
Excess MSE $J_{ex} = J_{\infty} - J_{\min}$	0.0008	0.0011	0.0028	0.0203
Misadjustment $J_{ex}/J_{\min}$	0.5714	0.6111	1.1200	4.8333

Table-6 Performance parameters of the autostep algorithm for the different levels of channel distortion.

Experiment-II: The aim of this experiment is to examine the effect of the meta-learning rate on the learning performance of the autostep algorithm. To do this, the autostep algorithm is invoked using different meta-learning rates $\kappa = 0.01, 0.10,$ and 0.30 for the same signal observed through the channel with $W = 3.1$.

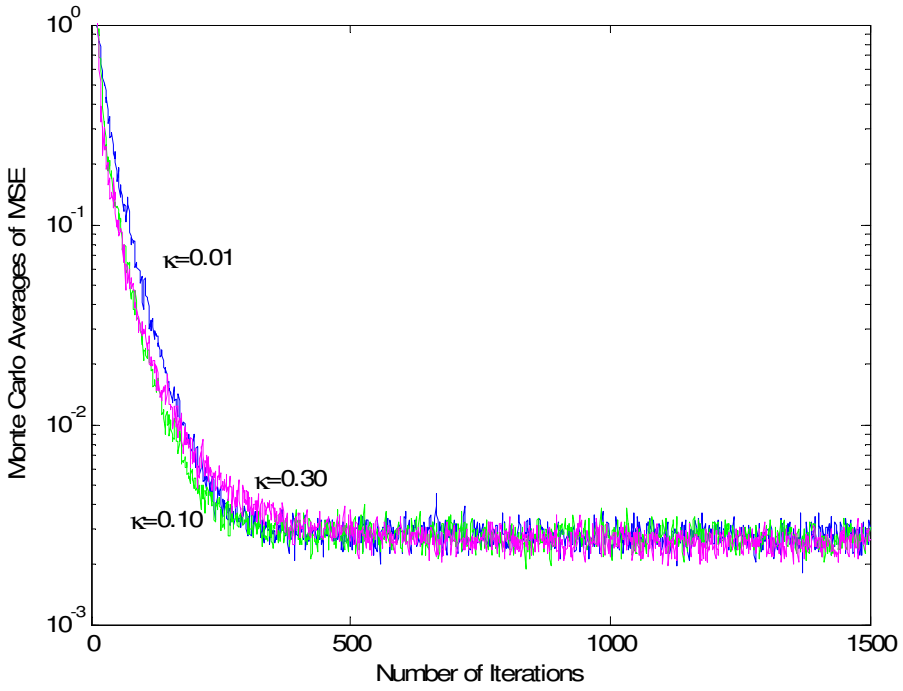


Fig.-10 Learning curves of the autostep algorithm for the different meta-learning parameter values.

Fig.-10 shows the learning curves of the autostep algorithm averaged over 200 Monte Carlo runs. It can be seen that the autostep algorithm's learning speed does not change for the large values of the meta-learning rate. However, Fig.-7 showed that the IDBD algorithm pays the price for the fast learning rates by high misadjustments. *This is the expected result, since the autostep algorithm is more robust*

than the IDBD algorithm. More explicitly, the improvements (i.e., normalization, modification, and high step detection) makes the autostep algorithm to less sensitive to the selection of meta-learning rate κ.

Results\Meta-learning κ	0.01	0.10	0.30
Final MSE J_{∞}	0.0029	0.0029	0.0029
Wiener Filter's MSE $J_{\min}$	0.0018	0.0018	0.0018
Excess MSE $J_{ex} = J_{\infty} - J_{\min}$	0.0011	0.0011	0.0011
Misadjustment $J_{ex}/J_{\min}$	0.6111	0.6111	0.6111
Convergence Rate (iteration No)	~340	~340	~340

Table-7 Performance parameters of the autostep algorithm for the different meta-learning rate values.

Table-7 shows the misadjustments and approximate convergence rates in iteration number for the autostep algorithm with the different the meta-learning rate values. *In contrast to the IDBD algorithm, it can be seen that the autostep algorithm does not suffer from the misadjustment if the meta-learning parameter* κ *is not manually tuned in advance*. This experiment demonstrates that the autostep algorithm is a tuning-free adaptive filter. In addition, observe that the approximate convergence rates in iteration number are same for three different meta-learning values.

Experiment-III: The aim of the last experiment in this section is to examine the change in the learning rates (i.e., stepsize parameters) for different tap-weights in the autostep algorithm as time proceeds. To do this, the autostep algorithm is invoked using a small meta-learning rate $\kappa = 0.01$ for the same signal observed through the channel with $W = 3.1$.

