

Solutions Manual
for
Statistical and Adaptive Signal Processing

This page is intentionally left blank.

Preface

This solutions manual provides solutions to the problems contained in the first edition of our book STATISTICAL AND ADAPTIVE SIGNAL PROCESSING. The solutions have been prepared and written by David Marden and ourselves. We have attempted to provide very detailed solutions to the problems with notation consistent with that used in the book. Where applicable, we have also given a printout of Matlab code and figures for the problem solution.

Despite our efforts, however, you may find that some of the solutions may be less detailed and refined than others. Inevitably through the use of this solutions manual, omissions and errors will be discovered. Our goal is to continually improve upon this manual using your comments. Periodically, we will issue changes to the solutions manual and new problems to you upon request. In this respect, we would appreciate any feedback including improved solutions, suggestions for new problems, corrections which can be sent to Prof. Vinay K. Ingle at vingle@ece.neu.edu or at

Prof. Vinay K. Ingle
Department of Electrical and Computer Engineering
Northeastern University
360 Huntington Avenue
Boston, MA 02115

Thank you.

*Dimitris G. Manolakis
Vinay K. Ingle
Stephen K. Kogon*

This page is intentionally left blank.

Chapter 2
Discrete-Time Signals and Systems

2.1 Sampling frequency $F_s = 100$ sam/sec

- (a) Continuous-time signal $x_c(t) = 2 \cos(40\pi t + \pi/3)$ has frequency of 20 Hz. Hence

$$x(n) = x_c(t)|_{t=n/F_s} = 2 \cos\left(\frac{40\pi n}{100} + \pi/3\right)$$

which implies that $\omega_0 = \frac{40\pi}{100} = \frac{2\pi}{5}$.

- (b) Steady-state response $y_{c,ss}(t)$: Given that $h(n) = 0.8^n u(n)$, the frequency response function is

$$H(e^{j\omega}) = \frac{1}{1 - 0.8e^{-j\omega}}$$

Since $\omega_0 = 2\pi/5$, the system response at ω_0 is

$$H(e^{j\omega_0}) = \frac{1}{1 - 0.8e^{-j2\pi/5}} = 0.9343 e^{-j0.2517\pi}$$

Hence $y_{ss}(n) = 2(0.9343) \cos(2\pi n/5 + \pi/3 - 0.2517\pi)$, or

$$y_{c,ss}(t) = 1.8686 \cos(40\pi t + 0.585\pi)$$

- (c) Any $x_c(t)$ that has the same digital frequency ω_0 after sampling and the same phase shift as above will have the same steady state response. Since $F_s = 100$ sam/sec, the two other frequencies are 120 and 220 Hz.

2.2 The discrete-time signal is

$$x(n) = A \cos(\omega_0 n) w_R(n)$$

where $w_R(n)$ is an N -point rectangular window.

- (a) The DTFT of $x(n)$ is determined as

$$\begin{aligned} X(e^{j\omega}) &= \mathcal{F}[A \cos(\omega_0 n) w_R(n)] \\ &= (A/2) \mathcal{F}[e^{j\omega_0} w_R(n)] + (A/2) \mathcal{F}[e^{-j\omega_0} w_R(n)] \end{aligned} \quad (1)$$

Using the DTFT of $w_R(n)$ as

$$\mathcal{F}[w_R(n)] = \sum_{n=0}^{N-1} e^{-j\omega n} = e^{-j\omega(N-1)/2} \frac{\sin(\omega N/2)}{\sin(\omega/2)} \quad (2)$$

and the fact that complex exponential causes a translation in the frequency domain (1) can be written after a fair amount of algebra and trigonometry as

$$X(e^{j\omega}) = X_R(e^{j\omega}) + j X_I(e^{j\omega})$$

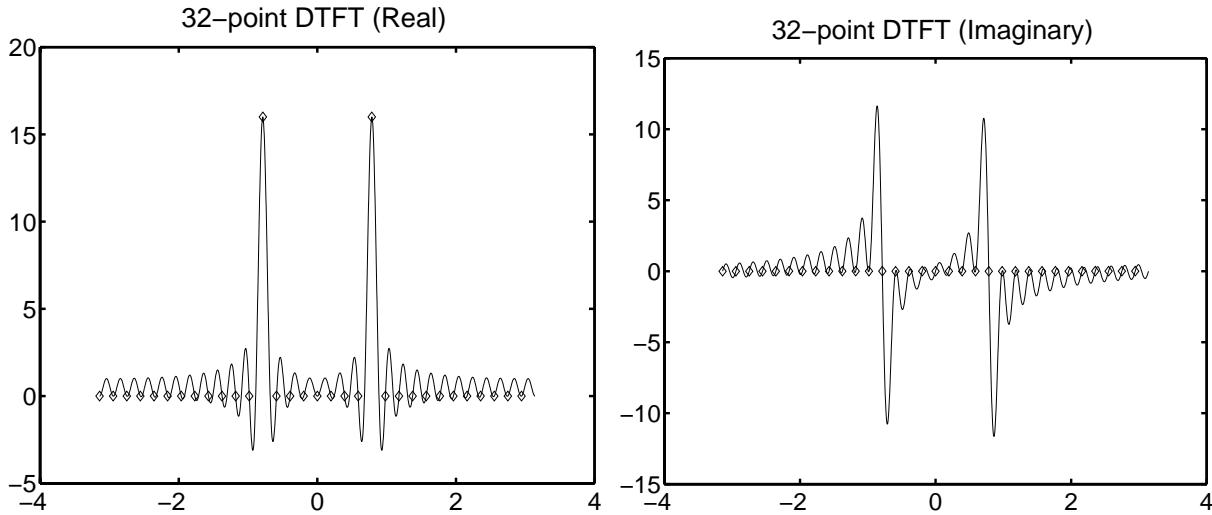


Figure 2.2bc: Real and Imaginary DTFT and DFT Plots($\omega_0 = \pi/4$)

where

$$\begin{aligned} X_R(e^{j\omega}) &= \frac{A}{2} \cos[(\omega - \omega_0)(N - 1)/2] \frac{\sin[(\omega - \omega_0)N/2]}{\sin[(\omega - \omega_0)/2]} \\ &\quad + \frac{A}{2} \cos[(\omega + \omega_0)(N - 1)/2] \frac{\sin\{[\omega - (2\pi - \omega_0)]N/2\}}{\sin\{[\omega - (2\pi - \omega_0)]/2\}} \end{aligned} \quad (3)$$

and

$$\begin{aligned} X_R(e^{j\omega}) &= -\frac{A}{2} \sin[(\omega - \omega_0)(N - 1)/2] \frac{\sin[(\omega - \omega_0)N/2]}{\sin[(\omega - \omega_0)/2]} \\ &\quad - \frac{A}{2} \sin[(\omega + \omega_0)(N - 1)/2] \frac{\sin\{[\omega - (2\pi - \omega_0)]N/2\}}{\sin\{[\omega - (2\pi - \omega_0)]/2\}} \end{aligned} \quad (4)$$

(b) $N = 32$ and $\omega_0 = \pi/4$. The DTFT plots are shown in Figure 2.2bc.

(c) The DFT samples are shown in Figure 2.2bc.

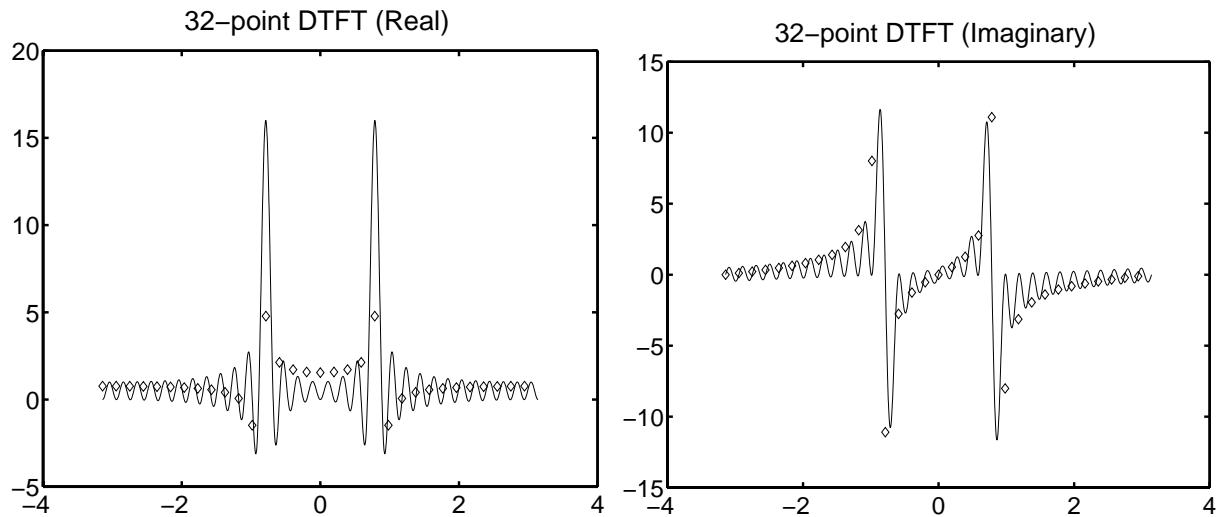
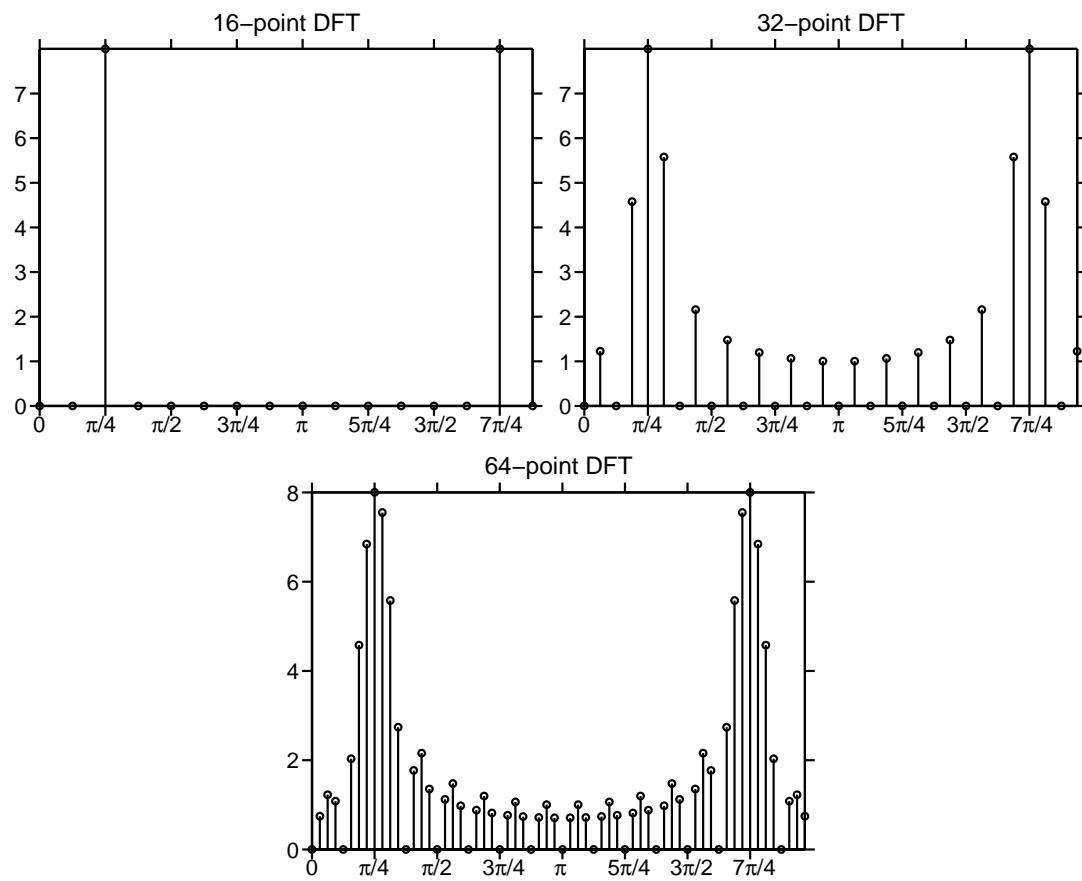
(d) $N = 32$ and $\omega_0 = 1.1\pi/4$. The plots are shown in Figure 2.2d.

The added spectrum for the second case above ($\omega_0 = 1.1\pi/4$) is a result of the periodic extension of the DFT. For a 32-point sequence, the end of each extension does not line up with the beginning of the next extension. This results in sharp edges in the periodic extension, and added frequencies in the spectrum.

2.3 The sequence is $x(n) = \cos(\pi n/4)$, $0 \leq n \leq 15$.

- (a) The 16-point DFT is shown in the top-left plot of Figure 2.3.
- (b) The 32-point DFT is shown in the top-right plot of Figure 2.3.
- (c) The 64-point DFT is shown in the bottom plot of Figure 2.3.
- (d) The zero padding results in a lower frequency sampling interval. Hence there are more terms in the DFT representation. The shape of the DTFT continues to fill in as N increases from 16 to 64.

2.4 $x(n) = \{1, 2, 3, 4, 3, 2, 1\}$; $h(n) = \{-1, 0, 1\}$

**Figure 2.2d:** Real and Imaginary DTFT Plots ($\omega_0 = 1.1\pi/4$)**Figure 2.3:** The 16, 32, and 64-point DFTs of $x(n) = \cos(\pi n/4)$

(a) Convolution using matrix-vector multiplication approach.

$$\begin{aligned} y &= Xh \\ \begin{bmatrix} -1 \\ -2 \\ -2 \\ -2 \\ 0 \\ 2 \\ 2 \\ 2 \\ 1 \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 2 & 1 \\ 4 & 3 & 2 \\ 3 & 4 & 3 \\ 2 & 3 & 4 \\ 1 & 2 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \end{aligned}$$

(b) The Matlab function `convtoep`

```
function [y]=convtoep(x,h)
% Convolution using Toeplitz Arrays
% y = convtoep(x,h)

nx = length(x); x = reshape(x,nx,1);
nh = length(h); h = reshape(h,nh,1);
X = toeplitz([x; zeros(nh-1,1)],[x(1) zeros(1,nh-1)]);
y = X*h;
```

(c) Verification:

2.5 $x(n) = 0.9^n u(n)$

(a) Analytical evaluation of $x(n) * x(n)$:

$$\begin{aligned} y(n) &= x(n) * x(n) = \sum_{k=-\infty}^{\infty} x(k)x(n-k) \\ &= \sum_{k=-\infty}^{\infty} (0.9)^k u(k)(0.9)^{n-k} u(n-k) \\ &= \sum_{k=0}^{\infty} (0.9)^k (0.9)^{n-k} u(n-k) \\ y(n) &= (n+1)(0.9)^n \end{aligned}$$

This sequence is shown in the leftmost plot in Figure 2.5.

(b) Convolution using the `conv` function: The sequence $x(n)$ is truncated to 51 samples. This convolution is done using

```
n = 0:50; x =(0.9).^n; y = conv(x,x);
```

This sequence is in the center plot in in Figure 2.5.

(c) Convolution using the `filter` function: To use this function, we have to represent one of the $x(n)$ by coefficients in an equivalent difference equation. This difference equation is given by

$$x(n) = \delta(n) + 0.9x(n-1)$$

which means that the filter coefficients are $b = 1$, $a = [1, -0.9]$. Thus this convolution is done using

```
y = filter(1,[1,-0.9],x);
```

This sequence is in the rightmost plot in Figure 2.5.

- (d) The three plots appear to be identical. However, the `conv` function gives the largest error since both sequences are truncated. The `filter` function would be best suited for infinite length sequences.

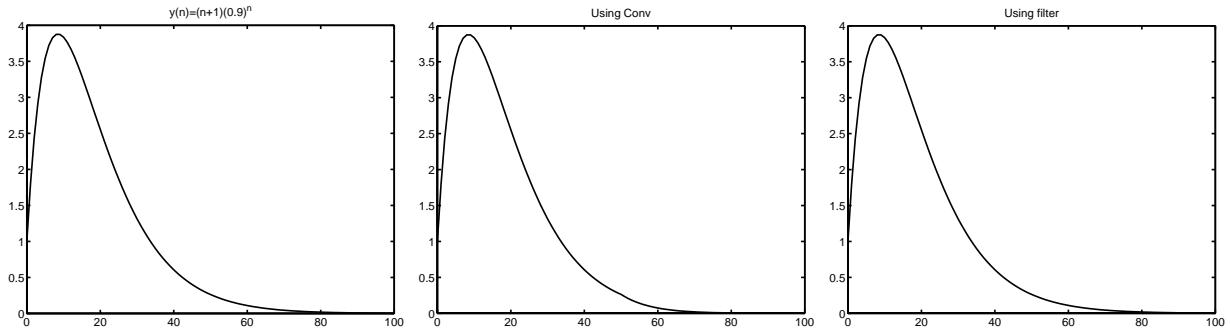


Figure 2.5: $x(n) = (0.9)^n u(n)$ convolved with itself

2.6 $H_{ap}(n)$ is a causal and stable all-pass system with input $x(n)$ and output $y(n)$. Then we have

$$\sum_{n=0}^{\infty} |y(n)|^2 = \sum_{n=0}^{\infty} |x(n)|^2$$

Consider

$$x(n) = x_0(n) + x_1(n) \quad \text{and} \quad y_i(n) = h(n) * x_i(n)$$

Then

$$\begin{aligned} \sum_{n=0}^{\infty} |y_0(n) + y_1(n)|^2 &= \sum_{n=0}^{\infty} |x_0(n) + x_1(n)|^2 \\ \sum_{n=0}^{\infty} |y_0(n)|^2 + \sum_{n=0}^{\infty} |y_1(n)|^2 + \sum_{n=0}^{\infty} |y_0(n)y_1(n)| &= \sum_{n=0}^{\infty} |x_0(n)|^2 + \sum_{n=0}^{\infty} |x_1(n)|^2 \\ \sum_{n=0}^{\infty} |y_0(n)|^2 + \sum_{n=0}^{\infty} |y_0(n)y_1(n)| &= \sum_{n=0}^{\infty} |x_0(n)|^2 \end{aligned}$$

Hence

$$\sum_{n=0}^{\infty} |y_0(n)|^2 \leq \sum_{n=0}^{\infty} |x_0(n)|^2$$

Define $x_0(n)$ as 0 above n_0 , then

$$\sum_{n=0}^{n_0} |y(n)|^2 \leq \sum_{n=0}^{n_0} |x(n)|^2$$

2.7 Monotone phase-response property of a causal and stable PZ-AP system:

- (a) A real first-order system: Consider

$$H(z) = \frac{p - z^{-1}}{1 - p z^{-1}}, \quad |p| < 1$$

Then

$$\angle H(e^{j\omega}) = \arctan\left(\frac{\sin \omega}{\cos \omega - 1/p}\right) - \arctan\left(\frac{\sin \omega}{\cos \omega - p}\right)$$

Clearly, $\angle H(e^{j\omega})$ decreases from $\angle H(e^{j0}) = \pi$ to $\angle H(e^{j2\pi}) = -\pi$. To show that it decreases monotonically, consider

$$\begin{aligned} \frac{d}{d\omega} (\angle H(e^{j\omega})) &= \frac{d}{d\omega} \left(\arctan\left(\frac{\sin \omega}{\cos \omega - 1/p}\right) - \arctan\left(\frac{\sin \omega}{\cos \omega - p}\right) \right) \\ &= \frac{p^2 - 1}{1 + p^2 - 2(\cos \omega)p} \end{aligned}$$

which is negative for $|p| < 1$. This proves that $\angle H(e^{j\omega})$ decreases *monotonically* from $\angle H(e^{j0}) = \pi$ to $\angle H(e^{j2\pi}) = -\pi$.

(b) A real second-order (complex-conjugate pole-pair) system: The system function is

$$H(z) = \left[\frac{(r\angle\theta) - z^{-1}}{1 - (r\angle\theta)^* z^{-1}} \right] \left[\frac{(r\angle\theta)^* - z^{-1}}{1 - (r\angle\theta) z^{-1}} \right], \quad 0 < r < 1$$

Consider the term

$$H_1(z) \triangleq \frac{(r\angle\theta) - z^{-1}}{1 - (r\angle\theta) z^{-1}}$$

Then angle calculations are similar to those for the real first-order case if we rotate the coordinate system by the angle θ , that is,

$$\begin{aligned} \angle H_1(e^{j\omega}) &= \left[\arctan\left(\frac{\sin \omega}{\cos \omega - 1/r}\right) + \theta \right] - \left[\arctan\left(\frac{\sin \omega}{\cos \omega - r}\right) + \theta \right] \\ &= \arctan\left(\frac{\sin \omega}{\cos \omega - 1/r}\right) - \arctan\left(\frac{\sin \omega}{\cos \omega - r}\right) \end{aligned}$$

Thus following the arguments in (a), we conclude that $\angle H_1(e^{j\omega})$ decreases *monotonically* from $\angle H_1(e^{j0})$ to $\angle H_1(e^{j0}) - \pi$. Similarly, consider

$$H_2(z) \triangleq \frac{(r\angle\theta) - z^{-1}}{1 - (r\angle\theta) z^{-1}}$$

Then $\angle H_2(e^{j\omega})$ decreases *monotonically* from $\angle H_2(e^{j0})$ to $\angle H_2(e^{j0}) - \pi$. Finally, since

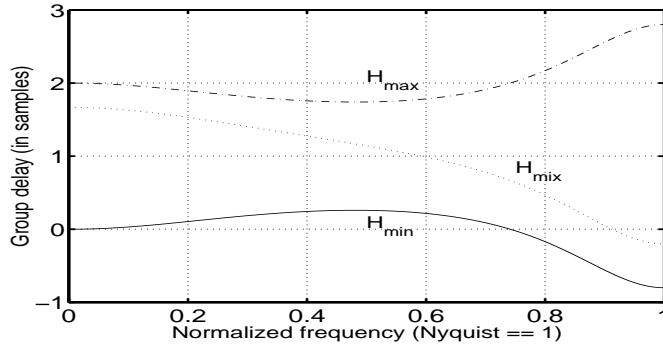
$$\angle H_1(e^{j\omega}) = \angle H_1(e^{j\omega}) + \angle H_2(e^{j\omega})$$

we conclude that $\angle H(e^{j\omega})$ decreases *monotonically* from $\angle H(e^{j0})$ to $\angle H(e^{j0}) - 2\pi$.

(c) Generalizing the above result for a real, causal, and stable PZ-AP system with P pole-zero pairs, we can show that $\angle H(e^{j\omega})$ decreases *monotonically* from $\angle H(e^{j0})$ to $\angle H(e^{j0}) - 2\pi P$.

2.8 Minimum group delay property: Consider the systems

$$\begin{aligned} H_{\min}(z) &= (1 - 0.25z^{-1})(1 + 0.5z^{-1}) \\ H_{\max}(z) &= (0.25 - z^{-1})(0.5 + z^{-1}) \\ H_{\text{mix}}(z) &= (1 - 0.25z^{-1})(0.5 + z^{-1}) \end{aligned}$$

**Figure 2.8:** Group delay plots

(a) Group delay response plots are shown in Figure 2.8.

(b) Proof of minimum group delay property

2.9 Minimum- and maximum-phase components

(a)

$$\begin{aligned} R_y(z) &= \frac{1 - 2.5z^{-1} + z^{-2}}{1 - 2.05z^{-1} + z^{-2}} \\ &= \frac{(1 - \frac{1}{2}z^{-1})(1 - 2z^{-1})}{(1 - \frac{4}{5}z^{-1})(1 - \frac{5}{4}z^{-1})} \end{aligned}$$

Hence the minimum phase component is $\frac{(1 - \frac{1}{2}z^{-1})}{(1 - \frac{4}{5}z^{-1})}$ and the maximum phase component is $\frac{(1 - 2z^{-1})}{(1 - \frac{5}{4}z^{-1})}$.

(b)

$$\begin{aligned} R_y(z) &= \frac{3z^2 - 10 + 3z^{-2}}{3z^2 + 10 + 3z^{-2}} \\ &= \frac{(1 - \frac{1}{3}z^{-2})(1 - 3z^{-2})}{(1 + \frac{1}{3}z^{-2})(1 + 3z^{-2})} \end{aligned}$$

Hence the minimum phase component is $\frac{(1 - \frac{1}{3}z^{-2})}{(1 + \frac{1}{3}z^{-2})}$ and the maximum phase component is $\frac{(1 - 3z^{-2})}{(1 + 3z^{-2})}$.

2.10 Consider the all-pass system function

$$H_{\text{ap}}(z) = \frac{1 - \alpha z^{-1}}{z^{-1} - \alpha^*}, |\alpha| < 1$$

(a)

$$\begin{aligned} |H_{\text{ap}}(z)|^2 &= H_{\text{ap}}(z)H_{\text{ap}}^*(z) \\ &= \left(\frac{1 - \alpha z^{-1}}{z^{-1} - \alpha^*}\right)\left(\frac{1 - \alpha^* z^{-1*}}{z^{-1*} - \alpha}\right) \\ &= \left(\frac{1 - \alpha z^{-1}}{z^{-1} - \alpha^*}\right)\left(\frac{1 - \alpha^* \frac{z}{|z|^2}}{\frac{z}{|z|^2} - \alpha}\right) \text{ where } z^{-1*} = \frac{z}{|z|^2} \\ &= \frac{|z|^2 - \alpha z^{-1} |z|^2 - \alpha^* z - |\alpha|^2}{1 - \alpha^* z - \alpha z^{-1} |z|^2 + |\alpha|^2 |z|^2} \end{aligned}$$

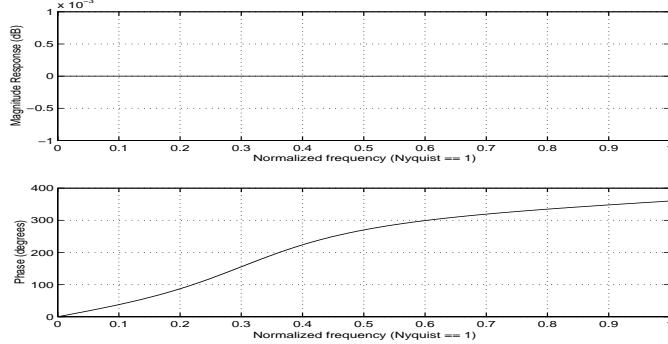


Figure 2.11: Frequency Response of $H(e^{j\omega})$

(b)

$$\begin{aligned}
 D_{|H|^2}(z) &= |z|^2 - \alpha z^{-1} |z|^2 - \alpha^* z - |\alpha|^2 \\
 A_{|H|^2}(z) &= 1 - \alpha^* z - \alpha z^{-1} |z|^2 + |\alpha|^2 |z|^2 \\
 D_{|H|^2}(z) - A_{|H|^2}(z) &= |z|^2 - |\alpha|^2 - 1 - |\alpha|^2 |z|^2 \\
 &= (|z|^2 - 1)(1 - |\alpha|^2)
 \end{aligned}$$

(c) Using the result from part 2 above (for $|\alpha| < 1$),

$$\begin{aligned}
 |H_{ap}(z)| > 1 &\quad \text{when} \quad (D_{|H|^2}(z) - A_{|H|^2}(z)) > 0 \Rightarrow |z| > 1 \\
 |H_{ap}(z)| = 1 &\quad \text{when} \quad (D_{|H|^2}(z) - A_{|H|^2}(z)) > 0 \Rightarrow |z| = 1 \\
 |H_{ap}(z)| < 1 &\quad \text{when} \quad (D_{|H|^2}(z) - A_{|H|^2}(z)) > 0 \Rightarrow |z| < 1
 \end{aligned}$$

2.11 Consider the system function

$$H(z) = \frac{a + b z^{-1} + c z^{-2}}{c + b z^{-1} + a z^{-2}}$$

(a) Magnitude of the frequency response function:

$$\begin{aligned}
 |H(e^{j\omega})|^2 &= H(e^{j\omega})H(e^{-j\omega}) \\
 &= \left(\frac{a + b e^{-j\omega} + c e^{-2j\omega}}{c + b e^{-j\omega} + a e^{-2j\omega}} \right) \left(\frac{a + b e^{j\omega} + c e^{2j\omega}}{c + b e^{j\omega} + a e^{2j\omega}} \right)
 \end{aligned}$$

Simplifying

$$|H(e^{j\omega})|^2 = 1, \text{ for all } \omega \Rightarrow |H(e^{j\omega})| = 1, \text{ for all } \omega$$

(b) Magnitude and phase response plots are shown in Figure 2.11.

2.12 Consider the third-order FIR system

$$\begin{aligned}
 H(z) &= 12 + 28z^{-1} - 29z^{-2} - 60z^{-3} \\
 &= 12(1 + 2.5z^{-1})(1 + 1.33z^{-1})(1 - 1.5z^{-1})
 \end{aligned}$$

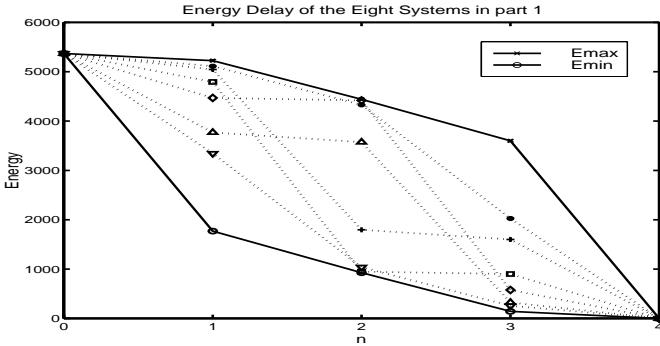


Figure 2.12: Energy Delay of Eight Systems

- (a) Therefore, an FIR system with three zeros results in eight (2^3) FIR systems with identical magnitude responses. They are obtained by reflecting each zero about the unit circle and applying the appropriate

	Zero@	z_1	z_2	z_3	Gain	System
gain factor.	$H_{\max}(z)$	-2.5	-1.33	1.5	12	$12 + 28z^{-1} - 29z^{-2} - 60z^{-3}$
	$H_{\min}(z)$	-0.4	-0.75	0.67	60	$60 + 29z^{-1} - 28z^{-2} - 12z^{-3}$
	$H_{\text{mix}1}(z)$	-2.5	-1.33	0.67	18	$18 + 57z^{-1} + 14z^{-2} - 40z^{-3}$
	$H_{\text{mix}2}(z)$	-2.5	-0.75	1.5	16	$16 + 28z^{-1} - 48z^{-2} - 45z^{-3}$
	$H_{\text{mix}3}(z)$	-2.5	-0.75	0.67	24	$24 + 62z^{-1} - 7z^{-2} - 30z^{-3}$
	$H_{\text{mix}4}(z)$	-0.4	-1.33	1.5	30	$30 + 7z^{-1} - 62z^{-2} - 24z^{-3}$
	$H_{\text{mix}5}(z)$	-0.4	-1.33	0.67	45	$45 + 48z^{-1} - 28z^{-2} - 16z^{-3}$
	$H_{\text{mix}6}(z)$	-0.4	-0.75	1.5	40	$40 - 14z^{-1} - 57z^{-2} - 18z^{-3}$

- (b) Clearly from the above table, $H_{\max}(z)$ and H_{\min} are the maximum and minimum phase systems respectively.
(c) Energy delay plots are shown in Figure 2.12.

2.13 Consider the system function

$$H(z) = \frac{1 + z^{-1} - 6z^{-2}}{1 + \frac{1}{4}z^{-1} - \frac{1}{8}z^{-2}}$$

- (a) After factorization, we obtain

$$H(z) = \frac{(1 + 3z^{-1})(1 - 2z^{-1})}{(1 + \frac{1}{2}z^{-1})(1 - \frac{1}{4}z^{-1})}$$

Zeros outside unit circle \Rightarrow Not minimum phase.

- (b) Minimum-phase system:

$$\begin{aligned} (1 + 3z^{-1})(1 - 2z^{-1}) &\Rightarrow 6(1 + \frac{1}{3}z^{-1})(1 - \frac{1}{2}z^{-1}) \\ &= 6 - z^{-1} - z^{-2} \end{aligned}$$

Hence

$$H_{\min}(z) = \frac{6 - z^{-1} - z^{-2}}{1 + (\frac{1}{4})z^{-1} - (\frac{1}{8})z^{-2}}$$

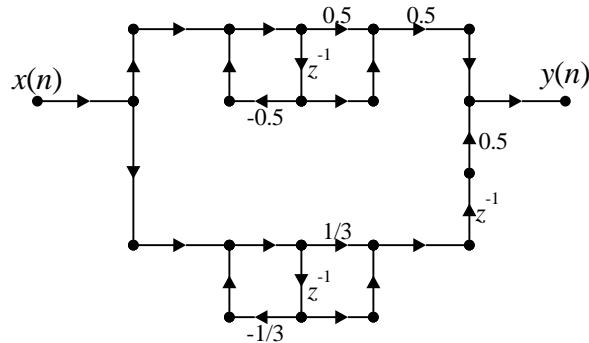


Figure 2.14: Parallel Structure of two AP systems

(c) This system is a Maximum-phase system since all zeros are outside the unit circle.

2.14 Parallel connection of two all-pass systems:

$$\begin{aligned}
 H(z) &= \frac{3(1+z^{-1})(1+z^{-1})(1+z^{-1})}{12(1+\frac{1}{2}z^{-1})(1+\frac{1}{3}z^{-1})} \\
 &= \frac{3}{12} \left[\frac{A(\frac{1}{2}+z^{-1})}{1+\frac{1}{2}z^{-1}} + \frac{Bz^{-1}(\frac{1}{3}+z^{-1})}{1+\frac{1}{3}z^{-1}} \right] \\
 \Rightarrow A &= 2, B = 2 \\
 H(z) &= \frac{\frac{1}{2}(\frac{1}{2}+z^{-1})}{1+\frac{1}{2}z^{-1}} + \frac{\frac{1}{2}z^{-1}(\frac{1}{3}+z^{-1})}{1+\frac{1}{3}z^{-1}}
 \end{aligned}$$

The block diagram is shown in Figure 2.14.

2.15 Impulse response of an all-pole system with lattice parameters:

$$k_1 = 0.2; \ k_2 = 0.2; \ k_3 = 0.2; \ k_4 = 0.2;$$

The direct form coefficients $\{a_k\}_0^4$ can be computed using the `lat2dir` function. The corresponding system function is

$$H(z) = \frac{1}{1 + 0.76z^{-1} - 0.731z^{-2} + 0.787z^{-3} + 0.7z^{-4}}$$

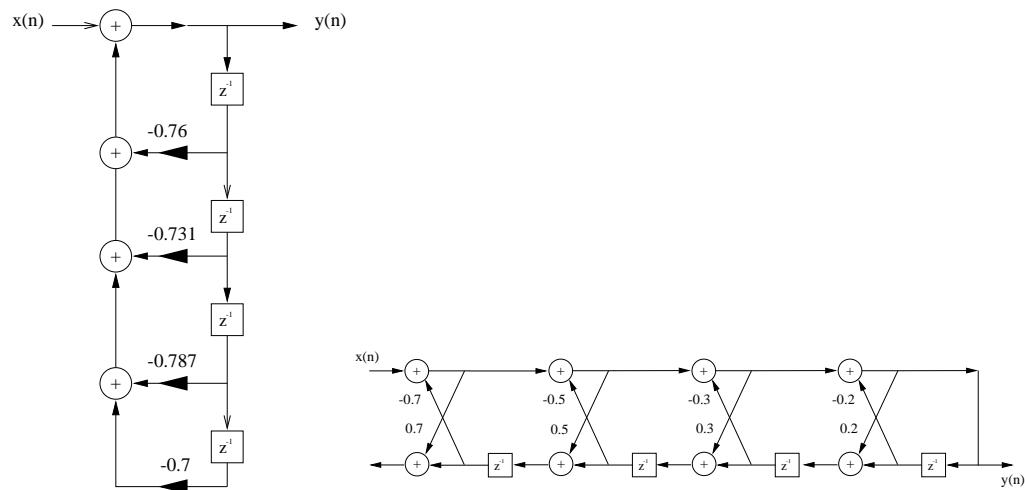
Using partial fraction expansion we obtain

$$H(z) = \frac{0.3506 - 0.4334z^{-1}}{1 - 0.6654z^{-1} + 0.9109z^{-2}} + \frac{0.6492 + 0.3656z^{-1}}{1 + 0.4254z^{-1} + 0.7685z^{-2}}$$

Hence upon inverse transformation we obtain

$$h(n) = [0.3506 (0.9544)^n \cos(0.3867\pi n) - 0.3540 (0.9544)^n \sin(0.3867\pi n)] u(n) + [0.6494 (0.8766)^n \cos(0.8021\pi n) - 0.1903 (0.8766)^n \sin(0.8021\pi n)] u(n)$$

The direct- and lattice-form structures are shown in Figure 2.15.

**Figure 2.15:** Direct and Lattice Form Structures

Chapter 3
Random Variables, Random Vectors, & Stochastic Processes

3.1 Exponential Density: $f_x(x) = \frac{1}{a}e^{-x/a} u(x)$

(a) Density plot for $a = 1$: $f_x(x) = e^{-x} u(x)$ is shown in Figure 3.1

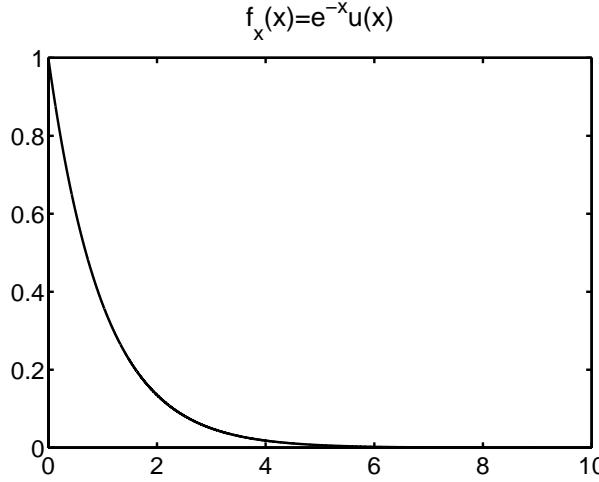


Figure 3.1: Exponential density function

(b) Moments:

- i. Mean: $\mu_x = \int_{-\infty}^{\infty} x f_x(x) dx = 1/a = 1$
- ii. Variance: $\sigma_x^2 = \int_{-\infty}^{\infty} (x - \mu_x)^2 f_x(x) dx = (1/a)^2 = 1$
- iii. Skewness: The third central moment is given by

$$\gamma_x^{(3)} = \int_{-\infty}^{\infty} (x - \mu_x)^3 f_x(x) dx = \int_0^{\infty} (x - 1)^3 (e^{-x}) dx = 2$$

Hence

$$\text{skewness} = \frac{1}{\sigma_x^3} \gamma_x^{(3)} = 2 \quad (\Rightarrow \text{leaning towards right})$$

- iv. Kurtosis: The fourth central moment is given by

$$\gamma_x^{(4)} = \int_{-\infty}^{\infty} (x - \mu_x)^4 f_x(x) dx = \int_0^{\infty} (x - 1)^4 (e^{-x}) dx = 9$$

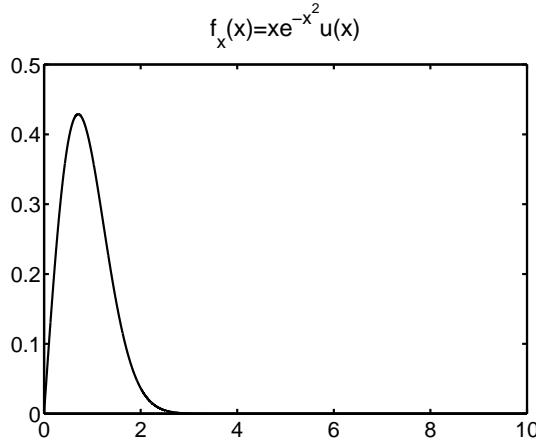
Hence

$$\text{kurtosis} = \frac{1}{\sigma_x^4} \gamma_x^{(4)} - 3 = 9 - 3 = 4$$

which means a much flatter shape compared to the Gaussian shape.

(c) Characteristic function:

$$\Phi_x = E\{e^{sx(\xi)}\} = \int_{-\infty}^{\infty} f_x(x) e^{sx} dx = \int_0^{\infty} \frac{1}{a} e^{-x(\frac{1}{a}-s)} dx = \frac{\frac{1}{a}}{\frac{1}{a}-s} = \frac{1}{1-as}$$

**Figure 3.2:** Rayleigh density function

3.2 Rayleigh density : $f_x(x) = \frac{x}{\sigma^2} e^{-x^2/(2\sigma^2)} u(x)$

(a) Density function for $\sigma = 1$: $f_x(x) = x e^{-x^2/2} u(x)$ is shown in Figure 3.2.

(b) Moments:

i. Mean: $\mu_x = \int_{-\infty}^{\infty} x f_x(x) dx = \int_0^{\infty} x^2 e^{-x^2/2} dx = \sqrt{\pi/2}$

ii. Variance: $\sigma_x^2 = \int_{-\infty}^{\infty} (x - \mu_x)^2 f_x(x) dx = \int_0^{\infty} (x - \sqrt{\pi/2})^2 x e^{-x^2/2} dx = 2 - \frac{\pi}{2}$

iii. Skewness: The third central moment is given by

$$\gamma_x^{(3)} = \int_{-\infty}^{\infty} (x - \mu_x)^3 f_x(x) dx = \int_0^{\infty} \left(x - \sqrt{\pi/2}\right)^3 (x e^{-x^2/2}) dx = \frac{1}{\sqrt{2}} (\sqrt{\pi})^3 - \frac{3}{\sqrt{2}} \sqrt{\pi}$$

Hence

$$\text{skewness} = \frac{1}{\sigma_x^3} \gamma_x^{(3)} = \frac{\frac{1}{\sqrt{2}} (\sqrt{\pi})^3 - \frac{3}{\sqrt{2}} \sqrt{\pi}}{\left(\sqrt{2 - \frac{\pi}{2}}\right)^3} = .63111 \text{ (} \Rightarrow \text{leaning towards right)}$$

iv. Kurtosis: The fourth central moment is given by

$$\gamma_x^{(4)} = \int_{-\infty}^{\infty} (x - \mu_x)^4 f_x(x) dx = \int_0^{\infty} \left(x - \sqrt{\pi/2}\right)^4 (x e^{-x^2/2}) dx = 8 - \frac{3\pi^2}{4}$$

: Hence

$$\text{kurtosis} = \frac{1}{\sigma_x^4} \gamma_x^{(4)} - 3 = \frac{8 - \frac{3\pi^2}{4}}{\left(2 - \frac{\pi}{2}\right)^2} - 3 = 0.2451$$

which means a flatter but almost a Gaussian shape.

(c) Characteristic function ($\sigma = 1$):

$$\begin{aligned} \Phi_x(s) &= E\{e^{sx(\xi)}\} = \int_{-\infty}^{\infty} f_x(x) e^{sx} dx = \int_0^{\infty} x e^{-x^2} e^{sx} dx \\ &= \frac{1}{4} s \sqrt{\pi} e^{\frac{1}{4}s^2} + \frac{1}{2} + \frac{1}{4} s \sqrt{\pi} e^{\frac{1}{4}s^2} \operatorname{erf}\left(\frac{1}{2}s\right) \end{aligned}$$

3.3 Relation between $r_x^{(m)}$ and $\gamma_x^{(m)}$:

$$\begin{aligned}\gamma_x^{(m)} &= E\{[x(\zeta) - \mu_x]^m\} = E\left\{\sum_{k=0}^m (-1)^k \binom{m}{k} x(\zeta)^k \mu_x^{m-k}\right\} \\ &= \sum_{k=0}^m (-1)^k \binom{m}{k} \mu_x^k E\{x(\zeta)^k\} \\ \gamma_x^{(m)} &= \sum_{k=0}^m (-1)^k \binom{m}{k} \mu_x^k r_x^{(n-k)}\end{aligned}$$

Similarly

$$\begin{aligned}r_x^{(m)} &= E\{(\underbrace{[x(\zeta) - \mu_x]}_{y(\zeta)} + \mu_x)^m\} = E\left\{\sum_{k=0}^m \binom{m}{k} y(\zeta)^k \mu_x^{m-k}\right\} \\ &= \sum_{k=0}^m \binom{m}{k} \mu_x^k E\{y(\zeta)^k\} \\ r_x^{(m)} &= \sum_{k=0}^m \binom{m}{k} \mu_x^k \gamma_x^{(n-k)}\end{aligned}$$

3.4 First four cumulants: The cumulants are given by

$$\kappa_x^{(m)} \triangleq \left. \frac{d^m [\bar{\Psi}_x(s)]}{ds^m} \right|_{s=0}$$

where

$$\bar{\Psi}_x(s) = \ln \bar{\Phi}_x(s) = \ln [E\{e^{sx(\zeta)}\}] = \ln [\Phi_x(s)]$$

and

$$\Phi_x(s) = \sum_{m=0}^{\infty} \frac{s^m}{m!} r_x^{(m)}$$

Thus

$$\begin{aligned}\kappa_x^{(1)} &= \left. \left(\frac{1}{\Phi_x(s)} \right) \left(\frac{d\Phi_x(s)}{ds} \right) \right|_{s=0} \\ &= \left. \frac{\sum_{m=0}^{\infty} \frac{s^m}{m!} r_x^{(m+1)}}{\sum_{m=0}^{\infty} \frac{s^m}{m!} r_x^{(m)}} \right|_{s=0} = r_x^{(1)} = \mu_x = 0 \quad [\because \text{zero mean } x(\zeta)]\end{aligned}$$

Similarly

$$\begin{aligned}\kappa_x^{(2)} &= \left. \frac{d}{ds} \left[\frac{\sum_{m=0}^{\infty} \frac{s^m}{m!} r_x^{(m+1)}}{\sum_{m=0}^{\infty} \frac{s^m}{m!} r_x^{(m)}} \right] \right|_{s=0} \\ &= \left. \frac{\left(\sum_{m=0}^{\infty} \frac{s^m}{m!} r_x^{(m+2)} \right) \left(\sum_{m=0}^{\infty} \frac{s^m}{m!} r_x^{(m)} \right) - \left(\sum_{m=0}^{\infty} \frac{s^m}{m!} r_x^{(m+1)} \right)^2}{\left(\sum_{m=0}^{\infty} \frac{s^m}{m!} r_x^{(m)} \right)^2} \right|_{s=0, r_x^{(1)}=0} \\ &= r_x^{(2)} = \sigma_x^2\end{aligned}$$

$$\begin{aligned}\kappa_x^{(3)} &= \frac{d}{ds} \left[\frac{\left(\sum_{m=0}^{\infty} \frac{s^m}{m!} r_x^{(m+2)} \right) \left(\sum_{m=0}^{\infty} \frac{s^m}{m!} r_x^{(m)} \right) - \left(\sum_{m=0}^{\infty} \frac{s^m}{m!} r_x^{(m+1)} \right)^2}{\left(\sum_{m=0}^{\infty} \frac{s^m}{m!} r_x^{(m)} \right)^2} \right] \Big|_{s=0, r_x^{(1)}=0} \\ &= \gamma_x^{(3)}\end{aligned}$$

3.5 Random vector $\mathbf{x}(\zeta) = [x_1(\zeta) \ x_2(\zeta)]$

(a) Mean vector:

$$\mu_y = \mathbf{A}\mu_x = \begin{bmatrix} 1 & 3 \\ -1 & 2 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 7 \\ 3 \\ 8 \end{bmatrix}$$

(b) Autocorrelation matrix:

$$\begin{aligned}\Gamma_y &= \mathbf{A}\Gamma_x\mathbf{A}^H = \begin{bmatrix} 1 & 3 \\ -1 & 2 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} 4 & 0.8 \\ 0.8 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & 2 \\ 3 & 2 & 3 \end{bmatrix} \\ &= \begin{bmatrix} 17.8 & 1.2 & 24.2 \\ 1.2 & 4.8 & -1.2 \\ 24.2 & -1.2 & 34.6 \end{bmatrix}\end{aligned}$$

(c) Crosscorrelation matrix:

$$\begin{aligned}\mathbf{R}_x &= \Gamma_x + \mu_x\mu_x^H = \begin{bmatrix} 5 & 2.8 \\ 2.8 & 5 \end{bmatrix} \\ \mathbf{R}_{xy} &= \mathbf{R}_x\mathbf{A}^H = \begin{bmatrix} 13.4 & 0.6 & 18.4 \\ 17.8 & 7.2 & 20.6 \end{bmatrix}\end{aligned}$$

3.6 Let $\mathbf{x}(\zeta)$ be a Gaussian random vector with mean vector μ_x and covariance matrix Γ_x , then the characteristic function is

$$\Phi_x(\xi) = \exp \left[j\xi^T \mu_x - \frac{1}{2} \xi^T \Gamma_x \xi \right]$$

Let $\mathbf{y}(\zeta) = \mathbf{Ax}$ where \mathbf{A} is a non-singular matrix. Its characteristic function is given by

$$\begin{aligned}\Phi_y(\xi) &= E \left\{ e^{j\xi^T \mathbf{y}} \right\} = E \left\{ e^{j\xi^T \mathbf{Ax}} \right\} \\ &= E \left\{ e^{j(\mathbf{A}^T \xi)^T \mathbf{x}} \right\} = \exp \left[j (\mathbf{A}^T \xi)^T \mu_x - \frac{1}{2} (\mathbf{A}^T \xi)^T \Gamma_x (\mathbf{A}^T \xi) \right] \\ &= \exp \left[j \xi^T (\mathbf{A} \mu_x) - \frac{1}{2} \xi^T (\mathbf{A} \Gamma_x \mathbf{A}^T) \xi \right]\end{aligned}$$

which is a characteristic function of a Gaussian random vector with mean vector $\mathbf{A}\mu_x$ and covariance matrix $\mathbf{A}\Gamma_x\mathbf{A}^T$.

3.7 Sum of independent exponential random variables $x_k(\zeta)$: $f_{x_k}(x) = e^{-x} u(x)$.

(a) $y_2(\zeta) = x_1(\zeta) + x_2(\zeta)$. Hence

$$\begin{aligned}f_{y_2}(x) &= f_{x_1}(x) * f_{x_2}(x) = \left[\int_0^x e^{-s} e^{s-x} ds \right] u(x) \\ &= x e^{-x} u(x)\end{aligned}$$

The plot of this density is shown in Figure 3.7(a).

(b) $y_3(\zeta) = y_2(\zeta) + x_3(\zeta)$. Hence

$$\begin{aligned} f_{y_3}(x) &= f_{y_2}(x) * f_{x_3}(x) = \left[\int_0^x s e^{-s} e^{s-x} ds \right] u(x) \\ &= \frac{1}{2} x^2 e^{-x} u(x) \end{aligned}$$

The plot of this density is shown in Figure 3.7(b).

(c) $y_4(\zeta) = y_3(\zeta) + x_4(\zeta)$. Hence

$$\begin{aligned} f_{y_4}(x) &= f_{y_3}(x) * f_{x_4}(x) = \frac{1}{2} \left[\int_0^x s^2 e^{-s} e^{s-x} ds \right] u(x) \\ &= \frac{1}{6} x^3 e^{-x} u(x) \end{aligned}$$

The plot of this density is shown in Figure 3.7(c).

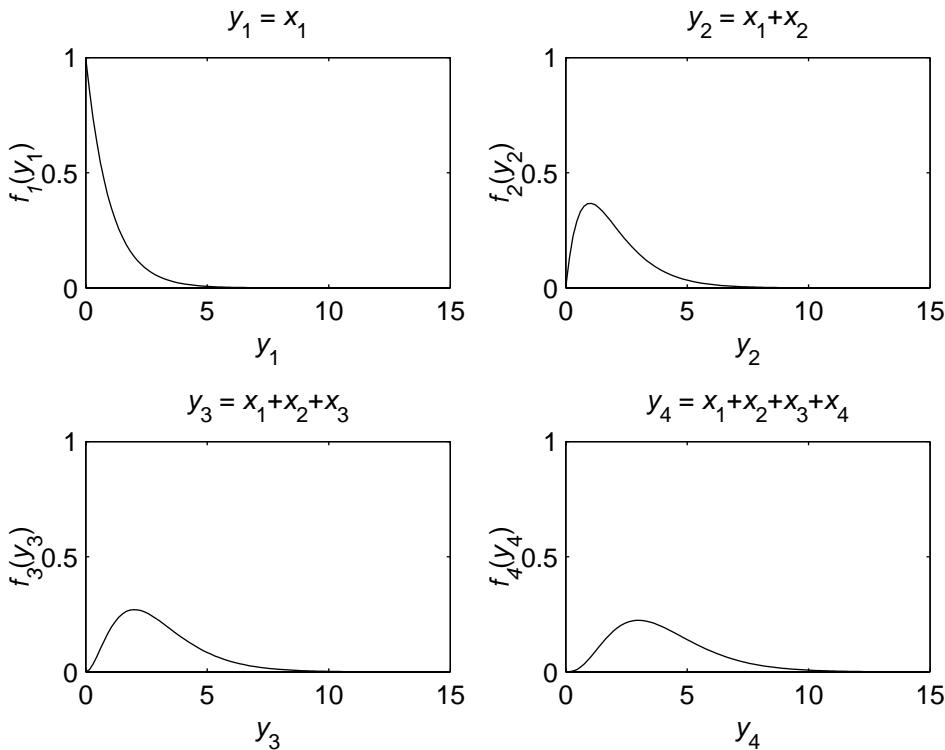


Figure 3.7: Sums of IID exponentially distributed random variables

(d) As k increases, the distribution of $y_k(\zeta)$ approaches a Gaussian distribution, with a mean equal to the sum of the exponential distribution means.

3.8 Test of WSS: $\mu_x = \text{constant}$ and $r_x(n_1, n_2) = r_x(n_1 - n_2)$

Test of m.s. ergodicity in the mean: $E \left\{ \frac{1}{2N+1} \sum_{-N}^N x(n, \zeta) \right\} = \mu_x$ and $\text{var} \left\{ \frac{1}{2N+1} \sum_{-N}^N x(n, \zeta) \right\} \xrightarrow{N \rightarrow \infty} 0$

(a) $x(n, \zeta) = A(\zeta)$, where random variable $A(\zeta)$ is uniformly distributed between 0 and 1. Now

$$\mu_x(n) = E \{x(n, \zeta)\} = E \{A(\zeta)\} = \frac{1}{2}$$

and

$$r_x(n_1, n_2) = E\{x(n_1, \zeta)x(n_2, \zeta)\} = E\{A^2(\zeta)\} = \frac{1}{3}$$

Hence the process is WSS. Consider

$$E\left\{\frac{1}{2N+1} \sum_{-N}^N x(n, \zeta)\right\} = \frac{1}{2N+1} \sum_{-N}^N E\{x(n, \zeta)\} = E\{A(\zeta)\} = \frac{1}{2} = \mu_x$$

However, the computed mean remains at the observed value of $x(n, \zeta_0) \triangleq a_0$ and does not converge to the true mean of 1/2. Hence the process is not m.s. ergodic in the mean.

(b) $x(n, \zeta) = A(\zeta) \cos \omega_0 n$, where random variable $A(\zeta)$ is a unit Gaussian random variable. Then

$$\mu_x(n) = E\{A(\zeta) \cos \omega_0 n\} = E\{A(\zeta)\} \cos \omega_0 n = 0$$

and

$$\begin{aligned} r_x(n_1, n_2) &= E\{x(n_1, \zeta)x(n_2, \zeta)\} = E\{A^2(\zeta)\} \cos \omega_0 n_1 \cos \omega_0 n_2 \\ &= \cos \omega_0 n_1 \cos \omega_0 n_2 \end{aligned}$$

Thus the process is not WSS. Therefore, the m.s. ergodicity cannot be determined because the process must be WSS.

(c) $x(n, \zeta)$ is Bernoulli process with $\Pr[x(n, \zeta) = 1] = p$ and $\Pr[x(n, \zeta) = -1] = 1 - p$. Then

$$\mu_x(n) = E\{x(n, \zeta)\} = p + (-1)(1 - p) = 2p - 1$$

and

$$r_x(n_1, n_2) = \begin{cases} (2p - 1)^2, & n_1 \neq n_2 \\ 1, & n_1 = n_2 \end{cases} \quad (5)$$

Hence it is WSS. Consider the random variable $\langle x(n, \zeta) \rangle_N = \frac{1}{2N+1} \sum_{-N}^N x(n, \zeta)$

$$E\{\langle x(n, \zeta) \rangle_N\} = \frac{1}{2N+1} \sum_{-N}^N E\{x(n, \zeta)\} = 2p - 1 = \mu_x(n)$$

Now consider the second moment of the random variable $\langle x(n, \zeta) \rangle_N$

$$\begin{aligned} E\left\{\left(\langle x(n, \zeta) \rangle_N\right)^2\right\} &= \frac{1}{(2N+1)^2} E\left\{\sum_{-N}^N x(n_1, \zeta) \sum_{-N}^N x(n_2, \zeta)\right\} \\ &= \frac{1}{(2N+1)^2} E\left\{\sum_{-N}^N \sum_{-N}^N x(n_1, \zeta)x(n_2, \zeta)\right\} \\ &= \frac{1}{(2N+1)^2} \sum_{-N}^N \sum_{-N}^N r_x(n_1, n_2) \end{aligned}$$

Let $M = 2N + 1$ and using (5)

$$E\left\{\left(\langle x(n, \zeta) \rangle_N\right)^2\right\} = \frac{1}{M^2} [M + (M^2 - M)(2p - 1)^2]$$

Thus

$$\begin{aligned}\text{var} \left[(\langle x(n, \zeta) \rangle_N)^2 \right] &= \text{E} \left\{ (\langle x(n, \zeta) \rangle_N)^2 \right\} - \{\text{E} \{ \langle x(n, \zeta) \rangle_N \}\}^2 \\ &= \frac{1}{M^2} [M + (M^2 - M)(2p - 1)^2] - (2p - 1)^2 \\ &\approx \frac{1}{M^2} [(M^2)(2p - 1)^2] - (2p - 1)^2 \text{ for large } M\end{aligned}$$

Thus $\text{var} \left[(\langle x(n, \zeta) \rangle_N)^2 \right] \rightarrow 0$ as $N \rightarrow \infty$. Hence the process is m.s. ergodic in the mean.

3.9 Harmonic process: $x(n) = \sum_{k=1}^N A_k \cos(\omega_k n + \phi_k)$, ϕ_k uniformly distributed over $[0, 2\pi]$

(a) Mean:

$$\begin{aligned}\text{E}\{x(n)\} &= \text{E}\left\{\sum_{k=1}^M A_k \cos(\omega_k n + \phi_k)\right\} = \sum_{k=1}^M A_k \text{E}\{\cos(\omega_k n + \phi_k)\} \\ &= \frac{1}{2\pi} \sum_{k=1}^M A_k \int_0^{2\pi} \cos(\omega_k n + \phi_k) d\phi_k = 0\end{aligned}$$

(b) Autocorrelation:

$$\begin{aligned}x(n) &= \sum_{k=1}^M A_k \cos(\omega_k n + \phi_k) \\ \stackrel{\mathcal{F}}{\Leftrightarrow} X(\omega) &= \frac{1}{2} \sum_{k=1}^M A_k (\delta(\omega - \omega_k) e^{-j\phi_k} + \delta(\omega + \omega_k) e^{j\phi_k}) \\ \stackrel{| \cdot |^2}{\Leftrightarrow} R_x(e^{j\omega}) &= \frac{1}{4} \sum_{k=1}^M |A_k|^2 (\delta(\omega - \omega_k) + \delta(\omega + \omega_k)) \\ \stackrel{\mathcal{F}^{-1}}{\Leftrightarrow} r_x(l) &= \frac{1}{2} \sum_{k=1}^M |A_k|^2 \cos \omega_k l\end{aligned}$$

3.10 Harmonic process with ϕ_k distributed as $f_{\phi_k}(\phi_k) = (1 + \cos \phi_k) / 2\pi$, $-\pi \leq \phi_k \leq \pi$. Consider the mean sequence

$$\begin{aligned}\text{E}\{x(n)\} &= \text{E}\left\{\sum_{k=1}^M A_k \cos(\omega_k n + \phi_k)\right\} = \sum_{k=1}^M A_k \text{E}\{\cos(\omega_k n + \phi_k)\} \\ &= \frac{1}{2\pi} \sum_{k=1}^M A_k \int_{-\pi}^{\pi} \cos(\omega_k n + \phi_k) (1 + \cos \phi_k) d\phi_k = (\cos \omega_k n) \pi\end{aligned}$$

Thus the process is not stationary.

$$\begin{aligned}\text{3.11 } \mu_x &= 4; \quad \gamma_x(n) = \begin{cases} 4 - |n|; & |n| \leq 3 \\ 0; & \text{otherwise} \end{cases} = \begin{cases} 1, 2, 3, 4, 3, 2, 1 \\ \uparrow \end{cases} \\ h(n) &= u(n) - u(n-4) = \{1, 1, 1, 1\}\end{aligned}$$

(a) Mean: $\mu_y(n) = \mu_x \sum_{-\infty}^{\infty} h(k) = 4 \times 4 = 16$

(b) Crosscorrelation: $\gamma_{xy}(n_1, n_2) = \gamma_x(n_1 - n_2) * h^*(n_2 - n_1)$ Hence

$$\begin{aligned}\gamma_{xy}(k) &= \gamma_x(k) * h^*(-k) = \{1, 2, 3, 4, 3, 2, 1\} * \{1, 1, 1, 1\} \\ &= \{1, 3, 6, 10, 12, 12, 10, 6, 3, 1\}\end{aligned}$$

(c) Autocorrelation: $\gamma_y(n_1, n_2) = h(n_1 - n_2) * \gamma_{xy}(n_1, n_2)$ Hence

$$\begin{aligned}\gamma_y(k) &= h(k) * \gamma_{xy}(k) \\ &= \{1, 1, 1, 1\} * \{1, 3, 6, 10, 12, 12, 10, 6, 3, 1\} \\ &= \{1, 4, 10, 20, 31, 40, 44, 40, 31, 20, 10, 4, 1\}\end{aligned}$$

3.12 LTI system: $y(n) = \frac{1}{2}y(n-1) + x(n) + \frac{1}{3}x(n-1) \Rightarrow H(z) = \frac{1+\frac{1}{2}z^{-1}}{1-\frac{1}{2}z^{-1}}$, $|z| < \frac{1}{2}$

Input process: zero-mean, WSS with $r_x(l) = 0.5^{|l|} \Rightarrow R_x(z) = \frac{\frac{3}{4}}{\frac{5}{4}-\frac{1}{2}(z+z^{-1})}$, $\frac{1}{2} < |z| < 2$

(a) PSD and autocorrelation of the output: Consider

$$\begin{aligned}R_y(z) &= H(z)H^*(1/z^*)R_x(z) = H(z)H(z^{-1})R_x(z) \\ &= \left(\frac{1+\frac{1}{3}z^{-1}}{1-\frac{1}{2}z^{-1}}\right)\left(\frac{1+\frac{1}{3}z}{1-\frac{1}{2}z}\right)\left(\frac{\frac{3}{4}}{\frac{5}{4}-\frac{1}{2}(z+z^{-1})}\right) \\ &= \frac{\frac{5}{6} + \frac{1}{4}(z+z^{-1})}{\left[\frac{5}{4} - \frac{1}{2}(z+z^{-1})\right]^2}, \quad \frac{1}{2} < |z| < 2\end{aligned}$$

Hence the PSD is

$$\begin{aligned}R_y(e^{j\omega}) &= \frac{\frac{5}{6} + \frac{1}{4}(e^{j\omega} + e^{-j\omega})}{\left[\frac{5}{4} - \frac{1}{2}(e^{j\omega} + e^{-j\omega})\right]^2} = \frac{\frac{5}{6} + \frac{1}{2}\cos\omega}{\left[\frac{5}{4} - \cos\omega\right]^2} \\ &= \frac{\frac{5}{6} + \frac{1}{2}\cos\omega}{\frac{33}{16} - \frac{5}{2}\cos\omega + \frac{1}{2}\cos 2\omega}\end{aligned}$$