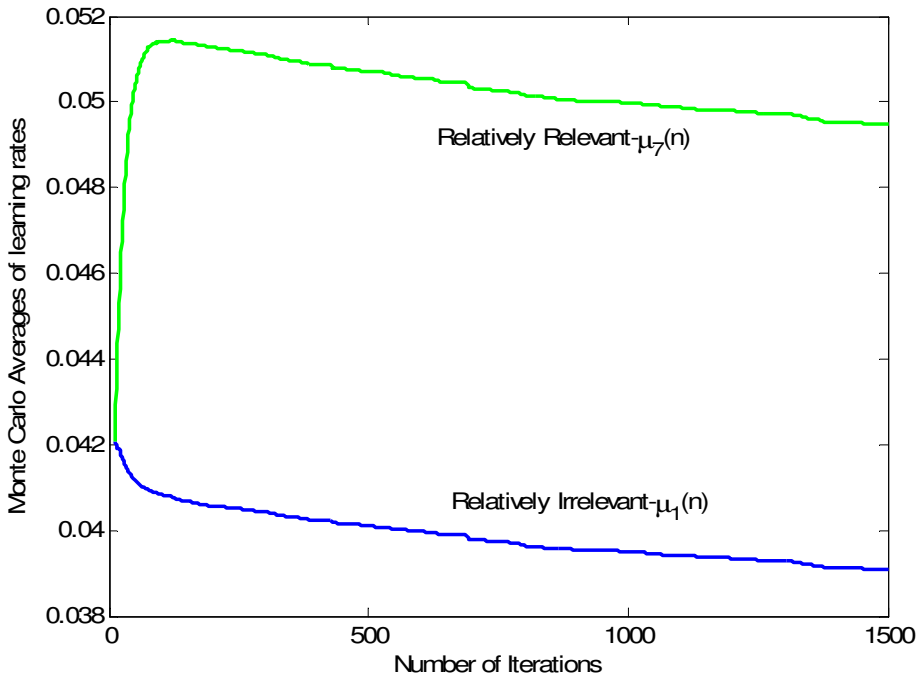


Fig.-11 Learning rates of the autostep algorithm for relatively relevant and irrelevant inputs.

Fig.-11 shows the average behaviors of the two learning rates $\mu_1(n)$ and, $\mu_7(n)$ over 200 Monte Carlo runs. Similar to the IDBD algorithm, the learning rate $\mu_7(n)$ gets higher values to emphasize the contribution of the corresponding input to the estimate via the associated tap-weight. In contrast to the IDBD algorithm, it does not reach a steady state value but decreases after a point. This can be explained by the high step detection property of the autostep algorithm. In contrast to $\mu_7(n)$, the learning rate $\mu_1(n)$ gets relatively small values during its transition period and its value decreases as time proceeds. This means that inputs for the tap-weight $\hat{w}_7(n)$ is relatively more relevant than the inputs for the tap-weight $\hat{w}_1(n)$ in order to estimate the desired signal. In addition, the high step detection property always force the autostep algorithm to decrease the divergence from the optimal solution after the convergence, i.e., when Brownian motion starts