To determine the autocorrelation, consider the PFE of

$$\begin{aligned}R_y(z) &= \frac{\frac{5}{6} + \frac{1}{4}(z+z^{-1})}{\left[\frac{5}{4} - \frac{1}{2}(z+z^{-1})\right]^2}, \quad \frac{1}{2} < |z| < 2 \\ &= \frac{4}{3} \frac{3z^3 + 10z^2 + 3z}{4z^4 - 20z^3 + 33z^2 - 20z + 4}, \quad \frac{1}{2} < |z| < 2 \\ &= \frac{4}{3} \left[\frac{35}{18(2z-1)^2} + \frac{253}{54(2z-1)} + \frac{70}{9(z-2)^2} - \frac{43}{27(z-2)} \right] \\ &= \frac{35}{27} z^{-1} \frac{\frac{1}{2}z^{-1}}{(1-\frac{1}{2}z^{-1})^2} + \frac{253}{81} z^{-1} \frac{1}{(1-\frac{1}{2}z^{-1})} \\ &\quad + \frac{140}{27} z^{-1} \frac{2z^{-1}}{(1-2z^{-1})^2} - \frac{172}{81} z^{-1} \frac{1}{(1-2z^{-1})}\end{aligned}$$

Hence upon inverse z -transformation, we obtain

$$\begin{aligned}r_y(l) &= \frac{35}{27} (l-1) \left(\frac{1}{2}\right)^{l-1} u(l-1) + \frac{253}{81} \left(\frac{1}{2}\right)^{l-1} u(l-1) \\ &\quad - \frac{140}{27} (l-1) 2^{l-1} u(-l) + \frac{172}{81} 2^{l-1} u(-l)\end{aligned}$$

(b) Cross-correlation and cross-PSD between the input and output: Consider

$$\begin{aligned} R_{xy}(z) &= H^*(1/z^*) R_x(z) = H(z^{-1}) R_x(z) \\ &= \left(\frac{1 + \frac{1}{3}z}{1 - \frac{1}{2}z} \right) \left(\frac{\frac{3}{4}}{\frac{5}{4} - \frac{1}{2}(z + z^{-1})} \right), \quad \frac{1}{2} < |z| < 2 \end{aligned}$$

Thus the cross-PSD is

$$R_{xy}(e^{j\omega}) = \frac{6e^{j\omega} + 2e^{2j\omega}}{-4 + 12e^{j\omega} - 9e^{j2\omega} + 2e^{j3\omega}}$$

To determine cross-correlation, consider the PFE of

$$\begin{aligned} R_{xy}(z) &= \left(\frac{1 + \frac{1}{3}z}{1 - \frac{1}{2}z} \right) \left(\frac{\frac{3}{4}}{\frac{5}{4} - \frac{1}{2}(z + z^{-1})} \right), \quad \frac{1}{2} < |z| < 2 \\ &= \frac{6z + 2z^2}{12z - 9z^2 - 4 + 2z^3}, \quad \frac{1}{2} < |z| < 2 \\ &= \frac{14}{9(2z-1)} + \frac{20}{3(z-2)^2} + \frac{2}{9(z-2)} \\ &= \frac{7}{9}z^{-1} \frac{1}{(1 - \frac{1}{2}z^{-1})} + \frac{10}{3}z^{-1} \frac{2z^{-1}}{(1 - 2z^{-1})^2} + \frac{2}{9}z^{-1} \frac{1}{(1 - 2z^{-1})} \end{aligned}$$

Hence upon inverse z -transformation, we obtain

$$r_{xy}(l) = \frac{7}{9} \left(\frac{1}{2} \right)^{l-1} u(l-1) - \frac{10}{3} (l-1) 2^{l-1} u(-l) - \frac{2}{9} 2^{l-1} u(-l)$$

3.13 LTI system: $y(n) = 0.6y(n-1) + x(n) + 1.25x(n-1) \Rightarrow H(z) = \frac{1+1.25z^{-1}}{1-0.6z^{-1}}, \quad |z| < 0.6$

Input process: WSS with PSD $R_x(e^{j\omega}) = \frac{1}{1.64+1.6\cos\omega}$

(a) The PSD of the output is given by

$$\begin{aligned} R_y(e^{j\omega}) &= |H(e^{j\omega})|^2 R_x(e^{j\omega}) = H(e^{j\omega}) H^*(e^{j\omega}) R_x(e^{j\omega}) \\ &= H(e^{j\omega}) H(e^{-j\omega}) R_x(e^{j\omega}) \\ &= \left(\frac{1 + 1.25e^{-j\omega}}{1 - 0.6e^{-j\omega}} \right) \left(\frac{1 + 1.25e^{j\omega}}{1 - 0.6e^{j\omega}} \right) R_x(e^{j\omega}) \\ &= \left(\frac{2.5625 + 2.5 \cos \omega}{1.36 - 1.2 \cos \omega} \right) \left(\frac{1}{1.64 + 1.6 \cos \omega} \right) \\ &= \frac{19.531}{17 - 15 \cos \omega} \end{aligned}$$

(b) The cross-PSD between input and the output is given by

$$\begin{aligned} R_{xy}(e^{j\omega}) &= H^*(e^{j\omega}) R_x(e^{j\omega}) = H(e^{-j\omega}) R_x(e^{j\omega}) \\ &= \left(\frac{1 + 1.25e^{j\omega}}{1 - 0.6e^{j\omega}} \right) \left(\frac{1}{1.64 + 1.6 \cos \omega} \right) \\ &= 31.25 \frac{4 + 5e^{j\omega}}{205 + 200 \cos \omega - 123e^{j\omega} - 120e^{j\omega} \cos \omega} \end{aligned}$$

3.14 Valid autocorrelation matrices

- (a) Valid
- (b) Valid
- (c) Not Valid since $\det R_3 < 0$
- (d) Not Valid since non-Hermitian: $R_4 \neq R_4^T$

3.15 $\mathbf{x}(\zeta)$: a normal random vector with $\rho_{i,j} = 0, i \neq j$

- (a) Consider the (i, j) th component of Γ_x which is given by

$$\gamma_{ij} = \rho_{ij}\sigma_i\sigma_j = \begin{cases} \rho_i\sigma_i^2 & i = j \\ 0 & \text{otherwise} \end{cases}$$

Hence Γ_x has elements only along the main diagonal

- (b) Since the components of $\mathbf{x}(\zeta)$ are mutually Gaussian with $\gamma_{ij} = 0, i \neq j$ then from the property of Gaussian random vectors, the uncorrelatedness also impiles independence.

3.16 Eigenvalues of \mathbf{R}^k : We have to show that $\mathbf{R}^k \mathbf{q} = \lambda^k \mathbf{q}$. Consider

$$\begin{aligned} \underbrace{\mathbf{R} \cdot \mathbf{R} \cdot \mathbf{R} \cdots \mathbf{R} \cdot \mathbf{q}_i}_k &= \underbrace{\mathbf{R} \cdot \mathbf{R} \cdot \mathbf{R} \cdots \mathbf{R}}_{k-1} \cdot \lambda_i \mathbf{q}_i \\ &= \lambda_i \underbrace{\mathbf{R} \cdot \mathbf{R} \cdot \mathbf{R} \cdots \mathbf{R}}_{k-1} \cdot \mathbf{q}_i \quad (\text{since } \lambda_i \text{ is a scalar}) \\ &= \lambda_i \underbrace{\mathbf{R} \cdot \mathbf{R} \cdot \mathbf{R} \cdots \mathbf{R}}_{k-2} \cdot \lambda_i \mathbf{q}_i \\ &= \lambda_i^2 \underbrace{\mathbf{R} \cdot \mathbf{R} \cdot \mathbf{R} \cdots \mathbf{R}}_{k-2} \cdot \mathbf{q}_i \\ &\quad \vdots \\ \mathbf{R}^k q &= \lambda^k \mathbf{q} \end{aligned}$$

3.17 Trace of \mathbf{R} :

$$\begin{aligned} \text{tr}(\mathbf{R}) &= \text{tr}(\mathbf{Q} \Lambda \mathbf{Q}^H) \\ &= \text{tr}(\Lambda \mathbf{Q} \mathbf{Q}^H) \\ &= \text{tr}(\Lambda \mathbf{I}) \\ &= \text{tr}(\Lambda) = \sum \lambda_i \end{aligned}$$

3.18 Determinant of \mathbf{R} :

$$\begin{aligned} \det \mathbf{R} &= |\mathbf{R}| \\ &= |\mathbf{Q} \Lambda \mathbf{Q}^H| \\ &= |\mathbf{Q}| |\Lambda| |\mathbf{Q}^H| \\ &= |\Lambda| = \prod_i \lambda_i \quad (\text{since } \mathbf{Q} \text{ is orthonormal}) \end{aligned}$$

3.19 Relation between the determinants of \mathbf{R} and Γ :

$$\begin{aligned}
 \mathbf{R} &= \Gamma + \boldsymbol{\mu}\boldsymbol{\mu}^T = \mathbf{L}\mathbf{L}^T + \boldsymbol{\mu}\boldsymbol{\mu}^T \\
 \mathbf{L}^{-1}\mathbf{R}\mathbf{L}^{-T} &= \mathbf{I} + \mathbf{L}^{-1}\boldsymbol{\mu}\boldsymbol{\mu}^T\mathbf{L}^{-T} \\
 |\mathbf{L}^{-1}\mathbf{R}\mathbf{L}^{-T}| &= |\mathbf{I} + \mathbf{L}^{-1}\boldsymbol{\mu}\boldsymbol{\mu}^T\mathbf{L}^{-T}| = 1 + \boldsymbol{\mu}^T\mathbf{L}^{-T}\mathbf{L}^{-1}\boldsymbol{\mu} = 1 + \boldsymbol{\mu}^T\Gamma^{-1}\boldsymbol{\mu} \\
 |\mathbf{L}^{-1}|\mathbf{|R|}|\mathbf{L}^{-T}| &= 1 + \boldsymbol{\mu}^T\Gamma^{-1}\boldsymbol{\mu} \\
 |\mathbf{R}| &= |\Gamma|(1 + \boldsymbol{\mu}^T\Gamma^{-1}\boldsymbol{\mu})
 \end{aligned}$$

3.20 $x(n)$: zero-mean WSS process

$\mathbf{x} = [x(0) \ x(2) \ x(3)]$; \mathbf{R}_x : correlation matrix of \mathbf{x} . Hence

$$\mathbf{R}_x = \begin{bmatrix} r_x(0) & r_x(2) & r_x(3) \\ r_x^*(2) & r_x(0) & r_x(2) \\ r_x^*(3) & r_x^*(2) & r_x(0) \end{bmatrix}$$

- (a) From \mathbf{R}_x above, the matrix is Hermitian, that is, $r_{ij} = r_{ji}^*$, Toeplitz since $r_{i,j} = r_{i-j}$, and nonnegative definite since it is correlation matrix of a random vector.
- (b) To determine the correlation matrix of $\bar{\mathbf{x}} = [x(0) \ x(1) \ x(2) \ x(3)]$, we need $r_x(1)$ which is not available.

3.21 Proof of $r_x(0) \geq |r_x(l)|$: Consider

$$\begin{aligned}
 E\{|x(n+l) \pm x(n)|^2\} &= E\{|x(n+l) \pm x(n)][x(n+l) \pm x(n)]^*\| \geq 0 \\
 &= r_x(0) \pm r_x(l) \pm r_x(-l) + r_x(0) \geq 0 \\
 &= 2[r_x(0) \pm r_x(l)] \geq 0
 \end{aligned}$$

which implies $r_x(0) \geq |r_x(l)| \forall l$.

3.22 Nonnegative definiteness of $r_x(l)$: Using the nonnegativeness of $E\{|\alpha_1 x(1) + \dots + \alpha_M x(M)|^2\}$, we have

$$\begin{aligned}
 E\{|\alpha_1 x(1) + \dots + \alpha_M x(M)|^2\} &= E\{[\alpha_1 x(1) + \dots + \alpha_M x(M)][\alpha_1^* x^*(1) + \dots + \alpha_M^* x^*(M)]\} \\
 &= \sum_{l=1}^M \sum_{k=1}^M \alpha_l \alpha_k^* E\{x(l)x^*(k)\} \\
 &= \sum_{l=1}^M \sum_{k=1}^M \alpha_l \alpha_k^* r_x(k-l) = \sum_{l=1}^M \sum_{k=1}^M \alpha_l r_x(k-l) \alpha_k^* \geq 0
 \end{aligned}$$

3.23 AP(1) system: $x(n) = \alpha x(n-1) + w(n)$; $n \geq 0$, $|\alpha| < 1$

$w(n)$: IID($0, \sigma_w^2$) or $r_w(n_1, n_2) = \sigma_w^2 \delta(n_1 - n_2)$

- (a) Autocorrelation $r_x(n_1, n_2)$: First consider the crosscorrelation between $x(n_1)$ and $w(n_2)$, $n_1, n_2 \geq 0$

$$\begin{aligned}
 E\{x(n_1)w^*(n_2)\} &= E\{[\alpha x(n_1-1) + w(n_1)]w^*(n_2)\} \\
 &= \alpha r_{xw}(n_1-1, n_2) + r_w(n_1, n_2), \quad n_1, n_2 \geq 0
 \end{aligned}$$

which results in a partial difference equation in n_1 with n_2 as a parameter

$$r_{xw}(n_1, n_2) = \alpha r_{xw}(n_1-1, n_2) + \sigma_w^2 \delta(n_1 - n_2), \quad n_1, n_2 \geq 0 \quad (6)$$

subject to $r_{xw}(-1, n_2) = E\{x(-1)w^*(n_2)\} = 0$. Taking unilateral z -transform of (6) with respect to n_1 , we have

$$R_{xw}(z_1, n_2) = \alpha z_1^{-1} R_{xw}(z_1, n_2) + \sigma_w^2 z_1^{-n_2}, \quad n_2 \geq 0$$

or

$$R_{xw}(z_1, n_2) = \frac{\sigma_w^2 z_1^{-n_2}}{1 - \alpha z_1^{-1}} \Rightarrow r_{xw}(n_1, n_2) = \sigma_w^2 \alpha^{n_1-n_2} u(n_1 - n_2), \quad n_1, n_2 \geq 0 \quad (7)$$

which as $(n_1, n_2) \rightarrow \infty$ tends to

$$r_{xw}(n_1 - n_2) = r_{xw}(l) = \sigma_w^2 \alpha^l u(l), \quad l = (n_1 - n_2)$$

Now consider the autocorrelation between $x(n_1)$ and $x(n_2)$, $n_1, n_2 \geq 0$

$$\begin{aligned} E\{x(n_1)x^*(n_2)\} &= E\{x(n_1)[\alpha x^*(n_2-1) + w^*(n_2)]\} \\ &= \alpha r_x(n_1, n_2-1) + r_{xw}(n_1, n_2), \quad n_1, n_2 \geq 0 \end{aligned}$$

which results in a partial difference equation in n_2 with n_1 as a parameter

$$r_x(n_1, n_2) = \alpha r_x(n_1, n_2-1) + r_{xw}(n_1, n_2), \quad n_1, n_2 \geq 0 \quad (8)$$

subject to $r_x(n_1, -1) = E\{x(n_1)x^*(-1)\} = 0$. From (7), since $r_{xw}(n_1, n_2) = 0$ for $n_2 > n_1$, first consider the interval $0 \leq n_2 \leq n_1$. Taking unilateral z -transform of (8) with respect to n_2 , we have

$$R_x(n_1, z_2) = \alpha z_2^{-1} R_x(n_1, z_2) + \mathcal{Z}[\sigma_w^2 \alpha^{n_1-n_2} u(n_1 - n_2)]$$

or

$$(1 - \alpha z_2^{-1}) R_x(n_1, z_2) = \sigma_w^2 \left[\frac{\alpha^{n_1}}{1 - \frac{1}{\alpha} z_2^{-1}} - \frac{1}{\alpha} \frac{z_2^{-1} z_2^{-n_1}}{1 - \frac{1}{\alpha} z_2^{-1}} \right]$$

or

$$R_x(n_1, z_2) = \frac{\sigma_w^2 \alpha^{n_1}}{(1 - \alpha z_2^{-1})(1 - \frac{1}{\alpha} z_2^{-1})} - \frac{\frac{\sigma_w^2}{\alpha} z_2^{-n_1-1}}{(1 - \alpha z_2^{-1})(1 - \frac{1}{\alpha} z_2^{-1})} \quad (9)$$

which after inverse z -transformation yields [note that the second term in (9) does not contribute in the interval $0 \leq n_2 \leq n_1$]

$$r_x(n_1, n_2) = \frac{\sigma_w^2}{1 - \alpha^2} \{\alpha^{n_1-n_2} - \alpha^2 \alpha^{n_1+n_2}\}, \quad 0 \leq n_2 \leq n_1 \quad (10)$$

Now consider the interval $0 \leq n_1 < n_2 < \infty$. In this interval, $r_{xw}(n_1, n_2) = 0$ for $n_1 < n_2$, the partial difference equation (8) has no forcing function part, that is, the partial difference equation is

$$r_x(n_1, n_2) = \alpha r_x(n_1, n_2-1), \quad n_2 > n_1 \geq 0 \quad (11)$$

subject to the boundary condition at $n_2 = n_1$ as

$$r_x(n_1, n_1) = \frac{\sigma_w^2}{1 - \alpha^2} \{1 - \alpha^2 \alpha^{2n_1}\}$$

Thus (11) can be easily solved to obtain

$$r_x(n_1, n_2) = \frac{\sigma_w^2}{1-\alpha^2} \{1 - \alpha^2 \alpha^{2n_1}\} \alpha^{n_2-n_1} u(n_2 - n_1), \quad n_1 \geq 0 \quad (12)$$

Finally, combining (10) and (12), we obtain

$$r_x(n_1, n_2) = \begin{cases} \frac{\sigma_w^2}{1-\alpha^2} \{\alpha^{n_1-n_2} - \alpha^2 \alpha^{n_1+n_2}\}, & 0 \leq n_2 \leq n_1 \\ \frac{\sigma_w^2}{1-\alpha^2} \{1 - \alpha^2 \alpha^{2n_1}\} \alpha^{n_2-n_1}, & n_2 > n_1 \geq 0 \end{cases} \quad (13)$$

Clearly, $r_x(n_1, n_2)$ is not stationary.

- (b) To probe the asymptotic behavior of $r_x(n_1, n_2)$, let $n_1, n_2 \rightarrow \infty$ keeping $n_1 - n_2 \triangleq l$ constant. Then from (13), we obtain

$$r_x(n_1, n_2) = \begin{cases} \frac{\sigma_w^2}{1-\alpha^2} \alpha^{n_1-n_2}, & 0 \leq n_2 \leq n_1 \Rightarrow l \geq 0 \\ \frac{\sigma_w^2}{1-\alpha^2} \alpha^{n_2-n_1}, & n_2 > n_1 \geq 0 \Rightarrow l < 0 \end{cases}$$

or

$$r_x(n_1, n_1 - l) = \begin{cases} \frac{\sigma_w^2}{1-\alpha^2} \alpha^l, & l \geq 0 \\ \frac{\sigma_w^2}{1-\alpha^2} \alpha^{-l}, & l < 0 \end{cases} = \frac{\sigma_w^2}{1-\alpha^2} \alpha^{|l|} \quad (14)$$

Thus $r_x(n_1, n_2)$ is asymptotically stationary.

3.24 \mathbf{x} is a random vector with mean $\boldsymbol{\mu}_x$ and autocorrelation \mathbf{R}_x .

- (a) Consider $\mathbf{y} = \mathbf{Q}^H \mathbf{x}$ where $\mathbf{R}_x \mathbf{Q} = \mathbf{Q} \Lambda$ and \mathbf{Q} is orthonormal. Then

$$\begin{aligned} \mathbf{R}_y &= E\{\mathbf{y}\mathbf{y}^H\} = E\{\mathbf{Q}^H \mathbf{x} \mathbf{x}^H \mathbf{Q}\} \\ &= \mathbf{Q}^H E\{\mathbf{x} \mathbf{x}^H\} \mathbf{Q} \\ &= \mathbf{Q}^H \mathbf{R}_x \mathbf{Q} = \Lambda \end{aligned}$$

Thus \mathbf{y} is an orthogonal component vector.

- (b) Geometric interpretation

3.25 A Gaussian random vector \mathbf{x} with $\boldsymbol{\mu}_x = [1 \ 2]^T$ and $\Gamma_x = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$. Contour plot of the density function is shown in Figure 3.25.

3.26 Random process $x(n) = ax(n-1) + w(n)$, $w(n) \sim \text{WN}(0, \sigma_w^2)$

- (a) Autocorrelation matrix \mathbf{R}_x : Consider

$$\begin{aligned} r_x(0) &= E\{x(n)x(n)\} = E\{(ax(n-1) + w(n-1))^2\} \\ &= a^2 r_x(0) + \sigma_w^2 = \frac{\sigma_w^2}{1-a^2} \end{aligned}$$

Also

$$\begin{aligned} r_x(1) &= E\{x(n)x(n+1)\} = E\{x(n)(ax(n) + w(n))\} \\ &= ar_x(0) \end{aligned}$$

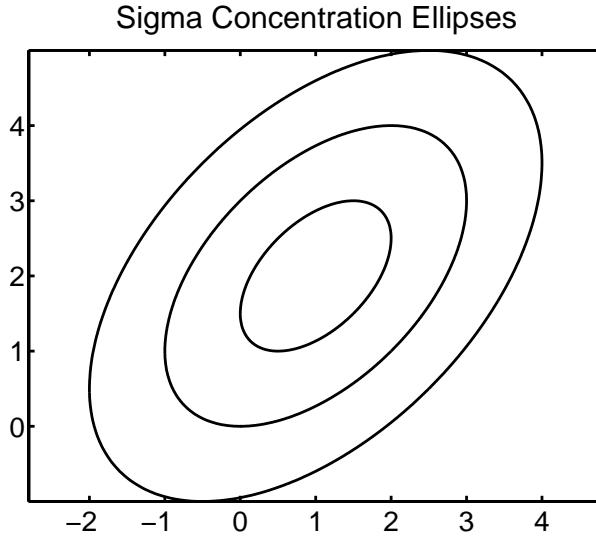


Figure 3.25: Sigma Concentration Ellipses

Similarly

$$r_x(M) = a^M r_x(0)$$

which gives the matrix

$$\mathbf{R}_x = \frac{\sigma_w^2}{1-a^2} \begin{bmatrix} 1 & a & \cdots & a^{M-1} \\ a & 1 & \cdots & a^{M-2} \\ \vdots & \vdots & \ddots & \vdots \\ a^{M-1} & a^{M-2} & \cdots & 1 \end{bmatrix}$$

that is symmetric Toeplitz.

(b) We will show that $\mathbf{R}_x \cdot \mathbf{R}_x^{-1} = \mathbf{I}$. Consider

$$\begin{aligned} \mathbf{R}_x \cdot \mathbf{R}_x^{-1} &= \frac{\sigma_w^2}{1-a^2} \begin{bmatrix} 1 & a & \cdots & a^{M-1} \\ a & 1 & \cdots & a^{M-2} \\ \vdots & \vdots & \ddots & \vdots \\ a^{M-1} & a^{M-2} & \cdots & 1 \end{bmatrix} \begin{bmatrix} 1 & -a & 0 & \cdots & 0 \\ -a & 1+a^2 & -a & \cdots & 0 \\ 0 & -a & 1+a^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 1+a^2 & -a \\ 0 & 0 & \cdots & -a & 1 \end{bmatrix} \\ &= \frac{1}{1-a^2} \begin{bmatrix} 1-a^2 & 0 & \cdots & 0 \\ 0 & 1-a^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1-a^2 \end{bmatrix} = \mathbf{I} \end{aligned}$$

(c) Let

$$\mathbf{L}_x = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ -a & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & -a & 1 \end{bmatrix}$$

Then

$$\begin{aligned}
 \mathbf{L}_x^T \mathbf{R}_x \mathbf{L}_x &= \frac{\sigma_w^2}{1-a^2} \begin{bmatrix} 1 & -a & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \ddots & 1 & -a \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & a & \cdots & a^{M-1} \\ a & 1 & \cdots & a^{M-2} \\ \vdots & \vdots & \ddots & \vdots \\ a^{M-2} & a^{M-3} & \cdots & a \\ a^{M-1} & a^{M-2} & \cdots & 1 \end{bmatrix} \\
 &\quad \times \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ -a & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & -a & 1 \end{bmatrix} \\
 &= \sigma_w^2 \mathbf{I}
 \end{aligned}$$

(d) Plots of the DKLT and the DFT is shown in Figure 3.26d. The PSD points approach the eigenvalues as

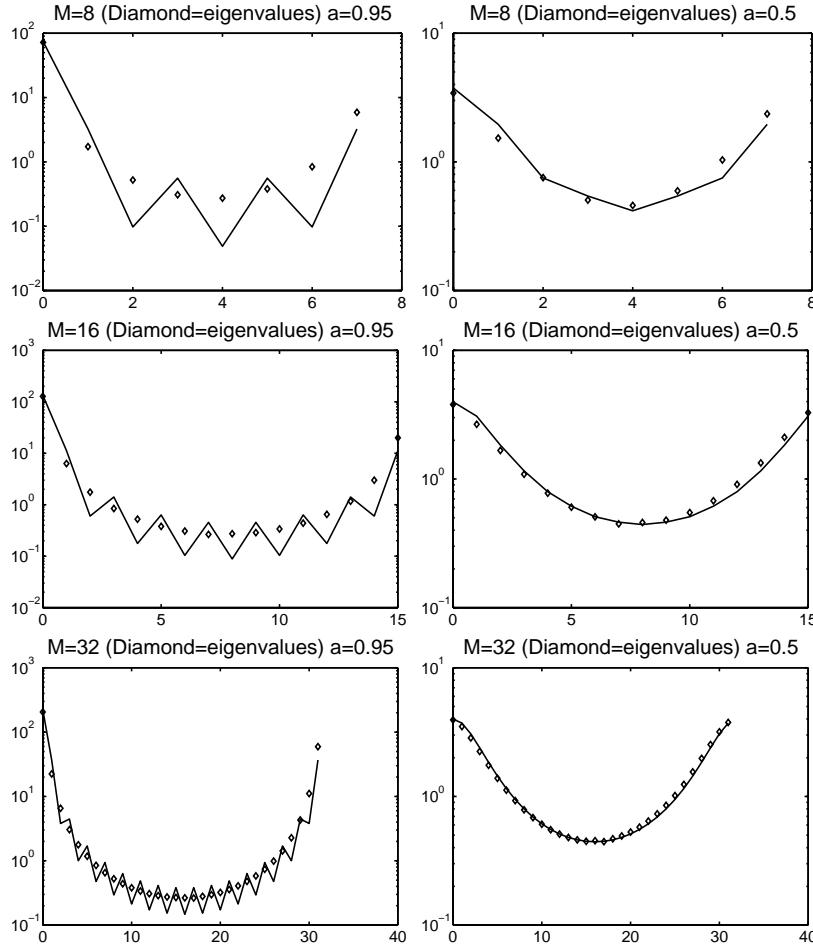


Figure 3.26d: Eigenvalue and PSD

the size of the correlation matrix increases.

- (e) Plots of the eigenvectors are shown in Figure 3.26e.
(f) Plots of the eigenvectors are shown in Figure 3.26f.

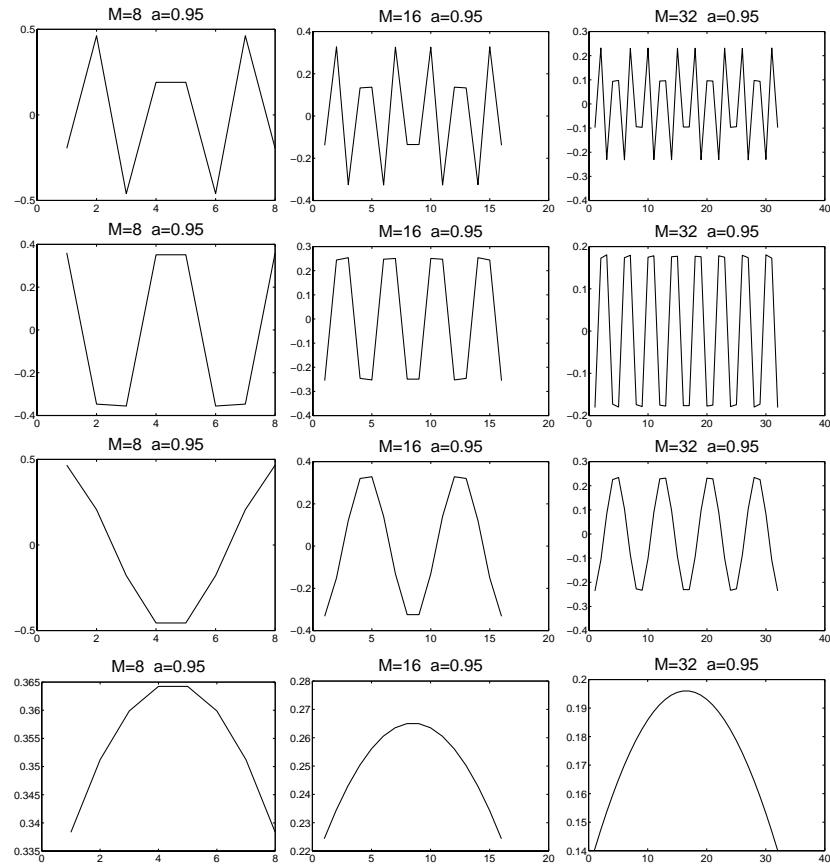


Figure 3.26e: Eigenvectors

3.27 Three different innovations representations:

Eigen-decomposition

$$\mathbf{R}_x = \mathbf{Q}_x \Lambda_x^H \mathbf{Q}_x = \begin{bmatrix} 0.7071 & 0.7071 \\ -0.7071 & 0.7071 \end{bmatrix} \begin{bmatrix} \frac{3}{4} & 0 \\ 0 & \frac{5}{4} \end{bmatrix} \begin{bmatrix} 0.7071 & -0.7071 \\ 0.7071 & 0.7071 \end{bmatrix}$$

$$\mathbf{w} \sim N(0, \Lambda_x), \quad \mathbf{x} = \mathbf{Q}_x \mathbf{w}$$

LDU Triangularization

$$\mathbf{R}_x = \mathbf{L}_x \mathbf{D}_L^{(x)} \mathbf{L}_x^H = \begin{bmatrix} 1 & 0 \\ \frac{1}{4} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0.9375 \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{4} \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{w} \sim N(0, \mathbf{D}_L^{(x)}), \quad \mathbf{x} = \mathbf{L}_x \mathbf{w}$$

UDL Triangularization

$$\mathbf{R}_x = \mathbf{U}_x \mathbf{D}_U^{(x)} \mathbf{U}_x^H = \begin{bmatrix} \frac{1}{4} & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0.9375 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & \frac{1}{4} \end{bmatrix}$$

$$\mathbf{w} \sim N(0, \mathbf{D}_U^{(x)}), \quad \mathbf{x} = \mathbf{U}_x \mathbf{w}$$

3.28 Eigenvalues and Eigenvectors of a MA(1) process.

- (a) Given that $x(n) = w(n) + b w(n-1)$ with $w(n) \sim WN(0, \sigma_w^2)$, the complex PSD is

$$R_x(z) = \sigma_w^2 (1 + bz^{-1})(1 + bz) = \sigma_w^2 \{bz + (1 + b^2) + bz^{-1}\}$$

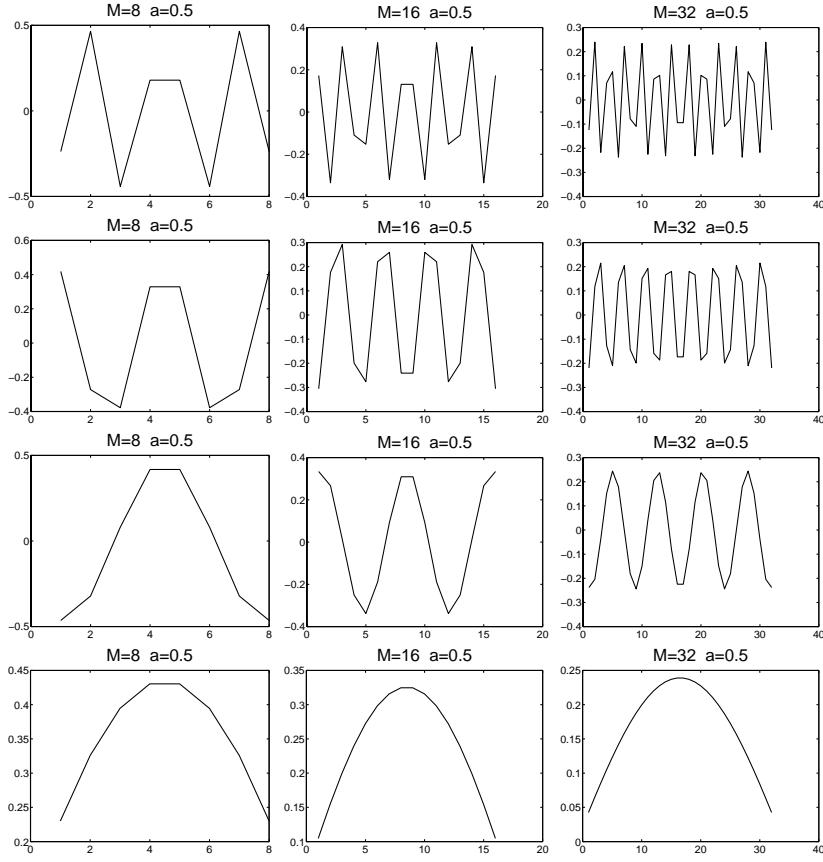


Figure 3.26f: Eigenvectors

the real PSD is

$$R_x(e^{j\omega}) = \sigma_w^2 \{(1 + b^2) + 2b \cos \omega\}$$

and the autocorrelation sequence is

$$r_x(l) = \sigma_w^2 \{b \delta(l+1) + (1 + b^2) + b \delta(l-1)\}$$

Hence the $M \times M$ correlation matrix is

$$\mathbf{R}_x = \sigma_w^2 \begin{bmatrix} 1 + b^2 & b & \cdots & 0 & 0 \\ b & 1 + b^2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 + b^2 & b \\ 0 & 0 & \cdots & b & 1 + b^2 \end{bmatrix}$$

Consider $M = 1$: Then

$$\mathbf{R}_x = \sigma_w^2 (1 + b^2)$$

with

$$\lambda = \sigma_w^2 (1 + b^2) = \sigma_w^2 \{1 + b^2 + 2b \cos(\pi/2)\} = R_x(e^{j\omega_1}), \quad \omega_1 = \pi/2 = \frac{\pi(1)}{1+1}$$

$$q_n^{(1)} = 1 = \sin \pi n/2 = \sin \omega_1 n; \quad n = 1, \quad \omega_1 = \pi/2 = \frac{\pi(1)}{1+1}$$

Consider $M = 2$: Then

$$\mathbf{R}_x = \sigma_w^2 \begin{bmatrix} (1+b^2) & b \\ b & 1+b^2 \end{bmatrix}$$

The eigenvalues are

$$\begin{aligned}\lambda_1 &= \sigma_w^2 (1 + b^2 + b) = \sigma_w^2 \{1 + b^2 + 2b \cos(\pi/3)\} = R_x(e^{j\omega_1}), \quad \omega_1 = \pi/3 = \frac{\pi(1)}{2+1} \\ \lambda_2 &= \sigma_w^2 (1 + b^2 - b) = \sigma_w^2 \{1 + b^2 + 2b \cos(2\pi/3)\} = R_x(e^{j\omega_2}), \quad \omega_2 = 2\pi/3 = \frac{\pi(2)}{2+1}\end{aligned}$$

with unnormalized eigenvectors

$$\begin{aligned}q_n^{(1)} &= \frac{\sqrt{3}}{2} [1, 1] = \sin \pi n/3 = \sin \omega_1 n; \quad n = 1, 2; \quad \omega_1 = \pi/3 = \frac{\pi(1)}{2+1} \\ q_n^{(2)} &= \frac{\sqrt{3}}{2} [1, -1] = \sin 2\pi n/3 = \sin \omega_2 n; \quad n = 1, 2; \quad \omega_2 = 2\pi/3 = \frac{\pi(2)}{2+1}\end{aligned}$$

Consider $M = 3$: Then

$$\mathbf{R}_x = \sigma_w^2 \begin{bmatrix} 1+b^2 & b & 0 \\ b & 1+b^2 & b \\ 0 & b & 1+b^2 \end{bmatrix}$$

The eigenvalues are

$$\begin{aligned}\lambda_1 &= \sigma_w^2 (1 + b^2 + \sqrt{2}b) = \sigma_w^2 \{1 + b^2 + 2b \cos(\pi/4)\} = R_x(e^{j\omega_1}), \quad \omega_1 = \pi/4 = \frac{\pi(1)}{3+1} \\ \lambda_2 &= \sigma_w^2 (1 + b^2) = \sigma_w^2 \{1 + b^2 + 2b \cos(2\pi/4)\} = R_x(e^{j\omega_2}), \quad \omega_2 = 2\pi/4 = \frac{\pi(2)}{3+1} \\ \lambda_3 &= \sigma_w^2 (1 + b^2 - \sqrt{2}b) = \sigma_w^2 \{1 + b^2 + 2b \cos(3\pi/4)\} = R_x(e^{j\omega_3}), \quad \omega_3 = 3\pi/4 = \frac{\pi(3)}{3+1}\end{aligned}$$

with unnormalized eigenvectors:

$$\begin{aligned}q_n^{(1)} &= \frac{1}{2}\sqrt{2} \begin{bmatrix} 1 \\ \sqrt{2} \\ 1 \end{bmatrix} = \begin{bmatrix} \sin \pi/4 \\ \sin 2\pi/4 \\ \sin 3\pi/4 \end{bmatrix} = \{\sin \omega_1 n\}_{n=1}^3 \\ q_n^{(2)} &= \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} \sin 2\pi/4 \\ \sin 4\pi/4 \\ \sin 6\pi/4 \end{bmatrix} = \{\sin \omega_2 n\}_{n=1}^3 \\ q_n^{(3)} &= \frac{1}{2}\sqrt{2} \begin{bmatrix} 1 \\ -\sqrt{2} \\ 1 \end{bmatrix} = \begin{bmatrix} \sin 3\pi/4 \\ \sin 6\pi/4 \\ \sin 9\pi/4 \end{bmatrix} = \{\sin \omega_3 n\}_{n=1}^3\end{aligned}$$

Therefore, for a general M , we have

$$\lambda_k = R_x(e^{j\omega_k}); \quad \omega_k = \pi k/(M+1), \quad k = 1, 2, \dots, M$$

and

$$q_n^{(k)} = \sin \omega_k n; \quad \omega_k = \pi k/(M+1), \quad k = 1, 2, \dots, M, \quad n = 1, 2, \dots, M$$

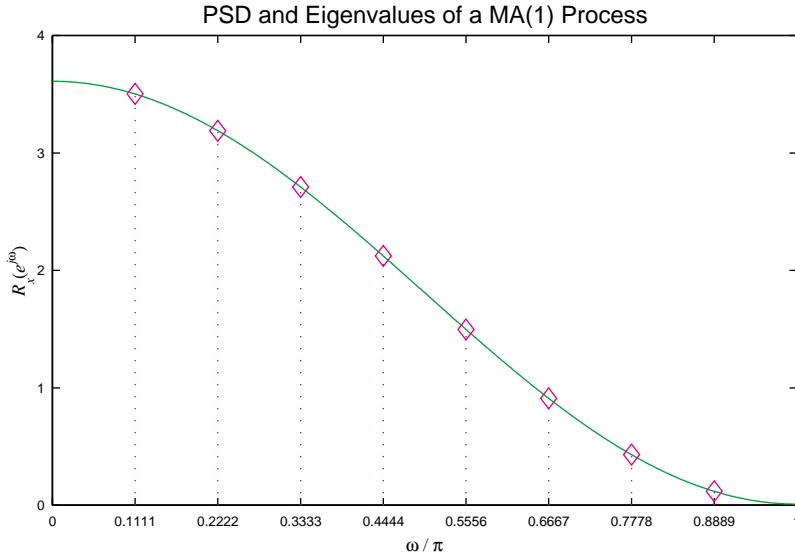


Figure 3.28: Plot of the PSD and eigenvalues of MA(1) process

(b) Assume $M = 8$, $b = 0.9$, and $\sigma_w^2 = 1$. The Matlab script is shown below and the plot is shown in Figure 3.28.

```

var_w = 1; M = 8; b = 0.9;
r = var_w*[(1+b*b),b,zeros(1,M-2)];
R = toeplitz(r);
Lambda = eig(R); Lambda = flipud(sort(Lambda));
k = 1:M;
omg_k = pi*k/(M+1);
omg = (0:500)*pi/500;
PSD = var_w*((1+b*b)+2*b*cos(omg));

Hf_1 = figure('units','SCRUN','position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',[0,0,5,3]);
set(Hf_1,'NumberTitle','off','Name','Pr0328b');

plot(omg/pi,PSD,'g'); hold on;
stem(omg_k/pi,Lambda,'md:');
xlabel('\omega / \pi','fontsize',label_fontsize,'fontname','times');
ylabel('{\it R_x}(\{e^{j\omega}\})','fontsize',label_fontsize,'fontname','times');
title('PSD and Eigenvalues of a MA(1) Process','FontSize',title_fontsize);
set(gca,'xtick',[0,omg_k/pi,1],'ytick',[0:1:4]); Hold off;

```

3.29 Random process: $x(n) = w(n) + b w(n - 1)$.

(a) DKLT for $M = 3$. The autocorrelation matrix is given by

$$\mathbf{R}_x = \sigma_w^2 \frac{b}{\alpha} \begin{bmatrix} 1 & \alpha & 0 \\ \alpha & 1 & \alpha \\ 0 & \alpha & 1 \end{bmatrix}, \quad \text{where } \alpha = \frac{b}{1+b^2}$$

Now to determine the eigenanalysis, consider

$$\begin{aligned}\det(\mathbf{R}_x - \lambda \mathbf{I}) &= 0 = \det \begin{bmatrix} 1-\lambda & \alpha & 0 \\ \alpha & 1-\lambda & \alpha \\ 0 & \alpha & 1-\lambda \end{bmatrix} \\ &= (1-\lambda)(1-2\alpha^2-2\lambda+\lambda^2) = 0\end{aligned}$$

which gives

$$\lambda_1 = 1, \quad \lambda_2 = \frac{1 + \sqrt{2}\alpha}{1 - 2\alpha^2}, \quad \lambda_3 = \frac{1 - \sqrt{2}\alpha}{1 - 2\alpha^2}$$

- (b) The variances of the DKLT coefficients are given by the eigenvalues which from above in the descending order are

$$1, \quad \frac{1}{1 - \sqrt{2}\alpha}, \quad \frac{1}{1 + \sqrt{2}\alpha}$$

- 3.30** Let $x(n)$ be a stationary random process with mean μ_x and covariance $\gamma_x(l)$. Let $\hat{\mu}_x$ be the sample mean estimator of μ_x given by

$$\hat{\mu}_x = \frac{1}{N} \sum_{n=0}^{N-1} x(n)$$

- (a) The mean of $\hat{\mu}_x$ is given by

$$E[\hat{\mu}_x] = \frac{1}{N} \sum_{n=0}^{N-1} E[x(n)] = \frac{1}{N} \sum_{n=0}^{N-1} \mu_x = \mu_x$$

and the variance of $\hat{\mu}_x$ is given by

$$\begin{aligned}\text{var}(\hat{\mu}_x) &= E[|\hat{\mu}_x - \mu_x|^2] = E[(\hat{\mu}_x - \mu_x)(\hat{\mu}_x^* - \mu_x^*)] \\ &= E\left[\left(\frac{1}{N} \sum_{n=0}^{N-1} x(n) - \mu_x\right)\left(\frac{1}{N} \sum_{m=0}^{N-1} x^*(m) - \mu_x^*\right)\right] \\ &= \frac{1}{N^2} \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} E[x(n)x^*(m)] - \frac{1}{N} \sum_{n=0}^{N-1} E[x(n)]\mu_x^* - \frac{1}{N} \sum_{m=0}^{N-1} E[x^*(m)]\mu_x + \mu_x\mu_x^* \\ &= \frac{1}{N^2} \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} E[x(n)x^*(m)] - \frac{1}{N} \sum_{n=0}^{N-1} |\mu_x|^2 - \frac{1}{N} \sum_{m=0}^{N-1} |\mu_x|^2 + |\mu_x|^2 \\ &= \frac{1}{N^2} \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} E[x(n)x^*(m)] - |\mu_x|^2 = \frac{1}{N^2} \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} E[x(n)x^*(m)] - \frac{1}{N^2} \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} |\mu_x|^2 \\ &= \frac{1}{N^2} \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} \{E[x(n)x^*(m)] - |\mu_x|^2\} = \frac{1}{N^2} \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} \gamma_x(n-m)\end{aligned}$$

Thus

$$\text{var}(\hat{\mu}_x) = \frac{1}{N^2} \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} \gamma_x(n-m) \quad (15)$$

Using change of variables $n - m = l$, we can reduce double summation into a single summation

$$\text{var}(\hat{\mu}_x) = \frac{1}{N} \sum_{l=-N}^N \left(1 - \frac{|l|}{N}\right) \gamma_x(l)$$

Since $\left(1 - \frac{|l|}{N}\right) \leq 1$,

$$\text{var}(\hat{\mu}_x) \leq \frac{1}{N} \sum_{l=-N}^N \gamma_x(l) \leq \frac{1}{N} \sum_{l=-N}^N |\gamma_x(l)|$$

(b) From (15)

$$\text{var}(\hat{\mu}_x) = \frac{\sigma_x^2}{N} \sum_{l=-N}^N \left(1 - \frac{|l|}{N}\right) \frac{\gamma_x(l)}{\sigma_x^2} = \frac{\sigma_x^2}{N} \sum_{l=-N}^N \left(1 - \frac{|l|}{N}\right) \rho_x(l), \quad \rho_x(l) \triangleq \frac{\gamma_x(l)}{\sigma_x^2}$$

Since for a real-valued random process $\rho_x(l) = \rho_x(-l)$ and using symmetry of $|l|$, we obtain

$$\begin{aligned} \text{var}(\hat{\mu}_x) &= \frac{\sigma_x^2}{N} \left[\sum_{l=-N}^{-1} \left(1 - \frac{-l}{N}\right) \rho_x(-l) + 1 + \sum_{l=1}^N \left(1 - \frac{l}{N}\right) \rho_x(l) \right] \\ &= \frac{\sigma_x^2}{N} \left[1 + 2 \sum_{l=1}^N \left(1 - \frac{l}{N}\right) \rho_x(l) \right] \\ &= \frac{\sigma_x^2}{N} [1 + \Delta_N(\rho_x)], \quad \Delta_N(\rho_x) \triangleq 2 \sum_{l=1}^N \left(1 - \frac{l}{N}\right) \rho_x(l) \end{aligned}$$

(c) When $x(n) \sim \text{WN}(\mu_x, \sigma_x^2)$, then $\rho_x(l) = \delta(l)$ or $\Delta_N(\rho_x) = 0$. Hence $\text{var}(\hat{\mu}_x) = \frac{\sigma_x^2}{N}$.

3.31 Let $x(n)$ be a stationary random process with mean μ_x and covariance $\gamma_x(l)$. Let $\hat{\sigma}_x^2$ be the sample variance estimator of σ_x^2 given by

$$\hat{\sigma}_x^2 = \frac{1}{N} \sum_{n=0}^{N-1} |x(n) - \hat{\mu}_x|^2$$

(a) The mean of $\hat{\sigma}_x^2$ is given by

$$\begin{aligned} E[\hat{\sigma}_x^2] &= \frac{1}{N} \sum_{n=0}^{N-1} E[|x(n) - \hat{\mu}_x|^2] = \frac{1}{N} \sum_{n=0}^{N-1} E[|x(n) - \mu_x - \hat{\mu}_x + \mu_x|^2] \\ &= \frac{1}{N} \sum_{n=0}^{N-1} E[|(x(n) - \mu_x) - (\hat{\mu}_x - \mu_x)|^2] \\ &= \frac{1}{N} \sum_{n=0}^{N-1} E[|x(n) - \mu_x|^2 - (x(n) - \mu_x)(\hat{\mu}_x^* - \mu_x^*) - (x^*(n) - \mu_x^*)(\hat{\mu}_x - \mu_x) + |\hat{\mu}_x - \mu_x|^2] \\ &= \sigma_x^2 + \text{var}(\hat{\mu}_x) - \frac{1}{N} \sum_{n=0}^{N-1} E[(x(n) - \mu_x)(\hat{\mu}_x^* - \mu_x^*) + (x^*(n) - \mu_x^*)(\hat{\mu}_x - \mu_x)] \end{aligned} \quad (16)$$

Consider

$$\begin{aligned} E[(x(n) - \mu_x)(\hat{\mu}_x^* - \mu_x^*)] &= E[x(n)\hat{\mu}_x^* - x(n)\mu_x^* - \mu_x\hat{\mu}_x^* + |\mu_x|^2] \\ &= E[x(n)\hat{\mu}_x^*] - |\mu_x|^2 - |\mu_x|^2 + |\mu_x|^2 = E[x(n)\hat{\mu}_x^*] - |\mu_x|^2 \end{aligned} \quad (17)$$

Similarly,

$$E[(x^*(n) - \mu_x^*)(\hat{\mu}_x - \mu_x)] = E[x^*(n)\hat{\mu}_x] - |\mu_x|^2 \quad (18)$$

Consider

$$\begin{aligned} E[x(n)\hat{\mu}_x^*] &= \frac{1}{N}E\left[x(n)\sum_{m=0}^{N-1}x^*(m)\right] = \frac{1}{N}\sum_{m=0}^{N-1}E[x(n)x^*(m)] \\ &= \frac{1}{N}\sum_{m=0}^{N-1}\left[\gamma_x(n-m) + |\mu_x|^2\right] = \frac{1}{N}\sum_{m=0}^{N-1}\gamma_x(n-m) + |\mu_x|^2 \end{aligned} \quad (19)$$

and

$$E[x^*(n)\hat{\mu}_x] = \frac{1}{N}\sum_{m=0}^{N-1}\left[\gamma_x^*(n-m) + |\mu_x|^2\right] = \frac{1}{N}\sum_{m=0}^{N-1}\gamma_x^*(n-m) + |\mu_x|^2 \quad (20)$$

Substituting (17), (18), (19), and (20) in (16), we obtain

$$E[\hat{\sigma}_x^2] = \sigma_x^2 + \text{var}(\hat{\mu}_x) - \frac{1}{N^2}\sum_{n=0}^{N-1}\sum_{m=0}^{N-1}\gamma_x(n-m) - \frac{1}{N^2}\sum_{n=0}^{N-1}\sum_{m=0}^{N-1}\gamma_x^*(n-m) \quad (21)$$

But from (15) in Problem 3.30

$$\text{var}(\hat{\mu}_x) = \frac{1}{N^2}\sum_{n=0}^{N-1}\sum_{m=0}^{N-1}\gamma_x(n-m) = \frac{1}{N^2}\sum_{n=0}^{N-1}\sum_{m=0}^{N-1}\gamma_x^*(n-m) \quad (22)$$

Substituting (22) in (21), we obtain

$$E[\hat{\sigma}_x^2] = \sigma_x^2 - \text{var}(\hat{\mu}_x) = \sigma_x^2 - \frac{1}{N}\sum_{l=-N}^N\left(1 - \frac{|l|}{N}\right)\gamma_x(l)$$

The last equality follows from Problem 3.30.

(b) When $x(n) \sim \text{WN}(\mu_x, \sigma_x^2)$, then $\gamma_x(l) = \sigma_x^2\delta(l)$. Hence $E[\hat{\sigma}_x^2] = \sigma_x^2 - \frac{\sigma_x^2}{N} = (N-1)\sigma_x^2/N$.

3.32 Cauchy distribution with mean μ : $f_x(x) = \frac{1}{\pi} \frac{1}{1+(x-\mu)^2}$. The mean estimator is $\hat{\mu}(\xi) = \frac{1}{N} \sum_{k=1}^N x_k(\xi)$. Then the mean of $\hat{\mu}(\xi) = \mu$ and

$$\begin{aligned} \text{var}(\hat{\mu}) &= E\{[\hat{\mu} - \mu]^2\} = E\{\hat{\mu}^2 - 2\mu\hat{\mu} + \mu^2\} \\ &= E\{\hat{\mu}^2\} - 2\mu E\{\hat{\mu}\} + \mu^2 = E\left\{\left(\frac{1}{N}\sum_{k=1}^N x_k(\xi)\right)^2\right\} - \mu^2 \\ &= \frac{1}{N^2}\sum_{k=1}^N E\{x_k(\xi)^2\} - \mu^2 \\ &= \frac{E\{x(\xi)^2\}}{N} - \mu^2 \neq 0 \text{ as } N \rightarrow \infty \end{aligned}$$

Therefore, $\hat{\mu}(\xi)$ is not a consistent estimator.

Chapter 4

Linear Signal Models

- 4.1** Show that a second-order pole p_i contributes the term $np_i^n u(n)$ and a third-order pole the terms $np_i^n u(n) + n^2 p_i^n u(n)$ to the impulse response of a causal PZ model.

The general expression for a PZ model with double poles in the z-domain is

$$H(z) = \frac{D(z)}{A(z)} = d_0 \frac{\prod_{k=1}^Q (1 - z_k z^{-1})}{(1 - p_0 z^{-1})^2 \prod_{k=1}^{P-2} (1 - p_k z^{-1})}; \quad Q < P, |z| > \max(|p_k|), \max|p_k| < 1$$

Using partial fraction expansion on this expression and only keeping the double term simplifies to

$$H(z) = \frac{A_1 z^{-1}}{(1 - p_0 z^{-1})^2} + \frac{A_2}{(1 - p_0 z^{-1})} + \frac{B(z)}{\prod_{k=1}^{P-2} (1 - p_k z^{-1})}$$

where $B(z)$ is a polynomial of order less than $P - 2$. After taking inverse z -transform,

$$h(n) = A_1 n p_0^n u(n) + A_2 p_0^n u(n) + \mathcal{Z}^{-1} \left[\frac{B(z)}{\prod_{k=1}^{P-2} (1 - p_k z^{-1})} \right]$$

Similarly, for a third-order pole, we can write

$$H(z) = \frac{A_1 z^{-2}}{(1 - p_0 z^{-1})^3} \frac{A_2 z^{-1}}{(1 - p_0 z^{-1})^2} + \frac{A_3}{(1 - p_0 z^{-1})} + \frac{B(z)}{\prod_{k=1}^{P-3} (1 - p_k z^{-1})}$$

which upon inverse z -transformation yields

$$h(n) = A_1 n^2 p_0^n u(n) + A_2 n p_0^n u(n) + A_3 p_0^n u(n) + \mathcal{Z}^{-1} \left[\frac{B(z)}{\prod_{k=1}^{P-2} (1 - p_k z^{-1})} \right]$$

- 4.2** A zero-mean random sequence $x(n)$ with PSD:

$$R_x(e^{j\omega}) = \frac{5 + 3 \cos \omega}{17 + 8 \cos \omega}$$

- (a) The innovations representations of the process $x(n)$ can be found by transforming $x(n)$ into the z-domain

$$R_x(z) = \frac{z(10 + 3(z + z^{-1}))}{z(34 + 8(z + z^{-1}))} = \frac{3z^2 + 10z + 3}{8z^2 + 34z + 8} = \frac{9(1 + \frac{1}{3}z)(1 + \frac{1}{3}z^{-1})}{32(1 + \frac{1}{4}z)(1 + \frac{1}{4}z^{-1})}$$

which can be written as

$$R_x(z) = \sigma_w^2 H_{\min}(z) H_{\min}^*(1/z^*) = \frac{9}{32} \frac{(1 + \frac{1}{3}z)(1 + \frac{1}{3}z^{-1})}{(1 + \frac{1}{4}z)(1 + \frac{1}{4}z^{-1})}$$

Clearly, $H_{\min}(z)$ is equal to

$$H_{\min}(z) = \frac{1 + \frac{1}{3}z^{-1}}{1 + \frac{1}{4}z^{-1}}$$

and the resulting innovations process is

$$x(n) = -\frac{1}{4}x(n-1) + w(n) + \frac{1}{3}w(n-1)$$

with $\sigma_w^2 = \frac{9}{32} = 0.28125$

(b) The autocorrelation sequence $r_x(l)$ can be found using the factorization of $R_x(z)$:

$$\begin{aligned} R_x(z) &= \frac{9}{32} \frac{(1 + \frac{1}{3}z)(1 + \frac{1}{3}z^{-1})}{(1 + \frac{1}{4}z)(1 + \frac{1}{4}z^{-1})} \\ &= \frac{3}{8} - \frac{11}{120(1 + \frac{1}{4}z^{-1})} + \frac{11}{120(1 + 4z^{-1})}, \quad \frac{1}{4} < |z| < 4 \end{aligned}$$

Hence

$$r_x(l) = \frac{3}{8}\delta(l) - \frac{11}{120} \left(-\frac{1}{4}\right)^{|l|}$$

4.3 A sequence with an autocorrelation $r_x(l) = (\frac{1}{2})^{|l|} + (-\frac{1}{2})^{|l|}$.

(a) From the autocorrelation, we can see that $x(n)$ has no dependence with $x(n-1)$ but does have a dependence on $x(n-2)$. We can assume $x(n)$ to have the following form

$$x(n) = ax(n-2) + bw(n)$$

Solving for a and b

$$\begin{aligned} r_x(0) &= a^2 E\{x(n-2)^2\} + b^2 E\{w(n)^2\} + 2abE\{x(n-2)w(n)\} \\ &= a^2 r_x(0) + b^2 \\ r_x(2) &= aE\{x(n-2)\} + bE\{w(n)x(n-2)\} = ar_x(0) \end{aligned}$$

which results in $a = \frac{1}{4}$ and $b = \frac{15}{8}$. Checking to see that $x(n)$ has no dependence on $x(n-1)$

$$r_x(1) = E\{x(n)x(n-1)\} = ar_x(1) = 0$$

Therefore,

$$x(n) = \frac{1}{4}x(n-2) + \frac{15}{8}w(n)$$

(b) A pdf of the realization of the process is shown in Figure 4.3b. The dotted line represents a true Gaussian pdf.

(c) The true and estimated autocorrelation are shown in Figure 4.3c.

4.4 Computing the autocorrelations directly for the two systems, where $w(n) \sim \text{WGN}(0,1)$

(a) $x_1(n) = w(n) + 0.3w(n-1) - 0.4w(n-2)$

$$\begin{aligned} r_x(l) &= E\{[w(n) + 0.3w(n-1) - 0.4w(n-2)][w(n+l) + 0.3w(n+l-1) - 0.4w(n+l-2)]\} \\ &= \delta(l) + 0.3\delta(l-1) - 0.4\delta(l-2) \\ &\quad + 0.3\delta(l-1) + (0.3)^2\delta(l) - (0.4)(0.3)\delta(l-1) \\ &\quad + (-0.4)\delta(l-2) + (-0.4)(0.3)\delta(l-1) + (-0.4)^2\delta(l) \\ &= 1.25\delta(l) + 0.36\delta(l-1) - 0.8\delta(l-2) \end{aligned}$$

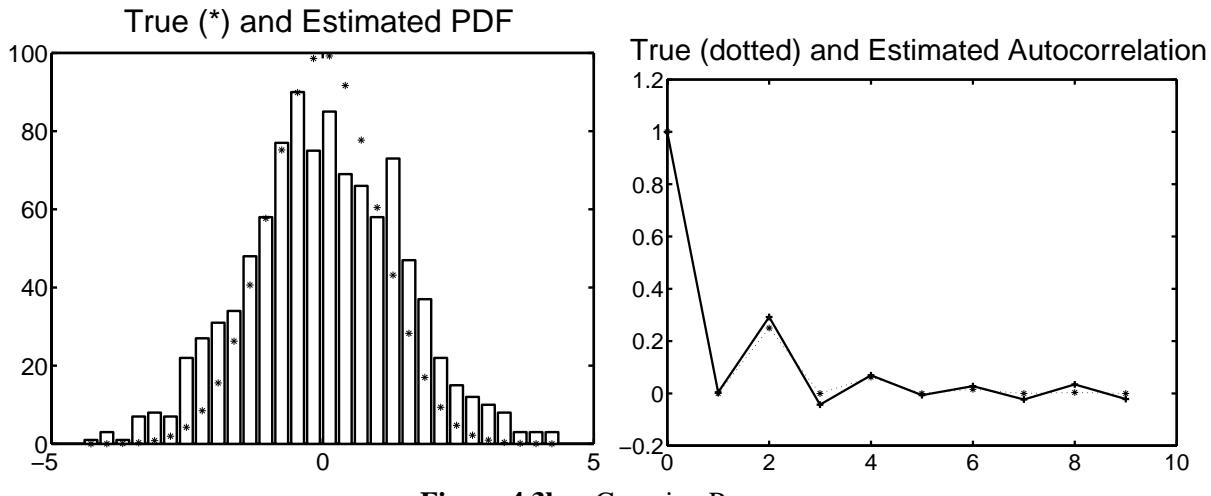


Figure 4.3bc: Gaussian Process

$$(b) x_2(n) = w(n) - 1.2w(n-1) - 1.6w(n-2)$$

$$\begin{aligned} r_x(l) &= E\{[w(n) - 1.2w(n-1) - 1.6w(n-2)][w(n+l) - 1.2w(n+l-1) - 1.6w(n+l-2)]\} \\ &= \delta(l) - 1.2\delta(l-1) - 1.6\delta(l-2) \\ &\quad - 1.2\delta(l-1) + (-1.2)^2\delta(l) + (-1.2)(-1.6)\delta(l-1) \\ &\quad + (-1.6)\delta(l-2) + (-1.6)(-1.2)\delta(l-1) + (-1.6)^2\delta(l) \\ &= 5\delta(l) + 1.44\delta(l-1) - 3.2\delta(l-2) \end{aligned}$$

Comparing these two autocorrelation, they both have the same ratio of dependence on the previous two inputs.