Matlab Codes

```
function mylms(Ns,W,I)
clc, close all;
% W  - Channel Distortion parameter
% Ns - Number of transmitted symbol
% I  - Number of iterations
%% Internal parameters
% Delay factor
D = 7;
% Noise parameters v~N(0,w)
var_v = 1e-3;
mean_v  = 0;
% The number of filter taps
Nt = 11;
% MSE for adaptive filter.
MSE=zeros(1,Ns);
%% Channel
% impulse response h(n) for n=1,2,3
n = 1:3;
disp('Channel Coefficients')
h= 0.5*(1+cos(2*pi/W*(n-2))),
%% Eigenvalue Spread
r(1) = sum(h.^2)+var_v;
r(2) = h(1)*h(2)+h(2)*h(3);
r(3) = h(1)*h(3);
R = CorrM(Nt,3,r);
lambda = eig(R);
eigS = max(lambda)/min(lambda);
disp(['Max Eigenvalue:',num2str(max(lambda),5)]);
disp(['Min Eigenvalue:',num2str(min(lambda),5)]);
disp(['Eigenvalue Spread:',num2str(eigS,5)]);
%% Equalizer
% The stepsize parameter mu
max_mu=1/(Nt*sum(h.^2)+var_v);
% 0<mu<2/max_mu to achieve convergence
mu = 0.0075;
%% Adaptive Equalizer using LMS algorithm
```

```
for mc=1:I

    % Transmitted sequence
    x = 2*round(rand(1,Ns))-1;
    % Delay the transmitted sequence
    x_ref = [zeros(1,D),x];

    %disp(['Iteration:',num2str(i)]);
    % Channel AWGN
    v = mean_v+sqrt(var_v)*randn(1,Ns);
    % Channel distorted signal
    u = filter(h,1,x)+v;
    % Initialize  tap-weights
    w = zeros(1,Nt);

    for j=Nt:Ns
        % Mx1 input tap vector
        uvec = u(j:-1:j-Nt+1);

        % e = d_delayed - d_est
        e(j)= x_ref(j)-w*uvec';

        % Update the weights
        w = w + mu*e(j)*uvec;
    end
    % Mean Square Error(MSE).
    MSE = MSE + e.^2;
end
MSE = MSE/I;
figure,semilogy(Nt:Ns,MSE(Nt:Ns));
xlabel('Number of Iterations');
ylabel('Monte Carlo Averages of MSE');

function [R] = CorrM(M,n,r)
R = zeros(M,M);
for i=1:M
    for j=1:M
        if abs(i-j)<=n-1
            k = abs(i-j)+1;
            R(i,j)=r(k);
        end
    end
end
```

```
function myIDBD(Ns,W,I,K)
clc, close all
% W  - Channel Distortion parameter
% Ns - Number of transmitted symbol
% I  - Number of iterations
%% Internal parameters
% Delay factor
D = 7;
% Noise parameters v~N(0,w)
var_v = 1e-3;
```

```
mean_v  = 0;
% The number of filter taps
Nt = 11;
% MSE for adaptive filter.
MSE=zeros(1,Ns);
% The history of two stepsizes
s2 = zeros(2,Ns);
%% Channel
% impulse response h(n) for n=1,2,3
n = 1:3;
disp('Channel Coefficients')
h= 0.5*(1+cos(2*pi/W*(n-2))),
%% Eigenvalue Spread
r(1) = sum(h.^2)+var_v;
r(2) = h(1)*h(2)+h(2)*h(3);
r(3) = h(1)*h(3);
R = CorrM(Nt,3,r);
lambda = eig(R);
eigS = max(lambda)/min(lambda);
disp(['Max Eigenvalue:',num2str(max(lambda),5)]);
disp(['Min Eigenvalue:',num2str(min(lambda),5)]);
disp(['Eigenvalue Spread:',num2str(eigS,5)]);
%% Adaptive Learning for stepsize
% Meta-learning parameter
if nargin == 3
    K = 1e-2;
end
%% Adaptive Equalizer using IDBD algorithm
for mc=1:I

    % Adaptive stepsize parameter
    mu = (0.1/max(lambda))*ones(1,Nt);
    % Initialize  tap-weights
    w = zeros(1,Nt);

    % Primary memory parameter
    B = log(mu).*ones(1,Nt);
    % Secondary memory parameter
    F = zeros(1,Nt);

     % Transmitted sequence
    x = 2*round(rand(1,Ns))-1;
    % Delay the transmitted sequence
    x_ref = [zeros(1,D),x];

    % Channel AWGN
    v = mean_v+sqrt(var_v)*randn(1,Ns);
    % Channel distorted signal
    u = filter(h,1,x)+v;

    for j=Nt:Ns
        % Mx1 input tap vector
        uvec = u(j:-1:j-Nt+1);

        % e = d_delayed - d_est
        e(j)= x_ref(j)-w*uvec';
```

```
        % Update the weights through adaptive stepsizes
        mu = exp(B);
        w = w + mu.*uvec*e(j);

        %Select 3 tap-weights
        s2(1,j)= s2(1,j)+mu(1);
        s2(2,j)= s2(2,j)+mu(7);

        % Learn the stepsize
        d = 1-mu.*(uvec.^2); % decay term
        for i=1:Nt