4.5 Investigate how well the all-zero systems approximate the single pole system

The single pole system with a pole at a has a system function of

$$H(z) = \frac{1}{1 - az^{-1}} = \sum_{n=0}^{\infty} a^n z^{-n}$$

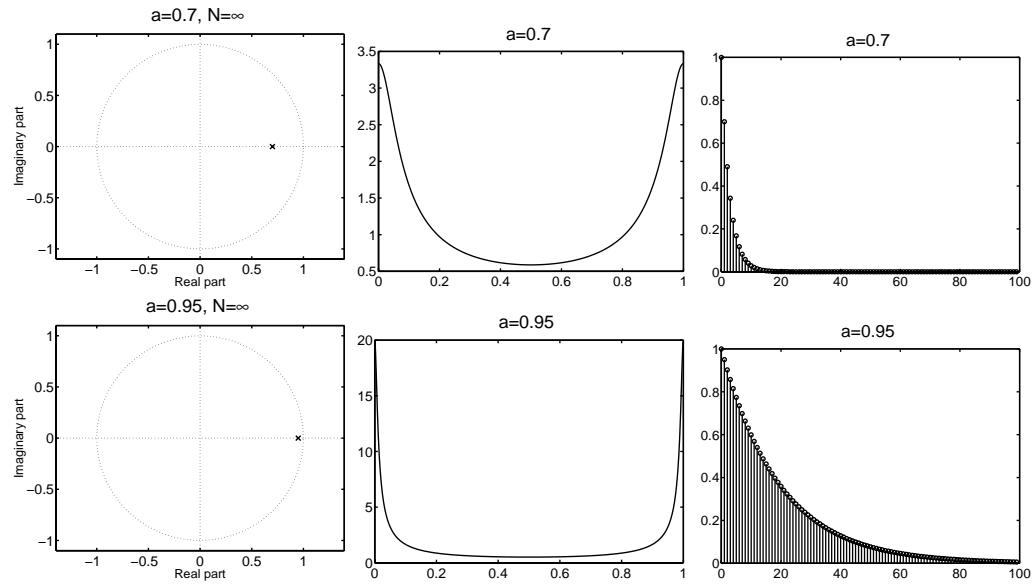
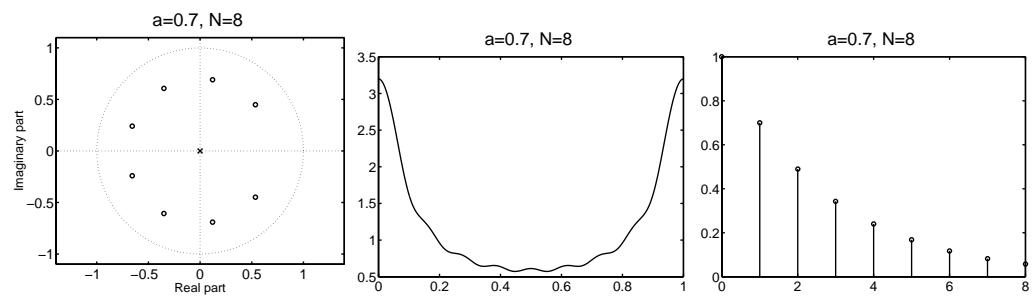
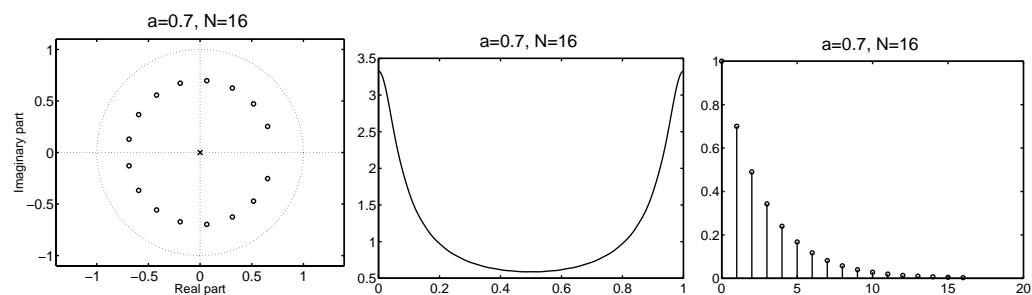
It can be approximated with a finite number of zeros with a system function of

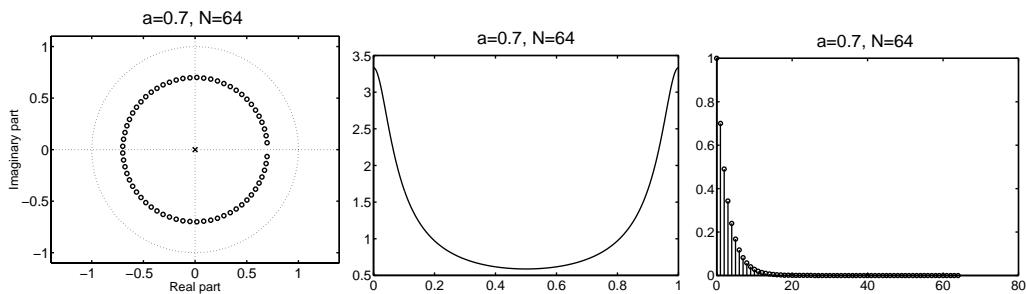
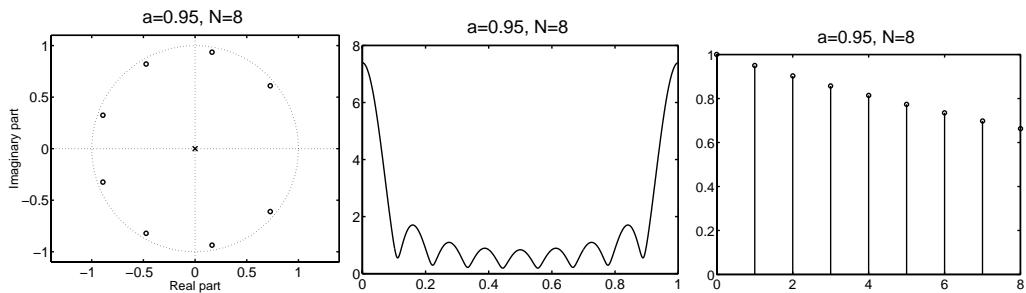
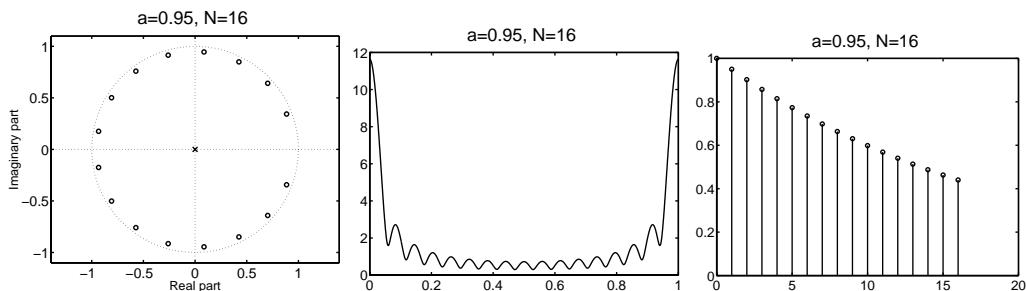
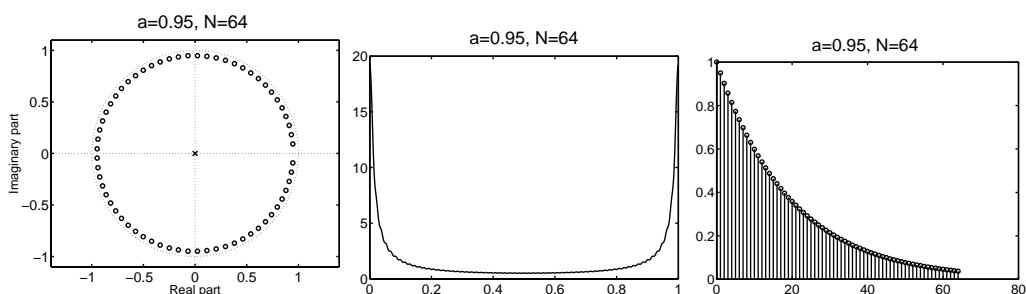
$$H_N(z) = \sum_{n=0}^N a^n z^{-n}$$

The Figure 4.5.1 shows the pole-zero plots, magnitude plots, and impulse response for a single pole system with the poles located at $a = 0.7$ and 0.95 respectively.

Figure 4.5.2 shows the pole-zero plots, magnitude plots and impulse response for $a = 0.7$ and $N = 8$ Figure 4.5.3 shows the pole-zero plots, magnitude plots and impulse response for $a = 0.7$ and $N = 16$ Figure 4.5.4 shows the pole-zero plots, magnitude plots and impulse response for $a = 0.7$ and $N = 64$ Figure 4.5.5 shows the pole-zero plots, magnitude plots and impulse response for $a = 0.95$ and $N = 8$ Figure 4.5.6 shows the pole-zero plots, magnitude plots and impulse response for $a = 0.95$ and $N = 16$ Figure 4.5.7 shows the pole-zero plots, magnitude plots and impulse response for $a = 0.95$ and $N = 64$

Clearly from these figures, it can be seen that an all zero system can approximate a single pole system. The number of zeroes necessary depends on how close to the unit circle the pole lies. As the pole moves closer to the unit circle, more zeroes are needed to have the same magnitude response.

**Figure 4.5.1:** Gaussian Process**Figure 4.5.2:** Gaussian Process**Figure 4.5.3:** Gaussian Process

**Figure 4.5.4:** Gaussian Process**Figure 4.5.5:** Gaussian Process**Figure 4.5.6:** Gaussian Process**Figure 4.5.7:** Gaussian Process

4.6 Consider (4.2.33) for $P = 2$:

$$-\begin{bmatrix} \rho_1 \\ \rho_2 \end{bmatrix} = \begin{bmatrix} 1 & \rho_1 \\ \rho_1 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} a_1 + \rho_1 a_2 \\ \rho_1 a_1 + a_2 \end{bmatrix}$$

Hence

$$\begin{aligned} -a_1 &= \rho_1 + \rho_1 a_2 = \rho_1(1 + a_2) \\ -a_2 &= \rho_1 a_1 + \rho_2 \end{aligned}$$

or

$$-\mathbf{a} = -\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 1 + a_2 & 0 \\ a_1 & 1 \end{bmatrix} \begin{bmatrix} \rho_1 \\ \rho_2 \end{bmatrix} = \left\{ \begin{bmatrix} 1 & 0 \\ a_1 & 1 \end{bmatrix} + \begin{bmatrix} a_2 & 0 \\ 0 & 0 \end{bmatrix} \right\} \begin{bmatrix} \rho_1 \\ \rho_2 \end{bmatrix} = \mathbf{A}\boldsymbol{\rho}$$

Thus the matrix \mathbf{A} is a sum of triangular Toeplitz and triangular Hankel matrices. Consider (4.2.33) for $P = 3$:

$$-\begin{bmatrix} \rho_1 \\ \rho_2 \\ \rho_3 \end{bmatrix} = \begin{bmatrix} 1 & \rho_1 & \rho_2 \\ \rho_1 & 1 & \rho_1 \\ \rho_2 & \rho_1 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} a_1 + \rho_1 a_2 + \rho_2 a_3 \\ \rho_1 a_1 + a_2 + \rho_1 a_3 \\ \rho_2 a_1 + \rho_1 a_2 + a_3 \end{bmatrix}$$

Hence

$$\begin{aligned} -a_1 &= \rho_1 + \rho_1 a_2 + \rho_2 a_3 = \rho_1(1 + a_2) + \rho_2 a_3 \\ -a_2 &= \rho_1 a_1 + \rho_2 + \rho_1 a_3 = \rho_1(a_1 + a_3) + \rho_2 \\ -a_3 &= \rho_2 a_1 + \rho_1 a_2 + \rho_3 = \rho_1 a_2 + \rho_2 a_1 + \rho_3 \end{aligned}$$

or

$$\begin{aligned} -\mathbf{a} &= -\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} (1 + a_2) & a_3 & 0 \\ (a_1 + a_3) & 1 & 0 \\ a_2 & a_1 & 1 \end{bmatrix} \begin{bmatrix} \rho_1 \\ \rho_2 \\ \rho_3 \end{bmatrix} \\ &= \left\{ \begin{bmatrix} 1 & 0 & 0 \\ a_1 & 1 & 0 \\ a_2 & a_1 & 1 \end{bmatrix} + \begin{bmatrix} a_2 & a_3 & 0 \\ a_3 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \right\} \begin{bmatrix} \rho_1 \\ \rho_2 \\ \rho_3 \end{bmatrix} = \mathbf{A}\boldsymbol{\rho} \end{aligned}$$

Again the matrix \mathbf{A} is a sum of triangular Toeplitz and triangular Hankel matrices. Similarly for $P = 4$, we obtain

$$-\begin{bmatrix} \rho_1 \\ \rho_2 \\ \rho_3 \\ \rho_4 \end{bmatrix} = \begin{bmatrix} 1 & \rho_1 & \rho_2 & \rho_3 \\ \rho_1 & 1 & \rho_1 & \rho_2 \\ \rho_2 & \rho_1 & 1 & \rho_1 \\ \rho_3 & \rho_2 & \rho_1 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} a_1 + \rho_1 a_2 + \rho_2 a_3 + \rho_3 a_4 \\ \rho_1 a_1 + a_2 + \rho_1 a_3 + \rho_2 a_4 \\ \rho_2 a_1 + \rho_1 a_2 + a_3 + \rho_1 a_4 \\ \rho_3 a_1 + \rho_2 a_2 + \rho_1 a_3 + a_4 \end{bmatrix}$$

or

$$-\mathbf{a} = -\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \left\{ \begin{bmatrix} 1 & 0 & 0 & 0 \\ a_1 & 1 & 0 & 0 \\ a_2 & a_1 & 1 & 0 \\ a_3 & a_2 & a_1 & 1 \end{bmatrix} + \begin{bmatrix} a_2 & a_3 & a_4 & 0 \\ a_3 & a_4 & 0 & 0 \\ a_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \right\} \begin{bmatrix} \rho_1 \\ \rho_2 \\ \rho_3 \\ \rho_4 \end{bmatrix} = \mathbf{A}\boldsymbol{\rho}$$

Hence for a general case, the matrix \mathbf{A} is given by the sum of triangular Toeplitz and triangular Hankel matrices.

4.7 To determine the autocorrelation and partial autocorrelation coefficients of the following two AR models, we can use the Yule-Walker equations

$$\mathbf{R}_x \mathbf{a} = -\mathbf{r}_x$$

(a) $x(n) = 0.5x(n-1) + w(n)$:

Solving directly,

$$\begin{aligned} \begin{bmatrix} r_x(0) & r_x(1) \\ r_x(1) & r_x(0) \end{bmatrix} \begin{bmatrix} 1 \\ a \end{bmatrix} &= \begin{bmatrix} |d_0|^2 \\ 0 \end{bmatrix} \\ \begin{bmatrix} r_x(0) & r_x(1) \\ r_x(1) & r_x(0) \end{bmatrix} \begin{bmatrix} 1 \\ -0.5 \end{bmatrix} &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{aligned}$$

Rearranging terms and computing the matrix inverse results in

$$\begin{aligned} \begin{bmatrix} -0.5 & 1 \\ 1 & -0.5 \end{bmatrix} \begin{bmatrix} r_x(1) \\ r_x(0) \end{bmatrix} &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ \begin{bmatrix} r_x(1) \\ r_x(0) \end{bmatrix} &= \begin{bmatrix} 0.667 & 1.333 \\ 1.333 & 0.667 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.667 \\ 1.333 \end{bmatrix} \end{aligned}$$

with a variance $\sigma_x^2 = r_x(0) = 1.333$. The corresponding partial autocorrelation is

$$\begin{aligned} a_1^{(1)} &= -\rho(1) = r_x(1)/r_x(0) \\ &= -0.667/1.333 \\ &= -0.5 \end{aligned}$$

Therefore the autocorrelation coefficients are $r_x(0) = 1.33$ and $r_x(1) = 0.667$ with a partial autocorrelation coefficient equal to $a_1^{(1)} = -0.5$.

(b) $x(n) = 1.5x(n-1) - 0.6x(n-2) + w(n)$

Solving directly,

$$\begin{aligned} \begin{bmatrix} r_x(0) & r_x(1) & r_x(2) \\ r_x(1) & r_x(0) & r_x(1) \\ r_x(2) & r_x(1) & r_x(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ a_2 \end{bmatrix} &= \begin{bmatrix} |d_0|^2 \\ 0 \\ 0 \end{bmatrix} \\ \begin{bmatrix} r_x(0) & r_x(1) & r_x(2) \\ r_x(1) & r_x(0) & r_x(1) \\ r_x(2) & r_x(1) & r_x(0) \end{bmatrix} \begin{bmatrix} 1 \\ -1.5 \\ 0.6 \end{bmatrix} &= \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \end{aligned}$$

Rearranging terms and computing the matrix inverse results in

$$\begin{aligned} \begin{bmatrix} 1 & -1.5 & 0.6 \\ -1.5 & 1.6 & 0 \\ 0.6 & -1.5 & 1 \end{bmatrix} \begin{bmatrix} r_x(0) \\ r_x(1) \\ r_x(2) \end{bmatrix} &= \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \\ \begin{bmatrix} r_x(0) \\ r_x(1) \\ r_x(2) \end{bmatrix} &= \begin{bmatrix} 12.9 & 4.84 & -7.74 \\ 12.01 & 5.16 & -7.26 \\ 10.4 & 4.84 & -5.24 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 12.9 \\ 12.01 \\ 10.4 \end{bmatrix} \end{aligned}$$

with a variance $\sigma_x^2 = r_x(0) = 12.9$. The corresponding partial autocorrelation is

$$\begin{aligned} a_1^{(1)} &= -\rho(1) = -12.01/12.9 = -0.931 \\ \begin{bmatrix} 1 & \rho(1) \\ \rho(1) & 1 \end{bmatrix} \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \end{bmatrix} &= -\begin{bmatrix} \rho(1) \\ \rho(2) \end{bmatrix} \\ \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \end{bmatrix} &= -\begin{bmatrix} 1 & 0.931 \\ 0.931 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0.931 \\ 0.806 \end{bmatrix} = \begin{bmatrix} 1.356 \\ 0.456 \end{bmatrix} \end{aligned}$$

Therefore the autocorrelation coefficients are $r_x(0) = 12.9$, $r_x(1) = 12.01$ and $r_x(3) = 10.4$ with the partial autocorrelation coefficients equal to $a_1^{(1)} = -0.931$ and $a_2^{(2)} = 0.456$.

4.8 Given $x(n) = x(n-1) - 0.5x(n-2) + w(n)$:

(a) To find $\rho_x(1)$ first solve for the first two autocorrelation coefficients

$$\begin{aligned} r_x(0) &= E\{[x(n-1) - 0.5x(n-2) + w(n)][x(n-1) - 0.5x(n-2) + w(n)]\} \\ &= (1.25)r_x(0) - r_x(1) + 1 \\ r_x(1) &= E\{x(n)x(n-1)\} = E\{[x(n-1) - 0.5x(n-2) + w(n)]x(n-1)\} \\ &= r_x(0) - 0.5r_x(1) \end{aligned}$$

or $1.5r_x(1) = r_x(0)$. Therefore, $\rho_x(1) = r_x(1)/r_x(0) = 2/3$.

(b) To compute $\rho_x(l)$ use the following difference equation

$$\rho_x(l) = \rho_x(l-1) - 0.5\rho_x(l-2)$$

(c) The plot of $\rho_x(l)$ is shown in the left plot of Figure 4.8cd.

(d) The corresponding PSD of the model is shown in the right plot of Figure 4.8cd.

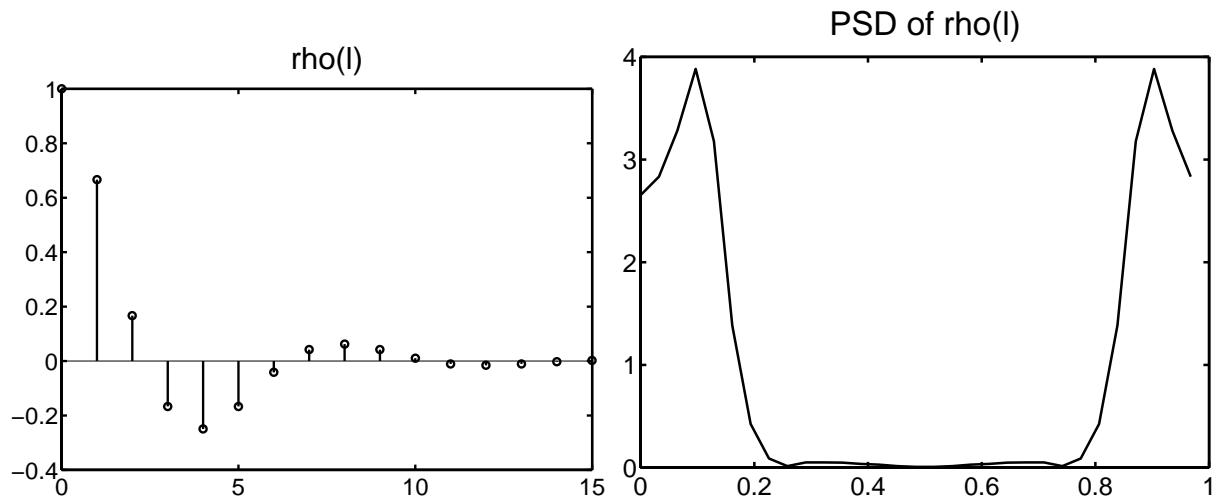


Figure 4.8cd: Period of $\rho(l)$

The period (T) shown in the plot of $\rho(l)$ is approximately 8. Thus, the frequency is $0.125(1/T)$. This is approximately the frequency peak in the PSD plot.

4.9 The ACS can be computed directly by rearranging terms and using the matrix inverse

$$\begin{aligned} \begin{bmatrix} r_x(0) & r_x(1) & r_x(2) & r_x(3) \\ r_x(1) & r_x(0) & r_x(1) & r_x(2) \\ r_x(2) & r_x(1) & r_x(0) & r_x(1) \\ r_x(3) & r_x(2) & r_x(1) & r_x(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} &= \begin{bmatrix} |d_0|^2 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\ \begin{bmatrix} 1 & a_1 & a_2 & a_3 \\ a_1 & 1+a_2 & a_3 & 0 \\ a_2 & a_1+a_3 & 1 & 0 \\ a_3 & a_2 & a_1 & 1 \end{bmatrix} \begin{bmatrix} r_x(0) \\ r_x(1) \\ r_x(2) \\ r_x(3) \end{bmatrix} &= \begin{bmatrix} |d_0|^2 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\ \begin{bmatrix} r_x(0) \\ r_x(1) \\ r_x(2) \\ r_x(3) \end{bmatrix} &= \begin{bmatrix} 1 & a_1 & a_2 & a_3 \\ a_1 & 1+a_2 & a_3 & 0 \\ a_2 & a_1+a_3 & 1 & 0 \\ a_3 & a_2 & a_1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} |d_0|^2 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{aligned}$$

Substituting in the values for the parameters from Example 4.2.3 results in the correct values for the ACS.

$$\begin{bmatrix} r_x(0) \\ r_x(1) \\ r_x(2) \\ r_x(3) \end{bmatrix} = \begin{bmatrix} 2.8840 & 2.0011 & 0.0003 & 0.4508 \\ 2.3074 & 2.5236 & 0.1444 & 0.3607 \\ 1.7309 & 2.1813 & 1.1534 & 0.2705 \\ 1.4426 & 1.6588 & 1.0093 & 1.2255 \end{bmatrix} \begin{bmatrix} 0.6937 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 2.0 \\ 1.6 \\ 1.2 \\ 1.0 \end{bmatrix}$$

Note: this problem can also be solved using Cramer's rule. The solution for Problem 4.12 shows how to use Cramer's rule.

4.10 Given the AP(3) model: $x(n) = 0.98x(n-3) + w(n)$

(a) In order to plot the PSD of $x(n)$ first compute the ACS

$$\begin{aligned} r_x(0) &= E\{x(n)x(n)\} = E\{[0.98x(n-3) + w(n)][0.98x(n-3) + w(n)]\} \\ &= 0.98^2r_x(0) + 1 = 25.25 \\ r_x(1) &= E\{[0.98x(n-3) + w(n)][x(n-1)]\} = 0.98r_x(2) \\ r_x(2) &= E\{[0.98x(n-3) + w(n)][x(n-2)]\} = 0.98r_x(1) \\ r_x(3) &= E\{[0.98x(n-3) + w(n)][x(n-3)]\} = 0.98r_x(0) \\ r_x(6) &= E\{[0.98x(n-3) + w(n)][x(n-6)]\} = 0.98r_x(3) = 0.98^2r_x(0) \end{aligned}$$

This results in an ACS of

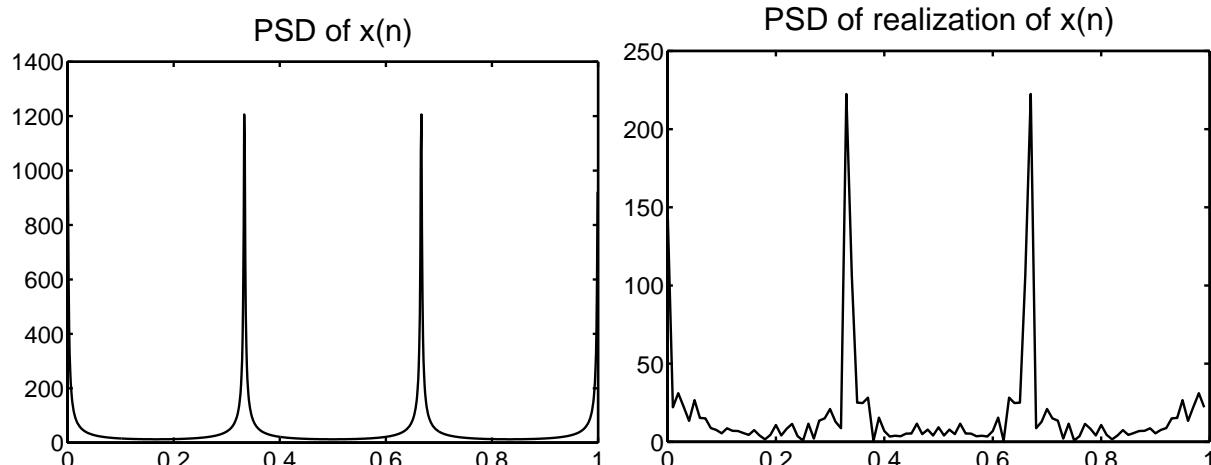
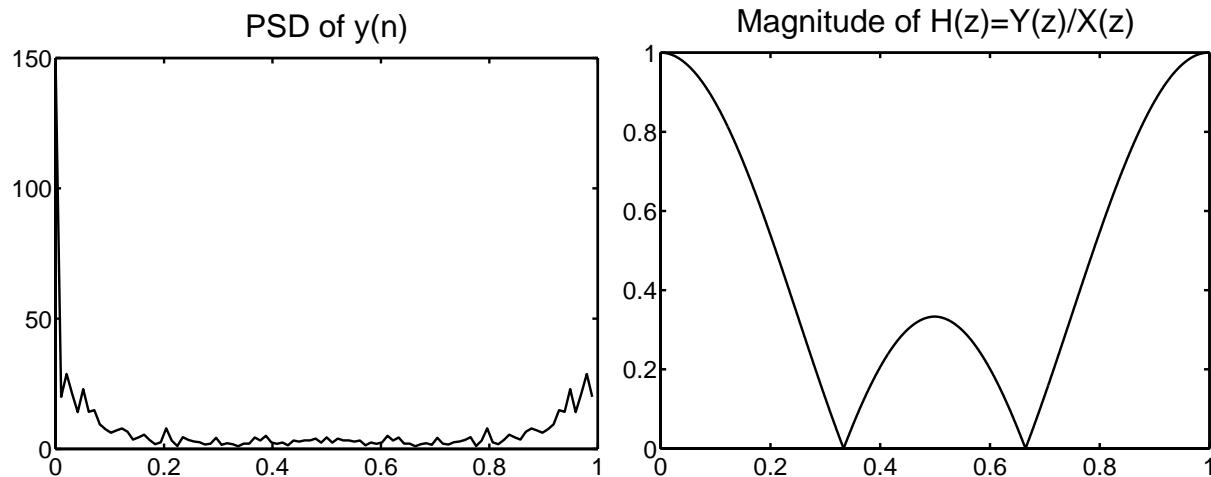
$$r_x(l) = \begin{cases} (0.98)^{\frac{l}{3}}r_x(0) & l = 0, 3, 6, 9, \dots \\ r_x(0) = 25.25 & \\ 0 & \text{otherwise} \end{cases}$$

The PSD can be found by taking the Fourier transform of the ACS. This is shown in the left plot of Figure 4.10ab.

(b) The PSD is computed by taking the squared-magnitude of the Fourier transform of realization of $x(n)$. This is shown in the right plot of Figure 4.10ab.

(c) Given $y(n) = \frac{1}{3}[x(n-1) + x(n) + x(n+1)]$, its ACS is computed as follows

$$r_y(l) = (\frac{1}{3})^2[3r_x(l) + 4r_x(l+1) + 2r_x(l+2)]$$

**Figure 4.10ab:** Power Spectral Density**Figure 4.10cd:** Power Spectral Density

- (d) The PSD of $y(n)$ is computed in a similar manner as $x(n)$ above. It is shown in the left plot of Figure 4.10cd. The right plot of Figure 4.10cd shows the magnitude response of the system $H(z) = Y(z)/x(z)$. Clearly, $H(z)$ is a notch filter, with the frequency notches corresponding to the peak frequency of the PSD of $x(n)$. This filter removes the strong correlation found in $x(n)$ and generates an uncorrelated signal found in part (c) above.

4.11 Given two AR(2) models:

- (a) The normalized autocorrelation sequence, for each of the two models, can be found directly as follows

- $x(n) = 0.6x(n - 1) + 0.3x(n - 2) + w(n)$:

$$\begin{aligned} r_x(0) &= E\{[0.6x(n-1) + 0.3x(n-2) + w(n)][0.6x(n-1) + 0.3x(n-2) + w(n)]\} \\ &= r_x(0)(0.6^2 + 0.3^2) + 2(0.6)(0.3)r_x(1) + 1 \end{aligned}$$

$$\begin{aligned} r_x(1) &= E\{x(n)x(n-1)\} = E\{[0.6x(n-1) + 0.3x(n-2) + w(n)][x(n-1)]\} \\ &= 0.6r_x(0) + 0.3r_x(1) = 0.857r_x(0) \end{aligned}$$

$$\begin{aligned} r_x(l) &= E\{x(n+l)x(n)\} = E\{x(n+l)[0.6x(n-1) + 0.3x(n-2) + w(n)]\} \\ &= 0.6r_x(l-1) + 0.3r_x(l-2) \end{aligned}$$

Thus $r_x(0) = \sigma_x^2 = 4.146$, and $r_x(1) = 3.553$. Therefore, the normalized autocorrelation sequence is

$$\rho_x(l) = r_x(l)/r_x(0) = 0.6\rho(l-1) + 0.3\rho(l-2)$$

ii. $x(n) = 0.8x(n-1) - 0.5x(n-2) + w(n)$:

$$\begin{aligned} r_x(0) &= E\{[0.8x(n-1) - 0.5x(n-2) + w(n)][0.8x(n-1) - 0.5x(n-2) + w(n)]\} \\ &= r_x(0)(0.8^2 + 0.5^2) + 2(0.8)(-0.5)r_x(1) + 1 \end{aligned}$$

$$\begin{aligned} r_x(1) &= E\{x(n)x(n-1)\} = E\{[0.8x(n-1) - 0.5x(n-2) + w(n)][x(n-1)]\} \\ &= 0.8r_x(0) - 0.5r_x(1) = 0.533r_x(0) \end{aligned}$$

$$\begin{aligned} r_x(l) &= E\{x(n+l)x(n)\} = E\{x(n+l)[0.8x(n-1) - 0.5x(n-2) + w(n)]\} \\ &= 0.8r_x(l-1) - 0.5r_x(l-2) \end{aligned}$$

Thus $r_x(0) = \sigma_x^2 = 1.863$ and $r_x(1) = 0.994$. Therefore, the normalized autocorrelation sequence is

$$\rho_x(l) = r_x(l)/r_x(0) = 0.8\rho(l-1) - 0.5\rho(l-2)$$

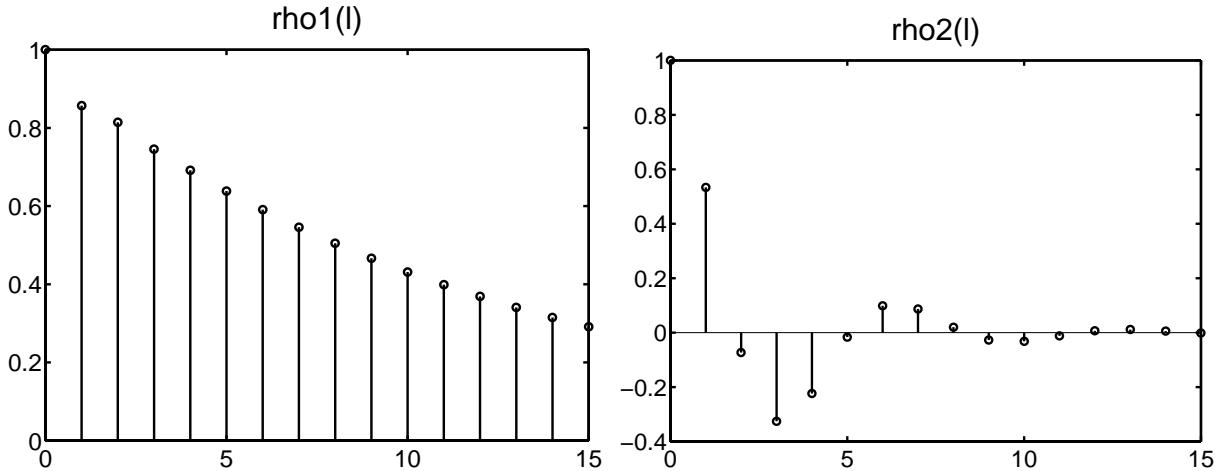
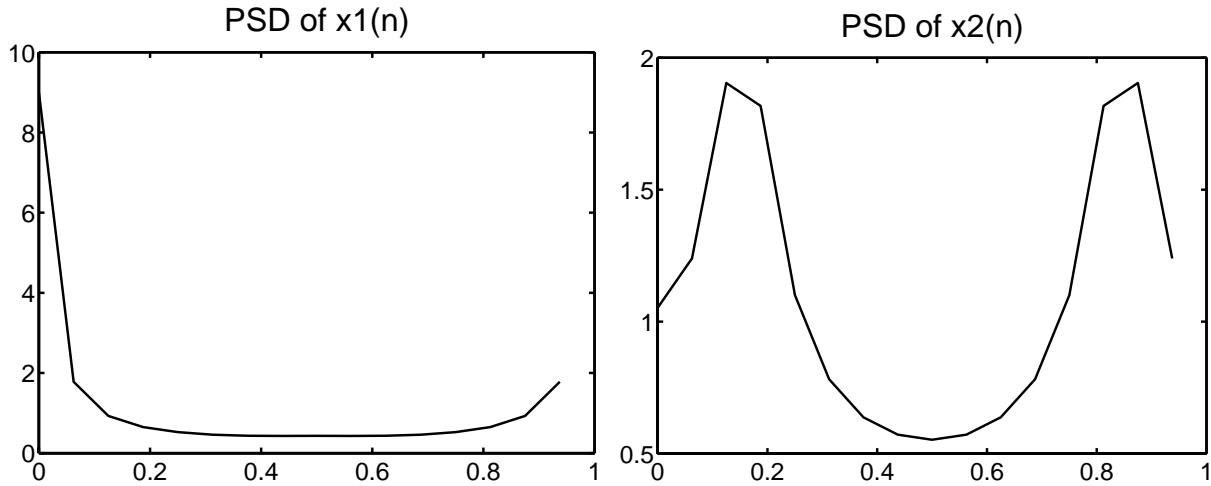


Figure 4.11b: AR(2) System

- (b) Figure 4.11b shows the normalized autocorrelation sequences of the two models respectively. From these plots, $x(n) = 0.6x(n-1) + 0.3x(n-2) + w(n)$ does not show any pseudo-periodic behavior, while $x(n) = 0.8x(n-1) - 0.5x(n-2) + w(n)$ does show pseudo-periodic behavior.
- (c) Figure 4.11c shows the PSD of the two models. Clearly, the first model has no fundamental frequency. While the second system does indeed have a large non-zero frequency component.

4.12 The PACS for an AP(3) model can be derived from the Yule-Walker equations and Cramer's rule.

**Figure 4.11c: AR(2) System**

(a) The Yule-Walker equations are

$$\mathbf{R}_x \mathbf{a} = -\mathbf{r}_x$$

Cramer's rule is used to determine vector values without having to compute the inverse of the entire matrix. If $x = A^{-1}b$ then the j th component of x can be computed using the determinants of two matrices $x_j = \frac{\det B_j}{\det A}$ where B_j is the A matrix with the j th column replaced with the b vector

$$B_j = \begin{bmatrix} a_{1,1} & \cdots & a_{1,j-1} & b_1 & a_{1,j+1} & \cdots & a_{1,n} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ a_{n,1} & \cdots & a_{n,j-1} & b_n & a_{n,j+1} & \cdots & a_{n,n} \end{bmatrix}$$

Using Yule-Walker and Cramer's rule, the PACS can be computed using

$$a_m^{(M)} = \frac{\det R_{x(m)}^{(M)}}{\det R_x^{(M)}}$$

where

$$R_x^{(M)} = \begin{bmatrix} r_x(0) & r_x(1) & r_x(2) \\ r_x^*(1) & r_x(0) & r_x(1) \\ r_x^*(2) & r_x^*(1) & r_x(0) \end{bmatrix}$$

and $R_{x(m)}^{(M)} = R_x^{(M)}$ with the j th column replaced with $r_x = \begin{bmatrix} r_x(1) \\ \vdots \\ r_x(M) \end{bmatrix}$

(b) The following MATLAB code can be used to compute the PACS of the AP(3) model in Example 4.2.3

MATLAB Code:

```
r0=2; r1=1.6; r2=1.2; r3=1;
R=det([r0 r1 r2; r1 r0 r1; r2 r1 r0])
a1 = -det([r1 r1 r2; r2 r0 r1; r3 r1 r0])/R
a2 = -det([r0 r1 r2; r1 r2 r1; r2 r3 r0])/R
a3 = -det([r0 r1 r1; r1 r0 r2; r2 r1 r3])/R
```

This code produces the following results

```
R = 1.0240
a1 = -0.9063
a2 = 0.2500
a3 = -0.1563
```

which is equivalent to the results in Example 4.2.3

- 4.13** To show that the spectrum of any PZ model with real coefficients has zero slope at $\omega = 0$ and $\omega = \pi$, the first derivative of the PSD of the model is computed and found to be equal to zero at $\omega = 0$ and $\omega = \pi$. The general expression for a PZ model in the z-domain is

$$H(z) = \frac{\sum \alpha_i z^{-i}}{\sum \beta_i z^{-i}}$$

Its squared-magnitude is

$$|H(z)|^2 = \frac{|\sum \alpha_i z^{-i}|^2}{|\sum \beta_i z^{-i}|^2} = \frac{\sum_m \sum_n \alpha_m \alpha_n z^{-m+n}}{\sum_k \sum_l \beta_k \beta_l z^{-k+l}}$$

Replacing $z = e^{-j\omega}$ gives the following general expression for the PSD of a PZ model

$$|H(e^{j\omega})|^2 = \frac{\sum_m \sum_n \alpha_m \alpha_n e^{-j(m-n)\omega}}{\sum_k \sum_l \beta_k \beta_l e^{-j(k-l)\omega}}$$

Taking its first derivative

$$\begin{aligned} \frac{d|H(e^{j\omega})|^2}{d\omega} &= (\sum_k \sum_l \beta_k \beta_l e^{-j(k-l)\omega})(\sum_m \sum_n \alpha_m \alpha_n (-j)(m-n)e^{-j(m-n)\omega}) \\ &\quad - (\sum_m \sum_n \alpha_m \alpha_n e^{-j(m-n)\omega})(\sum_k \sum_l \beta_k \beta_l (-j)(k-l)e^{-j(k-l)\omega}) \\ &= \sum_k \sum_l \sum_m \sum_n \alpha_m \alpha_n \beta_k \beta_l (-j)((m-n)-(k-l))e^{-j(k-l+m-n)\omega} \\ &= -j \sum_k \sum_l \sum_m \sum_n \alpha_m \alpha_n \beta_k \beta_l (m-n-k+l)e^{-j(k-l+m-n)\omega} \end{aligned}$$

At $\omega = 0$

$$\frac{d|H(e^{j\omega})|^2}{d\omega} = -j \sum_k \sum_l \sum_m \sum_n \alpha_m \alpha_n \beta_k \beta_l (m-n-k+l) = 0$$

and at $\omega = \pi$

$$\frac{d|H(e^{j\omega})|^2}{d\omega} = j \sum_k \sum_l \sum_m \sum_n \alpha_m \alpha_n \beta_k \beta_l (m-n-k+l) = 0$$

Therefore, the spectrum has zero slope.

- 4.14** Derive the equations for the minimum-phase region of the AP(2) model, which are

$$|a_2| < 1; \quad a_2 - a_1 > -1; \quad a_2 + a_1 > -1$$

(a) $|p_1| < 1$ and $|p_2| < 1$:

Separating the magnitude into two one-sided inequalities $p_1 < 1$ and $p_1 > 1$ The derivation follows [where $a_1 = -(p_1 + p_2)$ and $a_2 = p_1 p_2$]

$$p_1 < 1$$

Multiplying both sides by the negative quantity $p_2 - 1$

$$\begin{aligned} p_1(p_2 - 1) &> p_2 - 1 \\ p_1 p_2 - p_1 - p_2 &> -1 \\ a_2 + a_1 &> -1 \end{aligned}$$

Similarly,

$$p_1 > -1$$

Multiplying both sides by the positive quantity $p_2 + 1$

$$\begin{aligned} p_1(p_2 + 1) &> -p_2 - 1 \\ p_1 p_2 + p_1 + p_2 &> -1 \\ a_2 - a_1 &> -1 \end{aligned}$$

Lastly, with the magnitude of both p_1 and p_2 less than 1, their product is also less than one, and $|a_2| < 1$.

(b) $|k_1| < 1$ and $|k_2| < 1$:

Given $k_2 = a_2$, then $|a_2| < 1$. With

$$k_1 = \frac{a_1}{1 + a_2}$$

and direct substitution into $|k_1| < 1$ the equations can be derived as follows

$$\left| \frac{a_1}{1 + a_2} \right| < 1$$

Solving both side of the magnitude equation separately results in

$$a_1 > 1 + a_2 \implies a_2 - a_1 > -1$$

and

$$a_1 > -(1 + a_2) \implies a_2 + a_1 > -1$$

4.15 Spectrum of AP models

(a) Show that the spectrum of an AP(2) model with real coefficients can be obtained by the cascade connection of two AP(1) models with real coefficients.

First, the general equation for the denominator of the spectrum of an AP(2) model with real coefficients is computed. Starting in the z-domain

$$A_2(z) = (1 + p_1 z^{-1})(1 + p_2 z^{-1})$$

Finding its squared-magnitude

$$|A_2(z)|^2 = A_2(z) A_2^*(\frac{1}{z}) = (1 + p_1 z^{-1})(1 + p_2 z^{-1})(1 + p_1 z^1)(1 + p_2 z^1)$$

Converting to the frequency domain by substituting $z = e^{-j\omega}$ result in the spectrum as follows

$$|A_2(\omega)|^2 = 1 + (p_1 + p_2)^2 + (p_1 p_2)^2 + 2(p_1 + p_2)(1 + p_1 p_2) \cos \omega + 2(p_1 p_2) \cos 2\omega$$

The denominator of the spectrum of an AP(1) model is

$$|A_1(\omega)|^2 = 1 + p_1^2 + 2p_1 \cos \omega$$

Cascading two systems multiplies their corresponding spectra. In the case of AP models, it results in the multiplication of their two denominators.

$$|A_1(\omega)|^2 |A_2(\omega)|^2 = (1 + p_1^2 + 2p_1 \cos \omega)(1 + p_2^2 + 2p_2 \cos \omega)$$

Using the trigonometric identity $\cos 2\alpha = 2\cos^2 \alpha - 1$ and completing the above multiplication, results in the Therefore being equal.

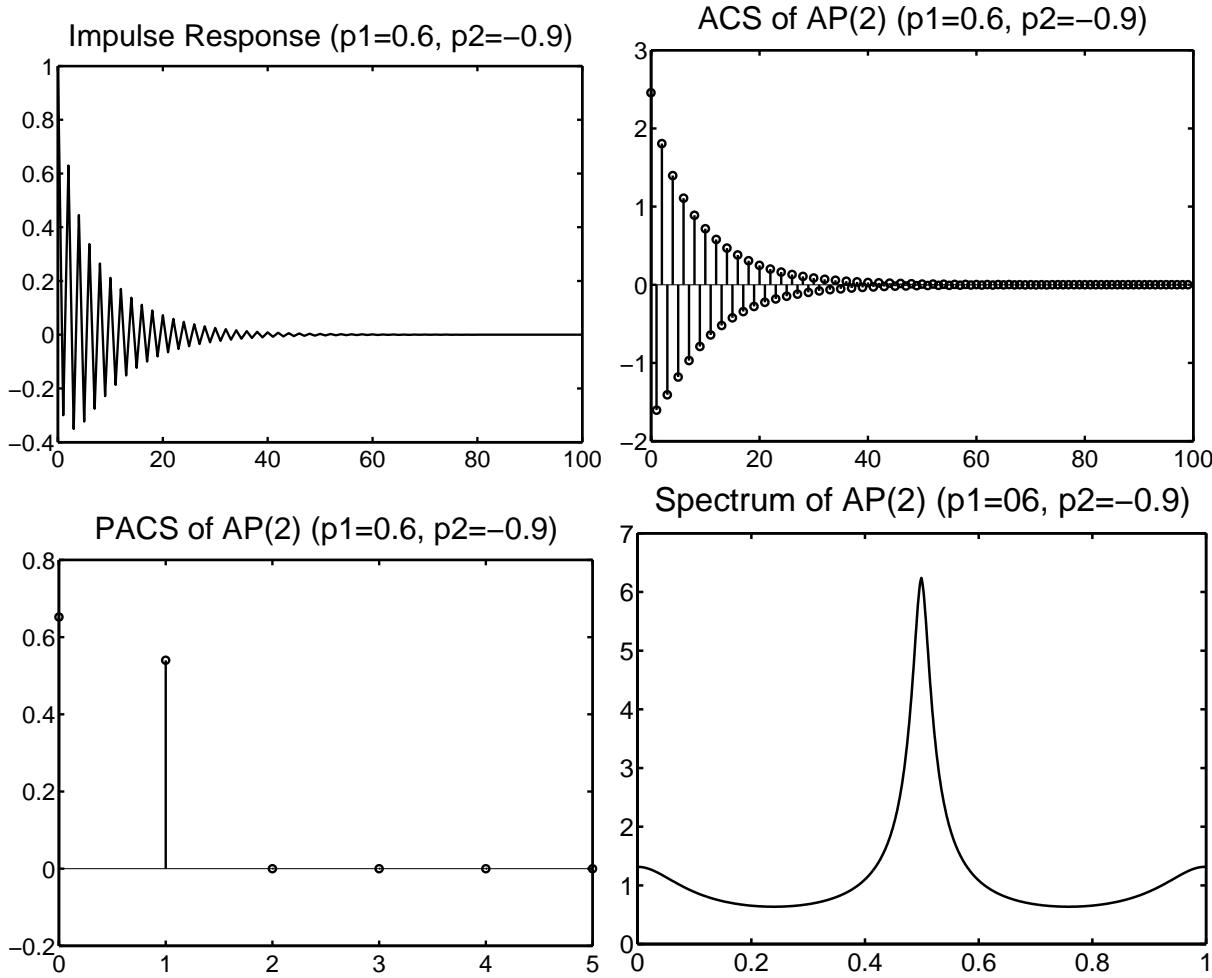
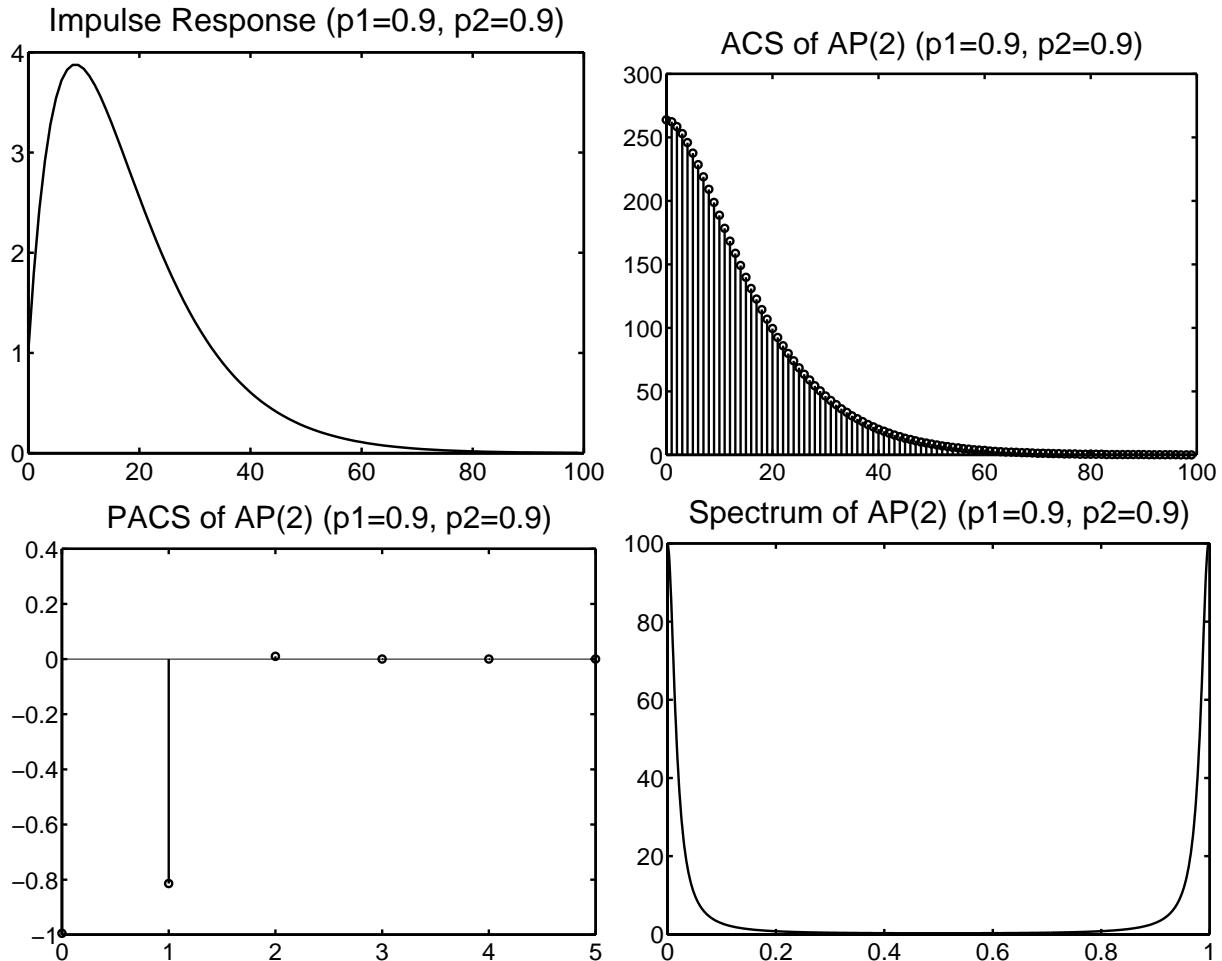


Figure 4.15.1: AP(2) System

- (b) Figure 4.15.1 and 4.15.2 show the impulse response, ACS, PACS, and spectrum of an AP(2) model. Figure 4.15.1 has two real poles located at $p_1 = 0.6$ and $p_2 = -0.9$. Figure 4.15.2 has a double pole located at $p = 0.9$.

**Figure 4.15.2:** AP(2) System with double poles

- 4.16** Prove Equation 4.2.89 which states that the peak of the spectrum in an AP(2) system with complex conjugate poles is

$$\cos \omega_c = \frac{(1 + r^2)}{2r} \cos \theta$$

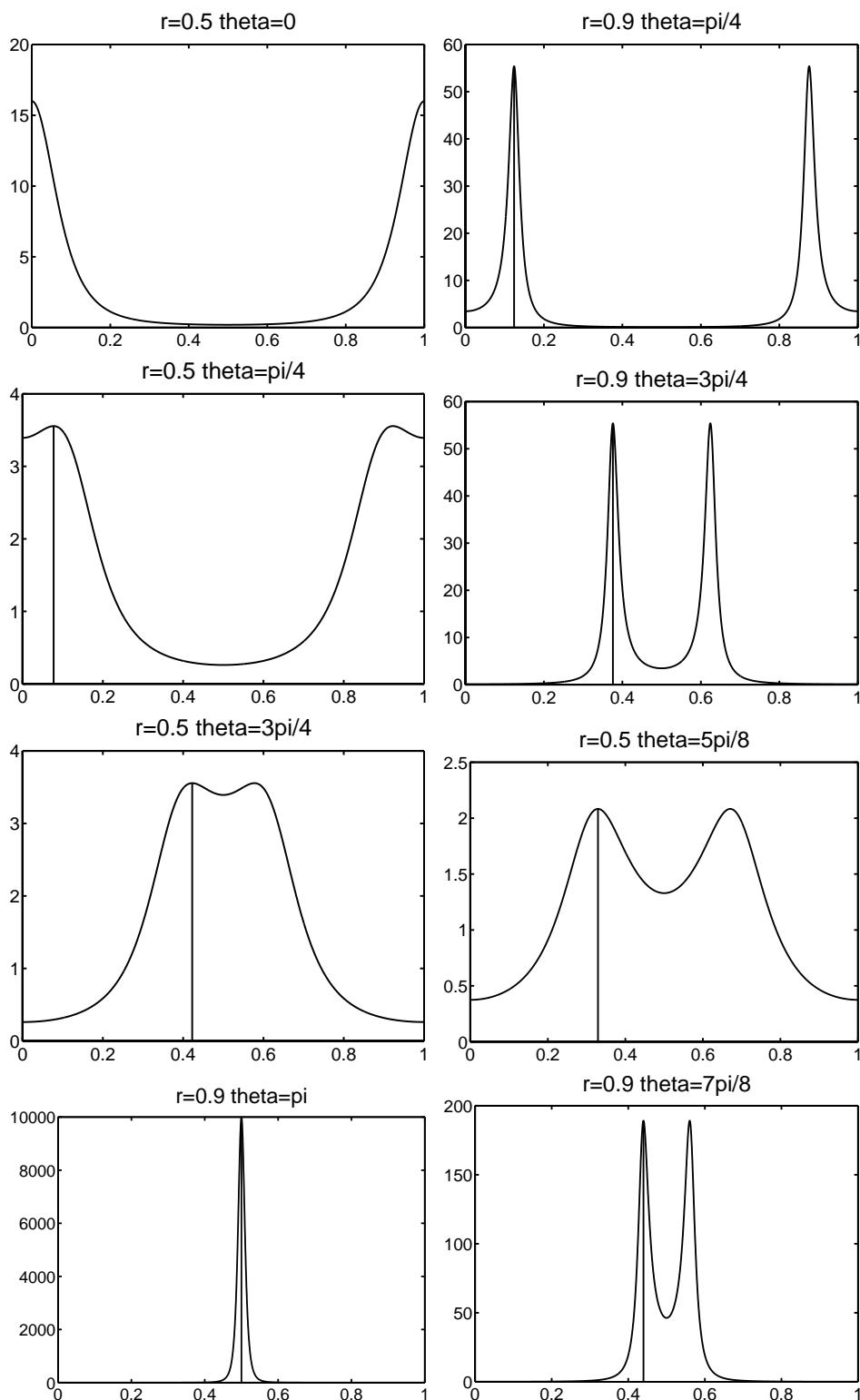
This can be solved directly from the spectrum of an AP(2) system with complex conjugate poles

$$R(e^{j\omega}) = \frac{d_0^2}{[1 - 2r \cos(\omega_c - \theta) + r^2][1 - 2r \cos(\omega_c + \theta) + r^2]}$$

Taking the derivative of the denominator and setting it equal to zero results in

$$(1 + r^2)(\sin \omega_c \cos \theta) - r \sin(2\omega_c) = 0$$

Using the following trigonometric identity $\sin 2\alpha = 2 \sin \alpha \cos \alpha$ and simplifying finishes the proof. Figure 4.16 shows the spectrum for various values of r and θ . The vertical line is computed using Equation 4.2.89

**Figure 4.16:** AP(2) Complex Pole Systems

4.17 Prove that if the AP(P) model $A(z)$ is minimum-phase, then

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} \log \frac{1}{|A(e^{j\omega})|^2} d\omega = 0$$

Start with

$$= \frac{1}{2\pi} \int_{-\pi}^{\pi} \log |A(e^{-j\omega})|^2 d\omega$$

Using the log identity

$$= \frac{1}{2\pi} 2Re \left(\int_{-\pi}^{\pi} \log A(e^{-j\omega}) d\omega \right)$$

Change to the z-domain, where the integral becomes a contour integral

$$= \frac{1}{2jz\pi} 2Re \left(\oint_{\Gamma} \log A(z^{-1}) dz \right)$$

and Γ is a contour that includes all the zeros of $A(z)$. This equation then simplifies to

$$= 2Re(\log[A(\infty)]) = 2Re[\log(1)]$$

Since the $\log(1) = 0$, this proves the above equation.

4.18 Recreate plots in Figure 4.8.

(a) Prove Equations (4.2.101 and 4.2.102) which are respectively

$$0 < k_2 < 1 \text{ and } k_1^2 < \frac{4k_2}{(1+k_2)^2}$$

Starting with the denominator of the system

$$A(z) = 1 + a_1 z^{-1} + a_2 z^{-2}$$

Using the quadratic formula, this system has complex poles when

$$a_1^2 < 4a_2$$

as long as a_2 is positive. Using $a_1 = k_1(1+k_2)$ and $a_2 = k_2$, we have

$$0 < a_2 < 1 \Rightarrow 0 < k_2 < 1 \text{ and } a_1^2 < 4a_2 \Rightarrow k_1^2 < \frac{4k_2}{(1+k_2)^2}$$

which proves the above equations. The Matlab script file to plot Figure 4.8a is given below and the plot is shown in Figure 4.18a.

```
% (a) Plot in Figure 4.8a
k2 = 0:0.01:1; k1 = 2*sqrt(k2)./(1+k2);
k2 = [fliplr(k2(2:end)),k2,1]; k1 = [-fliplr(k1(2:end)),k1,-1];

subplot(1,2,1);
fill([-1,1,1,-1,-1],[-1,-1,1,1,-1],[0.5,0.5,0.5]); hold on;
fill(k1,k2,[0.8,0.8,0.8]);
```

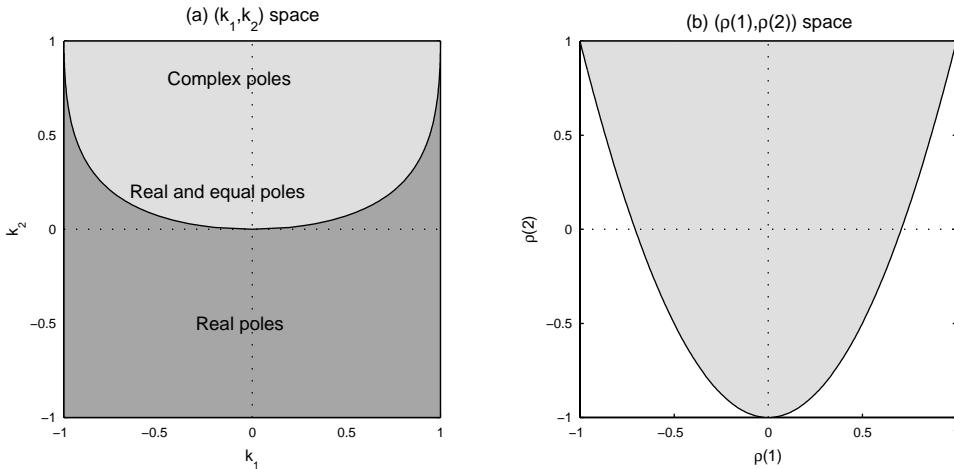


Figure 4.18: Minimum-phase and positive definiteness regions for the AP(2) model

```

plot([-1,1],[0,0], 'w:', [0,0],[-1,1], 'w:');
set(gca,'xtick',[-1:0.5:1], 'ytick', [-1:0.5:1]);
xlabel('k_1', 'fontsize', label_fontsize);
ylabel('k_2', 'fontsize', label_fontsize);
axis([-1,1,-1,1]); axis('square');
text(-0.3,-0.5,'Real poles');
text(-0.45,0.8,'Complex poles');
text(-0.65,0.2,'Real and equal poles');
title('(a) (k_1,k_2) space', 'fontsize', title_fontsize);
hold off;

```

(b) The Matlab script file to plot Figure 4.8b is given below and the plot is shown in Figure 4.18b.

```

% (b) Plot in Figure 4.8b
rho1 = -1:0.01:1;
rho2 = 2*rho1.*rho1-1;

subplot(1,2,2);
fill([rho1,-1],[rho2,1],[0.8,0.8,0.8]); hold on;
plot([-1,1],[0,0], 'w:', [0,0],[-1,1], 'w:');
set(gca,'xtick',[-1:0.5:1], 'ytick', [-1:0.5:1]);
xlabel('\rho(1)', 'fontsize', label_fontsize);
ylabel('\rho(2)', 'fontsize', label_fontsize);
axis([-1,1,-1,1]); axis('square');
title('(b) (\rho(1), \rho(2)) space', 'fontsize', title_fontsize);
hold off;

```

4.19 Given an AR(2) process $x(n)$ with $d_0 = 1$, $a_1 = -1.6454$, $a_2 = 0.9025$, and $w(n) \sim \text{WGN}(0,1)$

- (a) Figure 4.19a shows a realization of $x(n)$
- (b) Figure 4.19b shows the estimate of the ACS $\hat{\rho}_x(l)$ and theoretical values
- (c) Compute the system parameters from the ACS $\hat{\rho}_x(l)$. This can be done using the Yule-Walker equations.

$$\hat{a} = -\hat{R}_x^{-1}\hat{r}_x$$

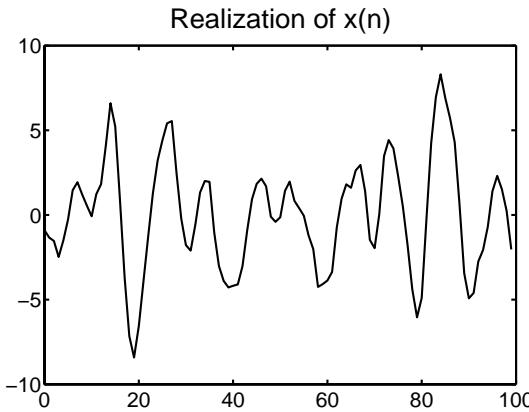


Figure 4.19a:

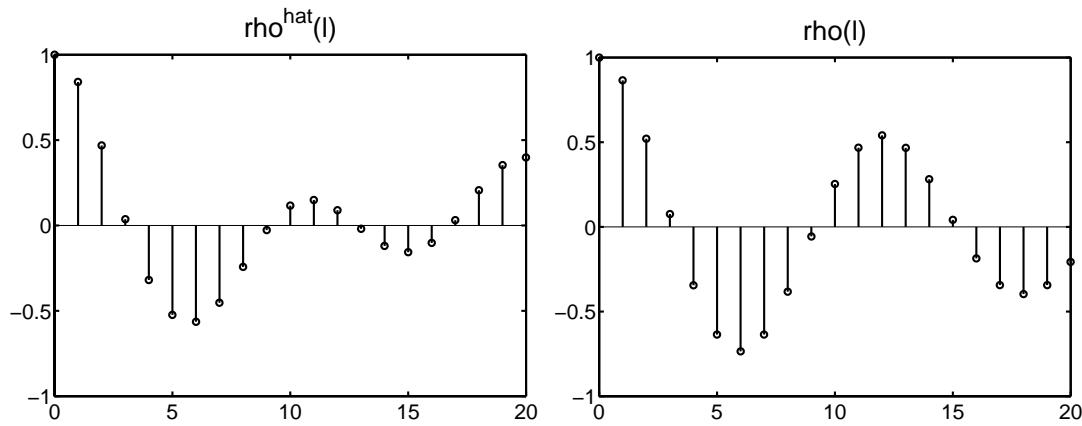


Figure 4.19b:

Figure 4.19c shows the estimate of the system parameters.

- (d) Figure 4.19d shows the estimated and true PSDs of the process
- (e) Figure 4.19e shows the PACS

- 4.20** Find a minimum-phase model with autocorrelation $\rho(0) = 1$, $\rho(\pm 1) = 0.25$, and $\rho(l) = 0$ for $|l| \geq 2$. Using Equation 4.3.13

$$\rho(l) = \begin{cases} 1 & l = 0 \\ \frac{d_1}{1+d_1^2} & l = \pm 1 \\ 0 & |l| \geq 2 \end{cases}$$

Clearly,

$$\frac{d_1}{1+d_1^2} = 0.25$$

and solving for d_1 results in $d_1 = 0.2679$. Inserting d_1 into the general model equation generates a minimum-phase model of

$$x(n) = w(n) + 0.2679w(n-1)$$

- 4.21** Given the MA(2) model $x(n) = w(n) - 0.1w(n-1) + 0.2w(n-2)$

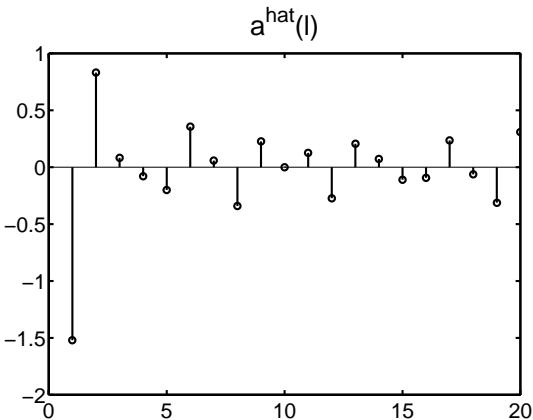


Figure 4.19c:

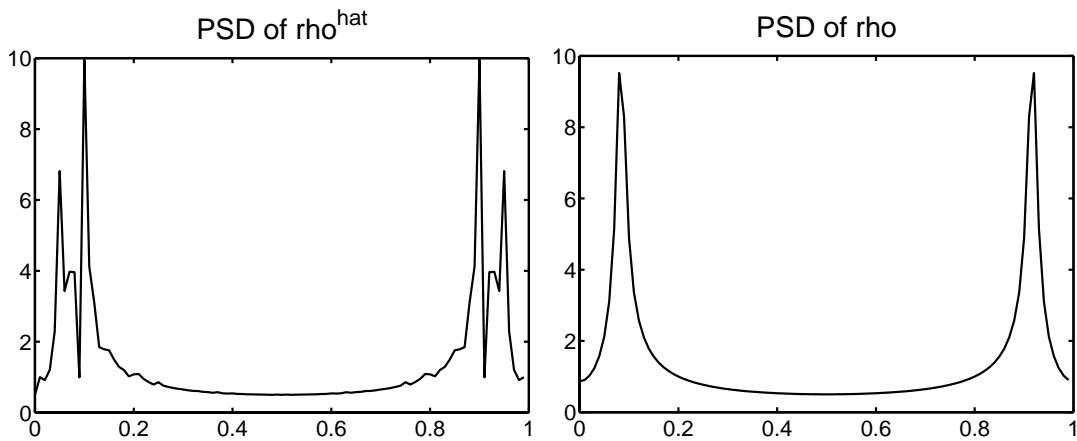


Figure 4.19d:

- (a) Is the process $x(n)$ stationary?