            if d(i) < 0
                d(i) = 0;
            end
        end
        F = F.*d+mu.*uvec*e(j);
        B = B + K*uvec.*F*e(j);
    end
    % Mean Square Error(MSE).
    MSE = MSE + e.^2;
end
MSE = MSE/I;
figure,semilogy(200:Ns,MSE(200:Ns));
xlabel('Number of Iterations');
ylabel('Monte Carlo Averages of MSE');

s2 = s2 /I;
figure,plot(200:Ns,s2(1,200:Ns),'b');
hold on, plot(200:Ns,s2(2,200:Ns),'g');
xlabel('Number of Iterations');
ylabel('Monte Carlo Averages of learning rates');

function [R] = CorrM(M,n,r)
R = zeros(M,M);
for i=1:M
    for j=1:M
        if abs(i-j)<=n-1
            k = abs(i-j)+1;
            R(i,j)=r(k);
        end
    end
end
```

```
function myAutoStep(Ns,W,I,K)
clc, close all
% W  - Channel Distortion parameter
% Ns - Number of transmitted symbol
% I  - Number of iterations
%% Internal parameters
% Delay factor
D = 7;
% Noise parameters v~N(0,w)
var_v = 1e-3;
```

```
mean_v  = 0;
% The number of filter taps
Nt = 11;
% MSE for adaptive filter.
MSE=zeros(1,Ns);
% The history of two stepsizes
s2 = zeros(2,Ns);
% The forgetting factor
G = 1e-4;
%% Channel
% impulse response h(n) for n=1,2,3
n = 1:3;
disp('Channel Coefficients')
h = 0.5*(1+cos(2*pi/W*(n-2))),
%% Eigenvalue Spread
r(1) = sum(h.^2)+var_v;
r(2) = h(1)*h(2)+h(2)*h(3);
r(3) = h(1)*h(3);
R = CorrM(Nt,3,r);
lambda = eig(R);
eigS = max(lambda)/min(lambda);
disp(['Max Eigenvalue:',num2str(max(lambda),5)]);
disp(['Min Eigenvalue:',num2str(min(lambda),5)]);
disp(['Eigenvalue Spread:',num2str(eigS,5)]);
%% Adaptive Learning for stepsize
% Meta-learning parameter
if nargin == 3
    K = 1e-2;
end
%% Adaptive Equalizer using autostep method
for mc=1:I

    % Adaptive stepsize parameter
    mu = (0.1/max(lambda))*ones(1,Nt);
    % Initialize  tap-weights
    w = zeros(1,Nt);

    % Primary memory parameter
    B = zeros(1,Nt);
    % Secondary memory parameter
    F = zeros(1,Nt);

    % Normalizer
    Nr = zeros(1,Nt);

    % Transmitted sequence
    x = 2*round(rand(1,Ns))-1;
    % Delay the transmitted sequence
    x_ref = [zeros(1,D),x];

    % AWGN for channel
    v = mean_v+sqrt(var_v)*randn(1,Ns);
    % Channel distorted signal
    u = filter(h,1,x)+v;

    for j=Nt:Ns
```

```
        % Mx1 input tap vector, u(n)
        uvec = u(j:-1:j-Nt+1);

        % e = d_delayed - d_est
        e(j)= x_ref(j)-w*uvec';
                for i=1:Nt
            temp = e(j)*uvec(i).*F(i);

            % Modification of Normalizer
            Nr(i) = max(abs(temp),....
                        Nr(i)+ G*(mu(i)*uvec(i)^2)*(abs(temp)-Nr(i)));

            % Normalize K
            if Nr(i) == 0
                B(i) = log(mu(i));
            else
                B(i) = log(mu(i))+K*temp/Nr(i);
            end

            % Update the stepsizes
            mu(i) = exp(B(i));
        end

        % High stepsize detection and protection
        S = mu*(uvec.^2)';
        mu = mu./max(S,1);

        % Update weights
        w = w + mu.*uvec*e(j);

        % Select 2 tap-weights to monitor
        s2(1,j)= s2(1,j)+mu(1);
        s2(2,j)= s2(2,j)+mu(7);

        % Update the second memory parameter
        d = 1-mu.*(uvec.^2); % decay term
        F = F.*d+mu.*uvec*e(j);
    end
    % Mean Square Error(MSE).
    MSE = MSE + e.^2;
end
MSE = MSE/I;
figure,semilogy(Nt:Ns,MSE(Nt:Ns));
xlabel('Number of Iterations');
ylabel('Monte Carlo Averages of MSE');

s2 = s2 /I;
figure,plot(Nt:Ns,s2(1,Nt:Ns),'b');
hold on, plot(Nt:Ns,s2(2,Nt:Ns),'g');
xlabel('Number of Iterations');
ylabel('Monte Carlo Averages of learning rates');

function [R] = CorrM(M,n,r)
R = zeros(M,M);
for i=1:M
```

```
    for j=1:M
        if abs(i-j)<=n-1
            k = abs(i-j)+1;
            R(i,j)=r(k);
        end
    end
end
```