The process $x(n)$ is a linear combination of a stationary process $w(n)$, therefore $x(n)$ is stationary.

- (b) Is the model minimum-phase?

Looking at the transfer function

$$H(z) = 1 - 0.1z^{-1} + 0.2z^{-2}$$

This has $d_1 = -0.1$ and $d_2 = 0.2$. Using Equation 4.3.21, which are the minimum-phase conditions for an AZ(2) model

$$\begin{aligned} d_2 &< |1| \\ d_2 - d_1 &> -1 \\ d_2 + d_1 &> -1 \end{aligned}$$

Clearly, using the above equations, the model is stationary.

- (c) Determine the autocorrelation and partial autocorrelation of the process.

The autocorrelation can be found directly using Equation 4.3.22

$$\rho(l) = \begin{cases} 1 & l = 0 \\ \frac{d_1(1+d_2)}{1+d_1^2+d_2^2} = -0.114 & l = \pm 1 \\ \frac{d_2}{1+d_1^2+d_2^2} = 0.191 & l = \pm 2 \\ 0 & |l| > 3 \end{cases}$$

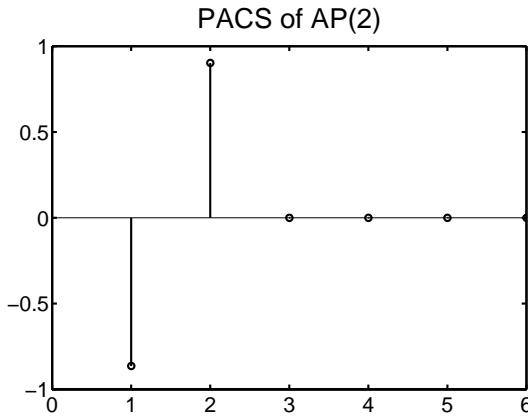


Figure 4.19e:

The partial autocorrelation sequence can be computed using the Yule-Walker equations and Cramer's rule. Where the PACS is

$$a_m^{(m)} = -\frac{|P_m^{(m)}|}{|P_m|}$$

and $P_m^{(m)}$ and P_m are defined below as

$$P_m^{(m)} = \begin{bmatrix} \rho_x(0) & \rho_x(1) & \cdots & \rho_x(m-1) & \rho_x(1) \\ \rho_x^*(1) & \rho_x(0) & \ddots & \rho_x(m-2) & \rho_x(2) \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \rho_x^*(m) & \rho_x^*(m-1) & \cdots & \rho_x(1) & \rho_x(m+1) \end{bmatrix}$$

$$P_m = \begin{bmatrix} \rho_x(0) & \rho_x(1) & \cdots & \rho_x(m) \\ \rho_x^*(1) & \rho_x(0) & \ddots & \rho_x(m-1) \\ \vdots & \ddots & \ddots & \vdots \\ \rho_x^*(m) & \rho_x^*(m-1) & \cdots & \rho_x(0) \end{bmatrix}$$

Using the above equation, the first five coefficients of the PACS are

$$a_m^{(m)} = [0.1140, -0.1803, -0.0402, 0.0321, 0.0113]$$

and Figure 4.21c shows the first 20 coefficients.

4.22 Given two ARMA models

(a) A general expression for the autocorrelation can be found directly

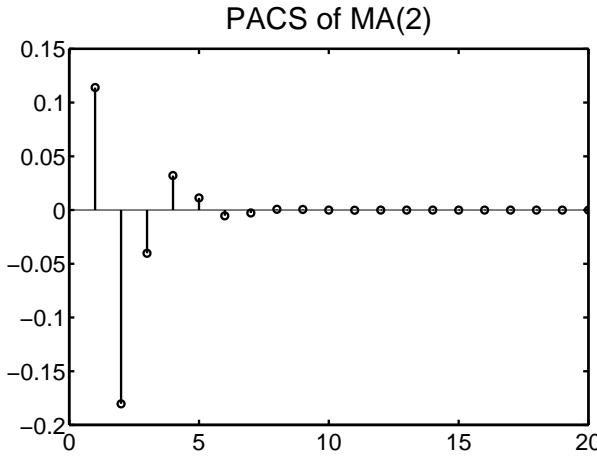
$$\text{i. } x(n) = 0.6x(n-1) + w(n) - 0.9w(n-1)$$

$$\begin{aligned} r_x(0) &= E\{x(n)x(n)\} = E\{[0.6x(n-1) + w(n) - 0.9w(n-1)][0.6x(n-1) + w(n) - 0.9w(n-1)]\} \\ &= \frac{1 + (0.9)^2}{1 - (0.6)^2} = 2.828 \end{aligned}$$

$$\begin{aligned} r_x(1) &= E\{x(n-1)x(n)\} = E\{[x(n-1)][0.6x(n-1) + w(n) - 0.9w(n-1)]\} \\ &= 0.6r_x(0) = 1.697 \end{aligned}$$

$$\begin{aligned} r_x(2) &= E\{x(n-2)x(n)\} = E\{[x(n-2)][0.6x(n-1) + w(n) - 0.9w(n-1)]\} \\ &= 0.6r_x(1) = 0.6^2r_x(0) = 1.018 \end{aligned}$$

$$\text{Therefore, } r_x(l) = (0.6)^l r_x(0) = 0.6^l (2.828)$$

**Figure 4.21c:**

ii. $x(n) = 1.4x(n-1) - 0.6x(n-2) + w(n) - 0.8w(n-1)$:

$$\begin{aligned} r_x(0) &= E\{x(n)x(n)\} = E\{[1.4x(n-1) - 0.6x(n-2) + w(n) - 0.8w(n-1)] \\ &\quad \times [1.4x(n-1) - 0.6x(n-2) + w(n) - 0.8w(n-1)]\} \\ &= r_x(0)(1.4^2 + 0.6^2) - r_x(1)(2)(1.4)(0.6) + 1 + 0.8^2 \end{aligned}$$

$$\begin{aligned} r_x(1) &= E\{[x(n-1)][1.4x(n-1) - 0.6x(n-2) + w(n) - 0.8w(n-1)]\} \\ &= \frac{1.4r_x(0)}{1+0.6} = 0.875r_x(0) \end{aligned}$$

Hence $r_x(0) = \frac{1+0.8^2}{1-1.4^2-0.6^2+2(1.4)(0.6)(0.875)} = 10.93$ and $r_x(1) = (0.875)(10.93) = 9.567$. Finally

$$\begin{aligned} r_x(2) &= E\{[x(n-2)][1.4x(n-1) - 0.6x(n-2) + w(n) - 0.8w(n-1)]\} \\ &= 1.4r_x(1) - 0.6r_x(0) = 6.836 \end{aligned}$$

Therefore, $r_x(l) = 1.4r_x(l-1) - 0.6r_x(l-2)$.

(b) Compute the partial autocorrelation

The PACS can be computed using Yule-Walker equations and Cramer's rule. The first three coefficient (k_1, k_2, k_3) of the PACS can be found using the following three equations

$$k_1 = -\rho(1) = -r_x(1)/r_x(0), \quad k_2 = -\frac{\begin{vmatrix} r_x(0) & r_x(1) \\ r_x(1) & r_x(2) \end{vmatrix}}{\begin{vmatrix} r_x(0) & r_x(1) \\ r_x(1) & r_x(0) \end{vmatrix}}, \quad k_3 = -\frac{\begin{vmatrix} r_x(0) & r_x(1) & r_x(1) \\ r_x(1) & r_x(0) & r_x(2) \\ r_x(2) & r_x(1) & r_x(3) \end{vmatrix}}{\begin{vmatrix} r_x(0) & r_x(1) & r_x(2) \\ r_x(1) & r_x(0) & r_x(1) \\ r_x(2) & r_x(1) & r_x(0) \end{vmatrix}}$$

i. The first four coefficients of the ACS for $x(n) = 0.6x(n-1) + w(n) - 0.9w(n-1)$ are

$$r_x = [2.828, 1.697, 1.018, 0.611]$$

Substituting these values into the above equations for the PACS, the resulting PACS coefficients are

$$k_m = [-0.6, 0, 0]$$

- ii. The first four coefficients of the ACS for $x(n) = 1.4x(n-1) - 0.6x(n-2) + w(n) - 0.8w(n-1)$ are

$$r_x = [10.93, 9.567, 6.836, 3.830]$$

Substituting these values into the above equations for the PACS, the resulting PACS coefficients are

$$k_m = [-0.8753, 0.6017, 0]$$

- (c) Generate a realization of $x(n)$ and compute $\hat{\rho}(l)$

- i. Figure 4.22c.1 shows a realization of $x(n) = 0.6x(n-1) + w(n) - 0.9w(n-1)$ and $\hat{\rho}(l)$

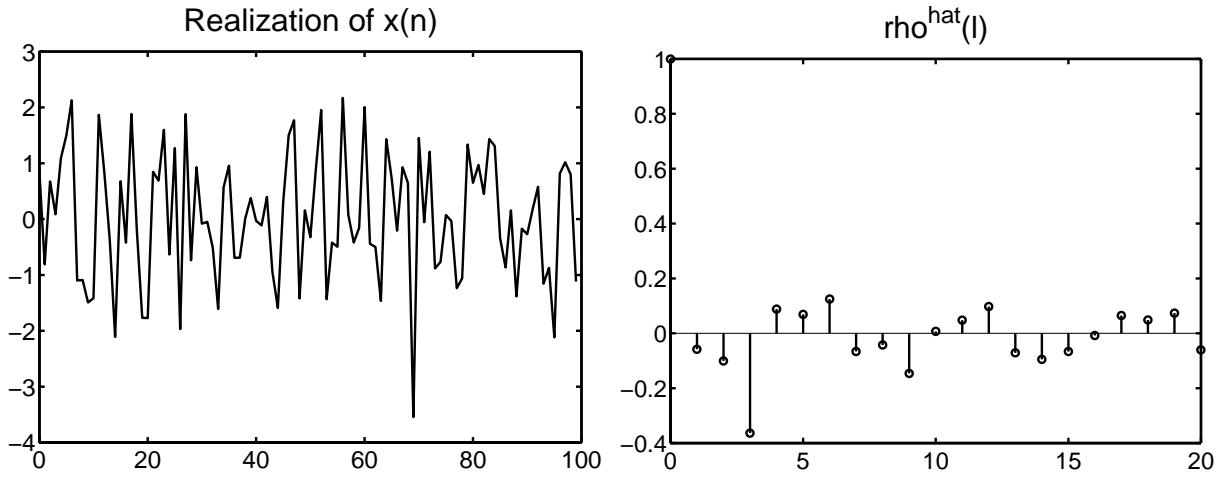


Figure 4.22c.1:

- ii. Figure 4.22c.2 shows a realization of $x(n) = 1.4x(n-1) - 0.6x(n-2) + w(n) - 0.8w(n-1)$ and $\hat{\rho}(l)$

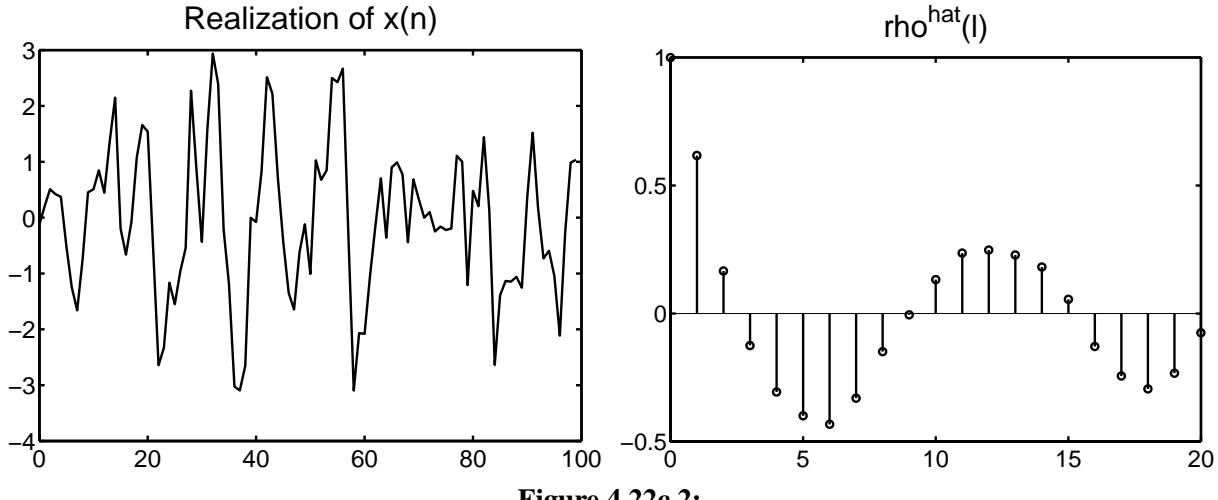


Figure 4.22c.2:

- (d) Figure 4.22d shows the estimate of \hat{k}_m . The left and right plot correspond to $x(n) = 0.6x(n-1) + w(n) - 0.9w(n-1)$ and $x(n) = 1.4x(n-1) - 0.6x(n-2) + w(n) - 0.8w(n-1)$ respectively.

4.23 see Example 4.4.1, page 179

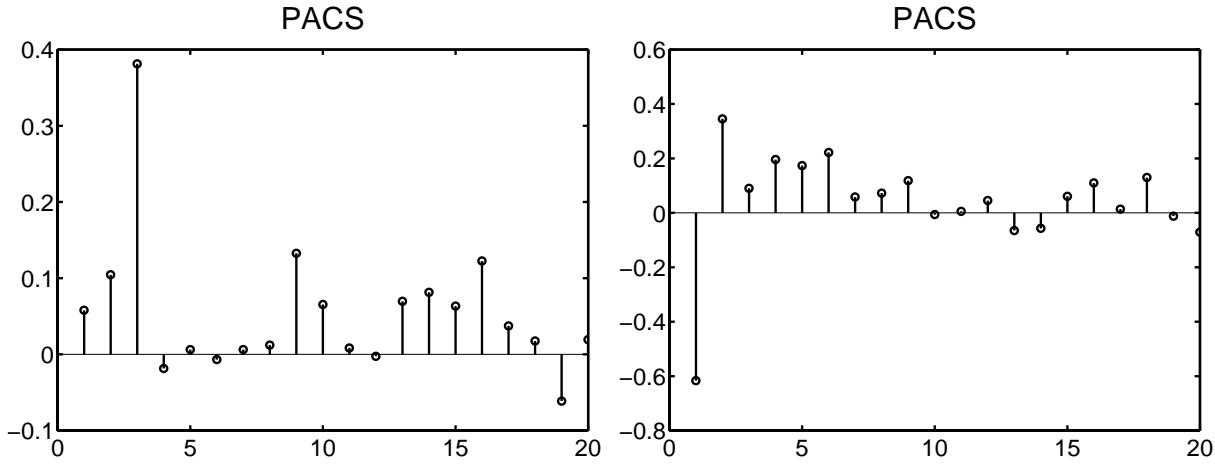


Figure 4.22d:

4.24 Given an AZ(Q) model

- (a) Show that the impulse response of an AZ(Q) model can be recovered from its response $\tilde{h}(n)$ to a periodic train with period L if $L > Q$

Assume the $x(n)$ has the general form

$$x(n) = w(n) + d_1 w(n-1) + d_2 w(n-2) + \dots + d_q w(n-q)$$

The impulse response to a periodic input is $\tilde{h}(n)$. Since the model is an AZ(Q) model, the impulse response will be zero for each point beyond Q taps. If the period of the periodic input pulses is greater than the order of the model, then the system response to each periodic impulse is

$$\tilde{h}(n) = \sum_{k=0}^q d_k h(n-k) + \sum_{k=q+1}^L 0$$

and the AZ(Q) model can be completely recovered from $\tilde{h}(n)$.

- (b) Show that the ACS of an AZ(Q) model can be recovered from the ACS or spectrum of $\tilde{h}(n)$ if $L \geq 2Q+1$
Start with the definition of the ACS

$$\tilde{r}_h(l) = \sum_{n=-\infty}^{\infty} \tilde{h}(n) \tilde{h}^*(n-l)$$

Since $\tilde{h}(n)$ is periodic, $\tilde{r}_h(l)$ is also periodic. The ACS is similar to convolution, with a signal being convolved with itself. As long as the period of the periodic impulses is greater than the aliasing point of the convolution, then the signal can be completely recovered. In order for there to be no overlap, the period must be greater than twice the length of the filter plus one. This can be seen by substituting directly into the above equation

$$\tilde{r}_h(l) = \sum_{n=-\infty}^{\infty} \left(\sum_{k=0}^q d_k h(n-k) + \sum_{k=q+1}^L 0 \right) \left(\sum_{k=0}^q d_k h(n-l-k) + \sum_{k=q+1}^L 0 \right)$$

These two signals do not overlap as long as $L \geq 2Q+1$.

4.25 To prove (4.3.17), note that

$$\rho(0) = 1, \quad \rho(1) = \rho(-1) = \frac{d_1}{1+d_1^2}, \quad \text{and } \rho(l) = 0, \quad |l| \geq 2 \quad (23)$$

and that

$$\mathbf{R}_m = \begin{bmatrix} 1 & \rho(1) & 0 & \cdots & 0 & 0 \\ \rho(1) & 1 & \rho(1) & \cdots & 0 & 0 \\ 0 & \rho(1) & 1 & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \ddots & 1 & \rho(1) \\ 0 & 0 & 0 & \cdots & \rho(1) & 1 \end{bmatrix} \quad (24)$$

is an $m \times m$ matrix and that the PACS (or the reflection coefficients) are computed using (4.2.23), that is,

$$\mathbf{R}_m \begin{bmatrix} a_1^{(m)} \\ a_2^{(m)} \\ \vdots \\ k_m \end{bmatrix} = - \begin{bmatrix} \rho(1) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (25)$$

We will consider explicit evaluations for $m = 1, 2$, and 3 .

- $m = 1$: Then from (24) and (25)

$$[1] k_1 = -\rho(1) \Rightarrow k_1 = -\frac{d_1}{1+d_1^2} = \frac{(1-d_1^2)(-d_1)^1}{(1-d_1^2)(1+d_1^2)} = \frac{(1-d_1^2)(-d_1)^1}{(1+d_1^{2(1+1)})} \quad (26)$$

using (23).

- $m = 2$: Then from (24) and (25)

$$\begin{bmatrix} 1 & \rho(1) \\ \rho(1) & 1 \end{bmatrix} \begin{bmatrix} a_1^{(2)} \\ k_2 \end{bmatrix} = - \begin{bmatrix} \rho(1) \\ 0 \end{bmatrix}$$

or

$$\begin{aligned} k_2 &= \frac{\begin{vmatrix} 1 & -\rho(1) \\ \rho(1) & 0 \end{vmatrix}}{\begin{vmatrix} 1 & \rho(1) \\ \rho(1) & 1 \end{vmatrix}} = \frac{\rho^2(1)}{1-\rho^2(1)} = \frac{\left(\frac{d_1}{1+d_1^2}\right)^2}{1-\left(\frac{d_1}{1+d_1^2}\right)^2} = \frac{d_1^2}{1+d_1^2+d_1^4} \\ &= \frac{(1-d_1^2)(-d_1)^2}{(1-d_1^2)(1+d_1^2+d_1^4)} = \frac{(1-d_1^2)(-d_1)^2}{1-d_1^6} = \frac{(1-d_1^2)(-d_1)^2}{1-d_1^{2(2+1)}} \end{aligned} \quad (27)$$

- $m = 3$: Then from (24) and (25)

$$\begin{bmatrix} 1 & \rho(1) & 0 \\ \rho(1) & 1 & \rho(1) \\ 0 & \rho(1) & 1 \end{bmatrix} \begin{bmatrix} a_1^{(3)} \\ a_2^{(3)} \\ k_3 \end{bmatrix} = - \begin{bmatrix} \rho(1) \\ 0 \\ 0 \end{bmatrix}$$

or

$$\begin{aligned}
 k_3 &= \frac{\left| \begin{array}{ccc} 1 & \rho(1) & -\rho(1) \\ \rho(1) & 1 & 0 \\ 0 & \rho(1) & 0 \end{array} \right|}{\left| \begin{array}{ccc} 1 & \rho(1) & 0 \\ \rho(1) & 1 & \rho(1) \\ 0 & \rho(1) & 1 \end{array} \right|} = -\frac{\rho^3(1)}{1 - 2\rho^2(1)} = \frac{-\left(\frac{d_1}{1+d_1^2}\right)^3}{1 - 2\left(\frac{d_1}{1+d_1^2}\right)^2} = \frac{-d_1^3}{(1+d_1^4)(1+d_1^2)} \\
 &= \frac{(1-d_1^2)(-d_1)^3}{(1-d_1^2)(1+d_1^4)(1+d_1^2)} = \frac{(1-d_1^2)(-d_1)^3}{1-d_1^8} = \frac{(1-d_1^2)(-d_1)^3}{1-d_1^{2(3+1)}} \tag{28}
 \end{aligned}$$

Thus from (26), (27) and (28) we conclude that

$$k_m = \frac{(1-d_1^2)(-d_1)^m}{1-d_1^{2(m+1)}}$$

For a general proof, consider (24). Then

$$\begin{aligned}
 \det \mathbf{R}_m &= (1) \det \left(\begin{bmatrix} 1 & \rho(1) & \cdots & 0 & 0 \\ \rho(1) & 1 & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \ddots & 1 & \rho(1) \\ 0 & 0 & \cdots & \rho(1) & 1 \end{bmatrix} \right) \\
 &\quad - \rho(1)\rho(1) \det \left(\begin{bmatrix} 1 & \ddots & 0 & 0 \\ \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & 1 & \rho(1) \\ 0 & \cdots & \rho(1) & 1 \end{bmatrix} \right)
 \end{aligned}$$

where the first determinant is of $(m-1) \times (m-1)$ matrix \mathbf{R}_{m-1} and the second determinant is of $(m-2) \times (m-2)$ matrix \mathbf{R}_{m-2} . Hence

$$\det \mathbf{R}_m = \det \mathbf{R}_{m-1} - \rho^2(1) \det \mathbf{R}_{m-2} \tag{29}$$

Also from (25) [and from (26), (27), and (28)], we have

$$k_m = \frac{-\rho^m(1)}{\det \mathbf{R}_m} \tag{30}$$

Dividing (29) by $-\rho^m(1)$ and using (30), we have

$$\frac{1}{k_m} = \frac{1}{-\rho(1)} \frac{1}{k_{m-1}} - \frac{1}{k_{m-2}}, \quad m \geq 3 \tag{31}$$

which is a recursive relation with initial conditions

$$k_1 = -\rho(1) = -\frac{d_1}{1+d_1^2} \text{ and } k_2 = \frac{\rho^2(1)}{1-\rho^2(1)} = \frac{d_1^2}{1+d_1^2+d_1^4} \tag{32}$$

Since the AZ(1) model is minimum phase, $|d_1| < 1$ which implies that the roots of the difference equation (31) in $\frac{1}{k_m}$ are real. Using $p(n) = \frac{1}{k_m}$, $n \geq 0$ with $p(-1) = \frac{1}{k_2}$, $p(-2) = \frac{1}{k_1}$, and using the unilateral z -transform approach along with fair amount of algebra, we obtain

$$k_m = \frac{(1 - d_1^2)(-d_1)^m}{1 - d_1^{2(m+1)}}, \quad m \geq 1.$$

The validity for a particular d_1 for lower values of m is established in (26) through (28). Follow that approach for $H(z) = 1 - 0.8z^{-1}$.

4.26 Prove Equation (4.3.24) that describe minimum-phase region of the AZ(2) model. Equation (4.3.24) is

$$\begin{aligned} \rho(2) + \rho(1) &> -0.5 \\ \rho(2) - \rho(1) &> -0.5 \\ \rho^2(1) &= 4\rho(2)[1 - 2\rho(2)] \end{aligned}$$

Using Equation (4.3.22) and direct substitution into $\rho(2) + \rho(1) > -0.5$ yields

$$\begin{aligned} \frac{d_2}{1 + d_1^2 + d_2^2} + \frac{d_1(1 + d_2)}{1 + d_1^2 + d_2^2} &> -0.5 \\ (d_1 + d_2)^2 + 2(d_1 + d_2) + 1 &> 0 \end{aligned}$$

Clearly, in order for the above equation to hold, $d_1 + d_2 > -1$, which is the minimum-phase region described in Equation (4.3.21).

Using Equation (4.3.22) and direct substitution into $\rho(2) - \rho(1) > -0.5$ yields

$$\begin{aligned} \frac{d_2}{1 + d_1^2 + d_2^2} - \frac{d_1(1 + d_2)}{1 + d_1^2 + d_2^2} &> -0.5 \\ (d_1 - d_2)^2 + 2(d_1 - d_2) + 1 &> 0 \end{aligned}$$

Clearly, in order for the above equation to hold, $d_1 - d_2 > -1$, which is the minimum-phase region described in Equation (4.3.21).

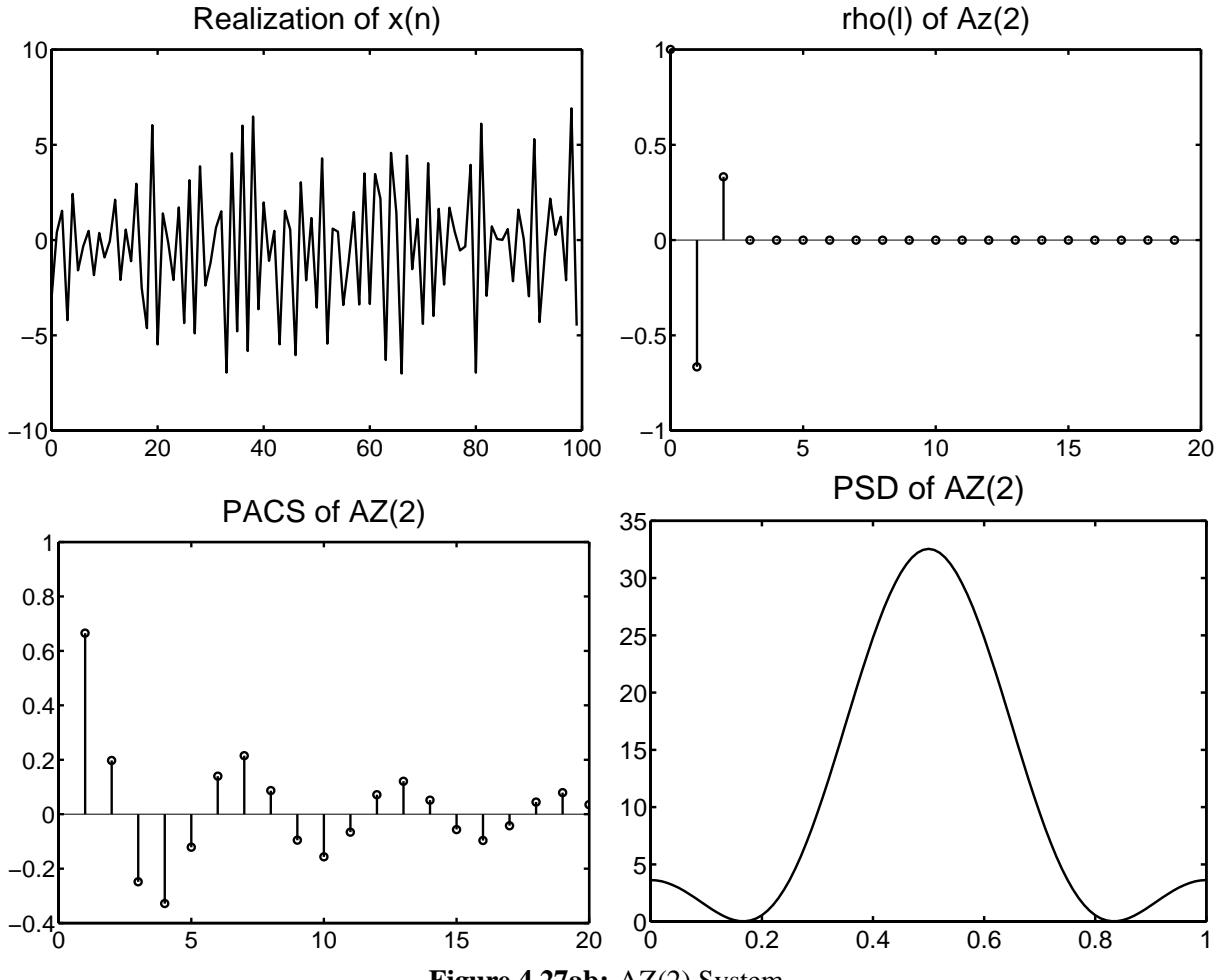
4.27 Given an AZ(2) model with $d_0 = 2$ and zeros $z_{1,2} = 0.95e^{\pm j\pi/3}$

- (a) The upper left plot of Figure 4.27ab shows a realization of $x(n) = -0.95x(n-1) + 0.9025x(n-2) + w(n)$
- (b) Figure 4.27ab shows the ACS, PACS, and the spectrum of the model
- (c) Repeating parts a and b above with an AP(2) model with poles $p_{1,2} = 0.95e^{\pm j\pi/3}$. Figure 4.27c has the plots for the realization, ACS, PACS, and spectrum to the AP(2) model.
- (d) The duality of the AZ(2) and AP(2) is shown in the above figures. The finite ACS for the AZ(2) model is similar to the finite PACS of the AP(2) model. The periodic nature of the PACS for the AZ(2) also appears in the ACS of the AP(2) model. Lastly, frequency of notch filter for the AZ(2) corresponds to the frequency peaks of the AP(2) model.

4.28 Given the autocorrelation sequence relations in (4.4.29) and (4.4.30), that is,

$$\rho(1) = \frac{(d_1 - a_1)(1 - a_1d_1)}{1 + d_1^2 - 2a_1d_1} \quad (33)$$

$$\rho(2) = -a_1\rho(1) \quad (34)$$

**Figure 4.27ab: AZ(2) System**

and the inequalities

$$-1 < d_1 < 1, \quad -1 < a_1 < 1 \quad (35)$$

we want to determine conditions on $\rho(1)$ and $\rho(2)$.

First note from (34) that

$$a_1 = -\frac{\rho(2)}{\rho(1)} \quad (36)$$

and using (35)

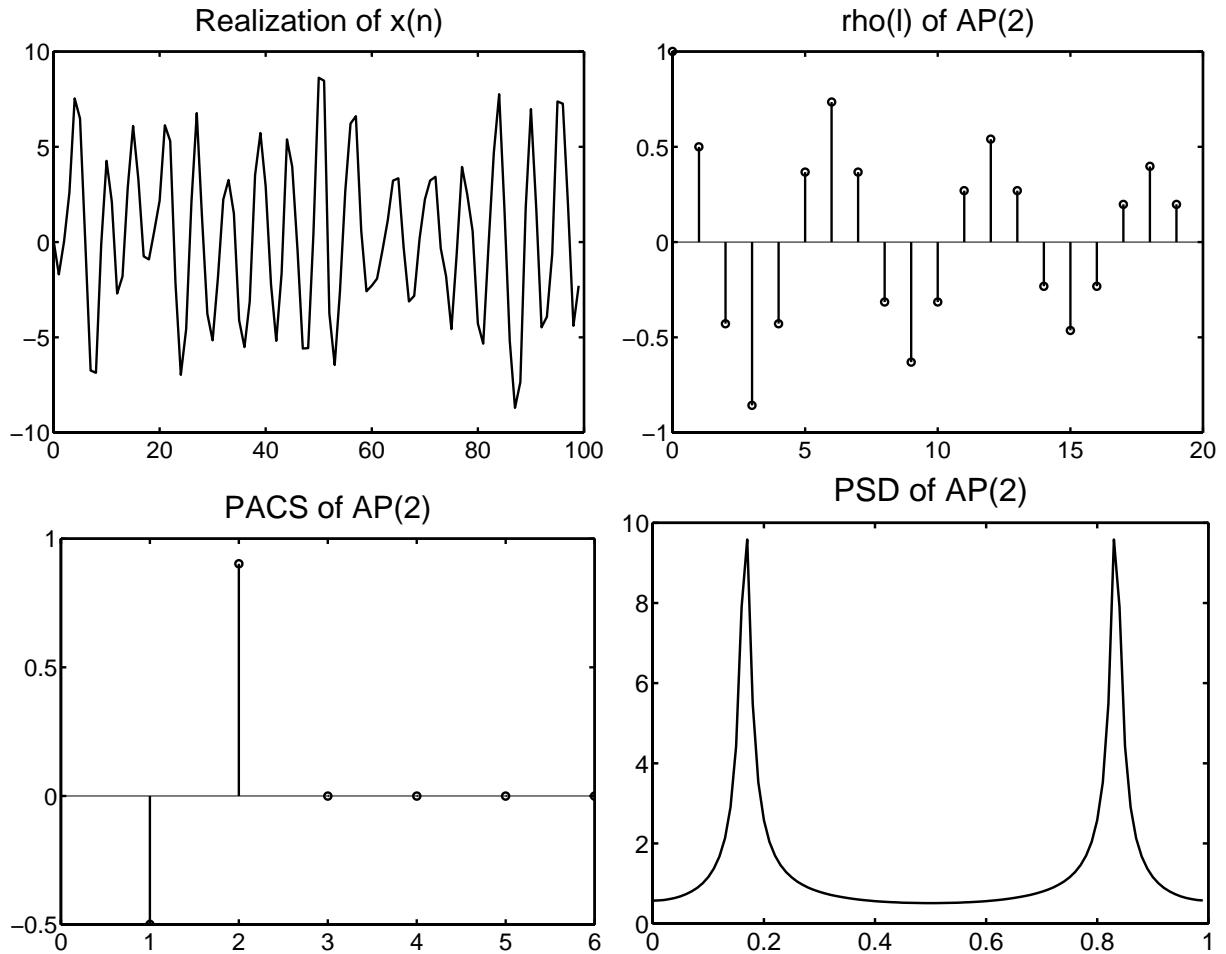
$$|a_1| < 1 \Rightarrow |\rho(2)| < |\rho(1)| \quad (37)$$

Now for the remaining conditions the general approach is the following: first substitute a_1 from (36) in (33) and solve for d_1 in terms of $\rho(1)$ and $\rho(2)$; then use (35) to determine conditions on $\rho(1)$ and $\rho(2)$. Consider (33). After simplification we obtain

$$\rho(1) + \rho(1)d_1^2 - 2\rho(1)a_1d_1 = d_1 - a_1d_1^2 - a_1 + a_1^2d_1$$

Substituting a_1 from (36) and simplifying, we obtain

$$\rho^3(1) + \rho^3(1)d_1^2 + 2\rho^2(1)\rho(2)d_1 = [\rho^2(1)d_1 + \rho(1)\rho(2)d_1^2 + \rho(1)\rho(2) + \rho^2(2)d_1]$$

**Figure 4.27c:** AP(2) System

or

$$d_1^2 [\rho^3(1) - \rho(1)\rho(2)] + d_1 [2\rho^2(1)\rho(2) - \rho^2(1) - \rho^2(2)] + [\rho^3(1) - \rho(1)\rho(2)] = 0 \quad (38)$$

Solving (38) for d_1 , we obtain

$$d_1 = \frac{1}{2(-b^3 + bc)} \left(2b^2c - b^2 - c^2 + \sqrt{(4b^4c^2 + 4b^4c - 4b^2c^3 + b^4 - 2b^2c^2 + c^4 - 4b^6)} \right) \quad (39)$$

and

$$d_1 = \frac{1}{2(-b^3 + bc)} \left(2b^2c - b^2 - c^2 - \sqrt{(4b^4c^2 + 4b^4c - 4b^2c^3 + b^4 - 2b^2c^2 + c^4 - 4b^6)} \right) \quad (40)$$

where we have used $b = \rho(1)$ and $c = \rho(2)$ for compactness. Now since $-1 < d < 1$ we have four possible equations: $d_1 = 1$ and $d_1 = -1$ in (39) and in (40). Thus

$$1 = \frac{1}{2(-b^3 + bc)} \left(2b^2c - b^2 - c^2 + \sqrt{(4b^4c^2 + 4b^4c - 4b^2c^3 + b^4 - 2b^2c^2 + c^4 - 4b^6)} \right)$$

yields $c = -b$ or $\rho(2) = -\rho(1)$ [which is part of (37)] and

$$c = 2b^2 - b \text{ or } \rho(2) = \rho(1)[2\rho(1) - 1] \quad (41)$$

Similarly

$$-1 = \frac{1}{2(-b^3 + bc)} \left(2b^2c - b^2 - c^2 + \sqrt{(4b^4c^2 + 4b^4c - 4b^2c^3 + b^4 - 2b^2c^2 + c^4 - 4b^6)} \right)$$

yields $c = b$ or $\rho(2) = -\rho(1)$ [which is part of (37)] and

$$c = 2b^2 + b \text{ or } \rho(2) = \rho(1)[2\rho(1) + 1] \quad (42)$$

Likewise from (40), we have

$$\rho(2) = -\rho(1) \text{ and } \rho(2) = \rho(1)[2\rho(1) - 1]$$

and

$$\rho(2) = \rho(1) \text{ and } \rho(2) = \rho(1)[2\rho(1) + 1]$$

which are the same conditions as those in (41) and (42). Since $|\rho(2)| < 1$, we conclude that in (41) $\rho(1)$ must be greater than 0 and in (42) $\rho(1)$ must be less than 0. Thus we conclude

$$\begin{aligned} \rho(2) &> \rho(1)[2\rho(1) - 1], \quad \rho(1) > 0 \\ \rho(2) &> \rho(1)[2\rho(1) + 1], \quad \rho(1) < 0 \end{aligned}$$

which together with (37) is (4.4.31).

Matlab Script for drawing Figure 4.12b is shown below and the resulting figure is shown in Figure 4.28.

```
% Generate figure for rho1 > 0
rho1p = 0:0.01:1;
rho2p = rho1p.*((2*rho1p)-1);
rho1p = [rho1p,0]; rho2p = [rho2p,0];

% Generate figure for rho1 < 0
rho1n = -1:0.01:0;
rho2n = rho1n.*((2*rho1n)+1);
rho1n = [rho1n,-1]; rho2n = [rho2n,1];

% Plot
fill(rho1p,rho2p,'g'); hold on;
fill(rho1n,rho2n,'r');
plot([-1,1],[0,0],'w:',[0,0],[-1,1],'w:');
set(gca,'xtick',[-1:0.5:1],'ytick',[-1:0.5:1]);
xlabel('rho(1)', 'fontsize',label_fontsize);
ylabel('rho(2)', 'fontsize',label_fontsize);
text(0.2,-0.2,'rho(2)=rho(1)[2\rho(1)-1]');
text(-0.9,-0.2,'rho(2)=rho(1)[2\rho(1)+1]');
text(0.2,0.6,'rho(2)=rho(1)');
text(-0.55,0.6,'rho(2)=-rho(1)');
axis([-1,1,-1,1]); axis('square');
```

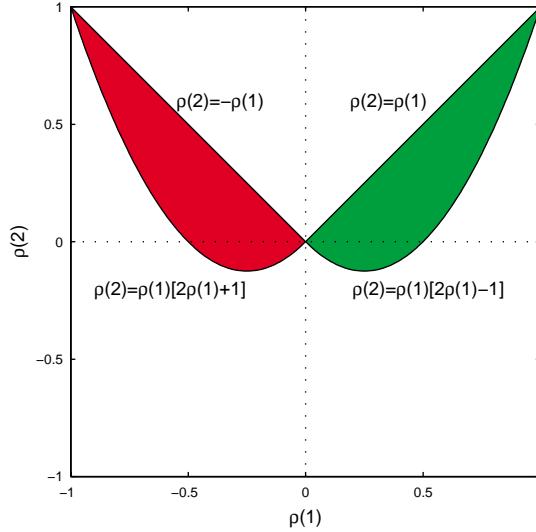


Figure 4.28: Minimum-phase and positive definite region for the PZ(1,1) model in the $\rho(1) - \rho(2)$ plane

4.29 The spectral flatness can be computed using Equation (4.1.20)

$$\text{SFM}_x = \frac{\sigma_w^2}{\sigma_x^2} = \frac{1}{\frac{1}{2\pi} \int_{-\pi}^{\pi} |H(e^{j\omega})|^2 d\omega}$$

Using Parseval's Theorem, this can be expressed as

$$\text{SFM}_x = \frac{1}{\sum_{n=-\infty}^{\infty} |h(n)|^2}$$

or in other words, one over the sum of the squared-magnitude of the impulse response.

(a) $x(n) = a_1x(n-1) + a_2x(n-2) + w(n)$

Finding the system function $H(z) = X(z)/W(z)$ is

$$H(z) = \frac{1}{1 - a_1z^{-1} - a_2z^{-2}}$$

This is an AP(2) system. The general equation for the impulse response to an AP(2) system is

$$h(n) = \frac{1}{p_1 - p_2} (p_1^{n+1} - p_2^{n+1}) u(n)$$

Therefore, the SFM_x is

$$\text{SFM}_x = \frac{1}{\sum_{n=0}^{\infty} |\frac{1}{p_1 - p_2} (p_1^{n+1} - p_2^{n+1})|^2}$$

where p_1 and p_2 are the two poles of the system.

(b) $x(n) = w(n) + b_1w(n-1) + b_2w(n-2)$

Finding the system function $H(z) = X(z)/W(z)$ is

$$H(z) = 1 + b_1z^{-1} + b_2z^{-2}$$

This is an AZ(2) system. The general equation for the impulse response to an AZ(2) system is

$$h(n) = \delta(n) + b_1\delta(n-1) + b_2\delta(n-2)$$

Therefore, the SFM_x is

$$\text{SFM}_x = \frac{1}{1 + |b_1|^2 + |b_2|^2}$$

4.30 The WSS process $x(n)$ is zero-mean with PSD $R_x(e^{j\omega})$. The matrix \mathbf{R}_x is an $M \times M$ correlation matrix with eigenvalues $\{\lambda_i\}_{i=1}^M$. Szegö's theorem states that

$$\lim_{M \rightarrow \infty} \frac{1}{M} \sum_{i=1}^M g(\lambda_i) = \frac{1}{2\pi} \int_{-\pi}^{\pi} g[R_x(e^{j\omega})] d\omega$$

Let $g(\cdot) = \ln(\cdot)$. Then

$$\lim_{M \rightarrow \infty} \frac{1}{M} \sum_{i=1}^M \ln \lambda_i = \frac{1}{2\pi} \int_{-\pi}^{\pi} \ln [R_x(e^{j\omega})] d\omega \quad (43)$$

Using the property of determinant of \mathbf{R}_x , namely $\det(\mathbf{R}_x) = \prod_{i=1}^M \lambda_i$, we can write

$$\ln [\det(\mathbf{R}_x)] = \sum_{i=1}^M \ln (\lambda_i) \quad (44)$$

Substituting (44) in (43), we obtain

$$\lim_{M \rightarrow \infty} \frac{1}{M} \ln [\det(\mathbf{R}_x)] = \frac{1}{2\pi} \int_{-\pi}^{\pi} \ln [R_x(e^{j\omega})] d\omega \quad (45)$$

Using $\frac{1}{M} \ln [\det(\mathbf{R}_x)] = \ln [\det(\mathbf{R}_x)]^{\frac{1}{M}}$, we can rewrite (45) as

$$\lim_{M \rightarrow \infty} [\det(\mathbf{R}_x)]^{\frac{1}{M}} = \exp \left\{ \frac{1}{2\pi} \int_{-\pi}^{\pi} \ln [R_x(e^{j\omega})] d\omega \right\}$$

which is the desired result.

4.31 Given the two linear random processes with system functions

$$H_1(z) = \frac{1 - 0.81z^{-1} - 0.4z^{-2}}{(1 - z^{-1})^2}$$

and

$$H_2(z) = \frac{1 - 0.5z^{-1}}{(1 - z^{-1})}$$

(a) Find a difference equation for the two system functions:

Since $H(z) = X(z)/W(z)$, the system function can be found directly

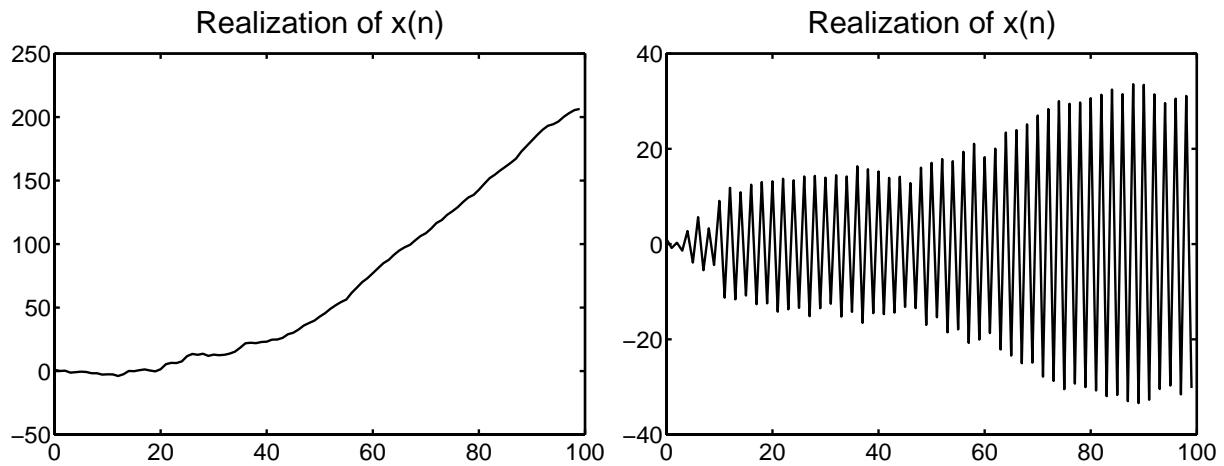


Figure 4.31b:

i. for $H_1(z) = \frac{1-0.81z^{-1}-0.4z^{-2}}{(1-z^{-1})^2}$ The system function can be separated into

$$X(z)(1 - 2z^{-1} + z^{-2}) = W(z)(1 - 0.81z^{-1} - 0.4z^{-2})$$

Taking the inverse z-transform gives a difference equation of

$$x(n) = 2x(n-1) - x(n-2) + w(n) - 0.81w(n-1) - 0.4w(n-2)$$

ii. for $H_2(z) = \frac{1-0.5z^{-1}}{(1-z^{-1})}$ The system function can be separated into

$$X(z)(1 - z^{-1}) = W(z)(1 - 0.5z^{-1})$$

Taking the inverse z-transform gives a difference equation of

$$x(n) = x(n-1) + w(n) - 0.5w(n-1)$$

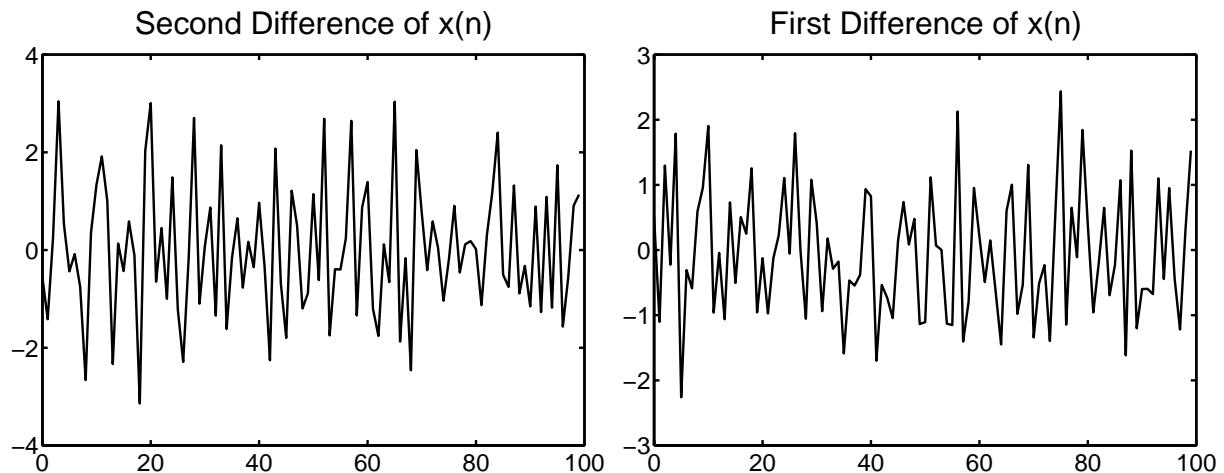


Figure 4.31c:

- (b) Figure 4.31b shows realization of $x(n) = 2x(n-1) - x(n-2) + w(n) - 0.81w(n-1) - 0.4w(n-2)$ and $x(n) = x(n-1) + w(n) - 0.5w(n-1)$ respectively. Both plots display aspects of non-stationarity. The left plot shows a random walk type system, with the mean changing with more samples. The second plot remains zero mean, but its variance is not constant with the number of samples.

(c) The first and second difference for the above two systems are shown in Figure 4.31c. Clearly, these two systems appear stationary. The second difference of $x(n) = 2x(n-1) - x(n-2) + w(n) - 0.81w(n-1) - 0.4w(n-2)$ is

$$\begin{aligned} y(n) &= x(n) - 2x(n-1) + x(n-2) \\ &= (2x(n-1) - x(n-2) + w(n) - 0.81w(n-1) - 0.4w(n-2)) - 2x(n-1) + x(n-2) \\ &= w(n) - 0.81w(n-1) - 0.4w(n-2) \end{aligned}$$

and the first difference of $x(n) = x(n-1) + w(n) - 0.5w(n-1)$ is

$$\begin{aligned} y(n) &= x(n) - x(n-1) = (x(n-1) + w(n) - 0.5w(n-1)) - x(n-1) \\ &= w(n) - 0.5w(n-1) \end{aligned}$$

4.32 Generate and plot 100 samples for each of the following linear processes and then estimate and examine the values of the ACS and PACS

(a) $H(z) = \frac{1}{(1-z^{-1})(1-0.9z^{-1})}$: The plots are shown in Figure 4.32a.

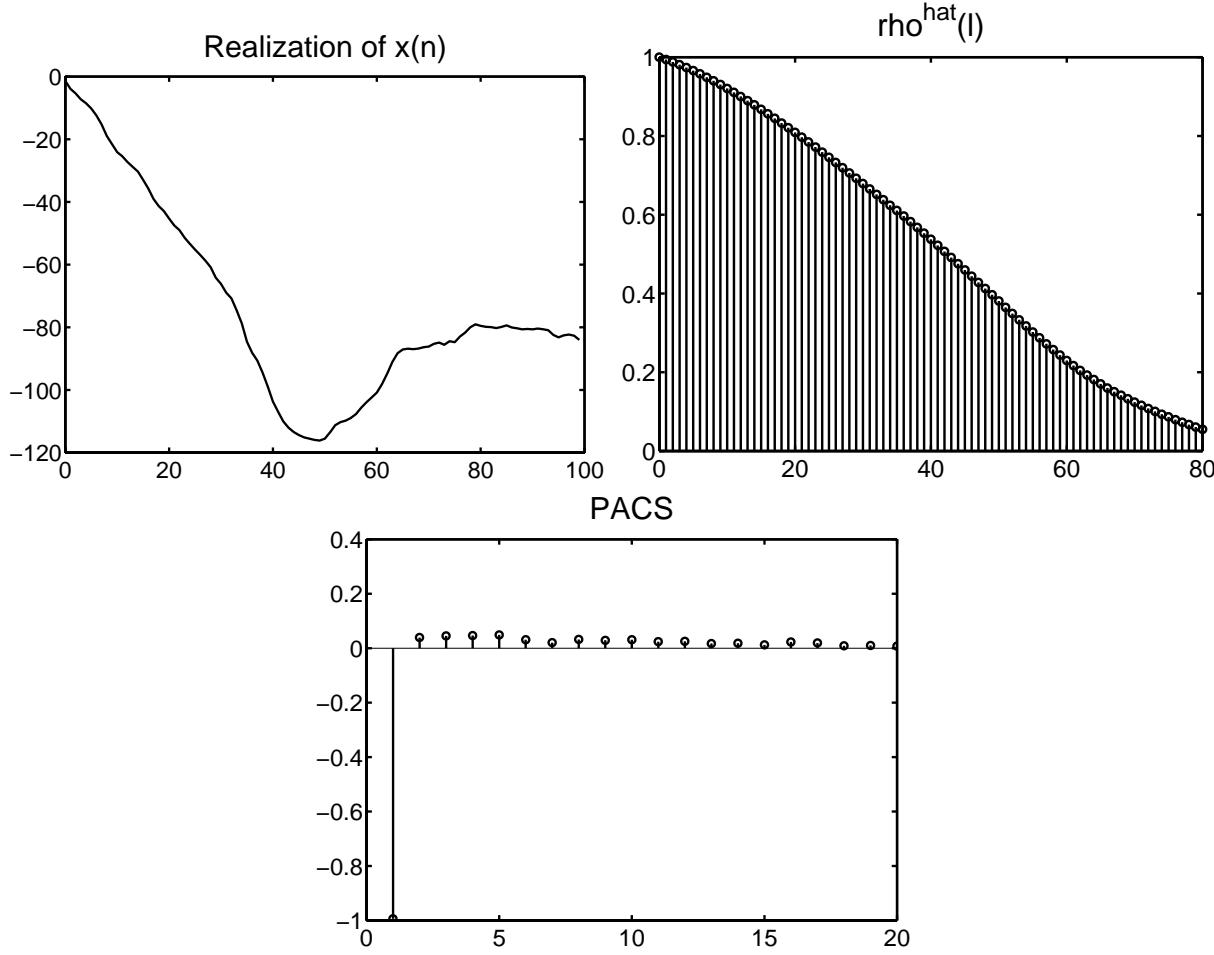
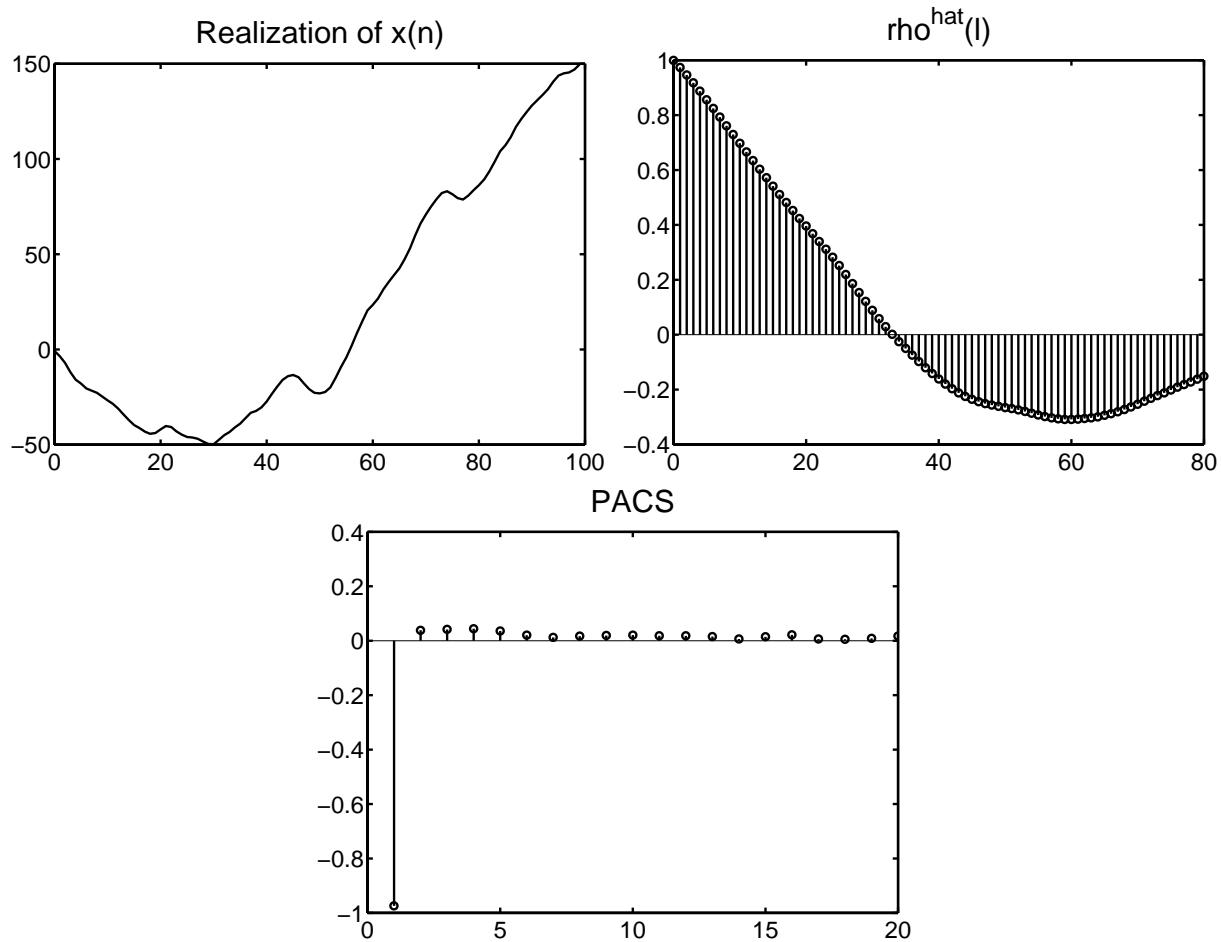


Figure 4.32a:

(b) $H(z) = \frac{1-0.5z^{-1}}{(1-z^{-1})(1-0.9z^{-1})}$: The plots are shown in Figure 4.32b.

4.33 Given $y(n) = d_0 + d_1n + d_2n^2 + x(n)$, where $x(n)$ is a stationary process with known autocorrelation $r_x(l)$

**Figure 4.32b:**

- (a) Show that the process $y^{(2)}(n)$ obtained by passing $y(n)$ through the filter $H(z) = (1 - z^{-1})^2$ is stationary.
Since $H(z) = Y^{(2)}(z)/Y(z)$, the output of the system is

$$Y^{(2)}(z) = Y(z)(1 - 2z^{-1} + z^{-2})$$

Taking the inverse z-transform

$$y^{(2)}(n) = y(n) - 2y(n-1) + y(n-2)$$

Substituting $y(n) = d_0 + d_1n + d_2n^2 + x(n)$ back into the above equation and simplifying

$$y^{(2)}(n) = 2d_2 + x(n) - 2x(n-1) + x(n-2)$$

Since the output is a linear combination of a stationary system, $y^{(2)}(n)$ is stationary.

- (b) The autocorrelation of $y^{(2)}(n)$ can be computed directly

$$\begin{aligned} r_y^{(2)}(l) &= E\{y^{(2)}(n+l)y^{(2)}(n)\} \\ &= E\{[2d_2 + x(n+l) - 2x(n+l-1) + x(n+l-2)][2d_2 + x(n) - 2x(n-1) + x(n-2)]\} \end{aligned}$$

Therefore, $r_y^{(2)}(l) = 4d_2^{(2)} + 5r_x(l) - 8r_x(l) + 2r_x(l-2)$.

4.34 Proof of (4.4.7).

Consider the AP(P) model

$$H(z) = \frac{d_0}{A(z)} = \frac{d_0}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_P z^{-P}}$$

Then the complex cepstrum is given by

$$C(z) = \log H(z) = \log d_0 - \log [1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_P z^{-P}]$$

and the cepstrum is given by

$$c(n) = \mathcal{Z}^{-1}[C(z)]$$

- $P = 1$: Then $A(z) = 1 + a_1 z^{-1}$ and

$$\log(1 + a_1 z^{-1}) = a_1 z^{-1} + \left(-\frac{1}{2}a_1^2\right)z^{-2} + \left(\frac{1}{3}a_1^3\right)z^{-3} + \left(-\frac{1}{4}a_1^4\right)z^{-4} + O(z^{-5})$$

Thus

$$\begin{aligned} c(n) &= 0, \quad n < 0; \quad c(0) = \log d_0 \\ c(1) &= -a_1, \\ c(2) &= \frac{1}{2}a_1^2 = -\frac{1}{2}(2-1)a_1(-a_1) = -\frac{1}{2}(2-1)a_1c(1) \\ c(3) &= -\frac{1}{3}a_1^3 = -\frac{1}{3}(3-1)a_1\left(\frac{1}{2}a_1^2\right) = -\frac{1}{3}(3-1)a_1c(2) \\ \vdots &= \vdots \\ c(n) &= -\frac{1}{n}(n-1)a_1c(n-1) = -\frac{1}{n} \sum_{k=1}^{P=1} (n-k)a_k c(n-k), \quad n > P \end{aligned}$$

- $P = 2$: Then $A(z) = 1 + a_1 z^{-1} + a_2 z^{-2}$ and

$$\begin{aligned} \log(1 + a_1 z^{-1} + a_2 z^{-2}) &= a_1 z^{-1} + \left(a_2 - \frac{1}{2}a_1^2\right)z^{-2} + \left(-\frac{1}{3}a_1 a_2 + \frac{1}{3}(-2a_2 + a_1^2)a_1\right)z^{-3} + \\ &\quad \left(\frac{1}{4}(-2a_2 + a_1^2)a_2 + \frac{1}{4}(3a_1 a_2 - a_1^3)a_1\right)z^{-4} + O(z^{-5}) \end{aligned}$$

Thus

$$\begin{aligned} c(n) &= 0, \quad n < 0; \quad c(0) = \log d_0 \\ c(1) &= -a_1 = -a_1 - \frac{1}{1}(1-1)a_0 c(0) \\ c(2) &= -a_2 + \frac{1}{2}a_1^2 = -a_2 - \frac{1}{2}(2-1)a_1 c(2-1) \\ &= -a_n - \frac{1}{n} \sum_{k=1}^{n-1} (n-k)a_k c(n-k), \quad n = 2 \end{aligned}$$

$$\begin{aligned}
c(3) &= -\left(-\frac{1}{3}a_1a_2 + \frac{1}{3}(-2a_2 + a_1^2)a_1\right) = -\frac{1}{3}\left(2a_1\left(-a_2 + \frac{1}{2}a_1^2\right) - a_1a_2\right) \\
&= -\frac{1}{3}\left((3-1)a_1\left(-a_2 + \frac{1}{2}a_1^2\right) + a_2(-a_1)\right) \\
&= -\frac{1}{3}((3-1)a_1c(3-1) + (3-2)a_2c(3-2)) \\
&= -\frac{1}{3}\sum_{k=1}^{P=2}(n-k)a_kc(n-k), \quad n=3 \\
&\vdots = \vdots \\
c(n) &= -\frac{1}{n}\sum_{k=1}^{P=2}(n-k)a_kc(n-k), \quad n>P
\end{aligned}$$

Thus from the above verifications, we conclude that

$$c(n) = \begin{cases} -a_n - \frac{1}{n}\sum_{k=1}^{n-1}(n-k)a_kc(n-k), & 1 \leq n \leq P \\ -\frac{1}{n}\sum_{k=1}^{P=2}(n-k)a_kc(n-k) & n > P \end{cases}$$

For a general case, consider the complex cepstrum

$$C(z) = \log H(z)$$

Differentiating with respect to z , we obtain

$$C'(z) = \frac{H'(z)}{H(z)} \Rightarrow H'(z) = C'(z)H(z)$$

or

$$zH'(z) = zC'(z)H(z)$$

Taking inverse z -transform, we get

$$n h(n) = [n c(n)] * h(n) = \sum_{k=-\infty}^{\infty} h(k)(n-k)c(n-k)$$

or after dividing by n , we obtain

$$h(n) = \frac{1}{n} \sum_{k=-\infty}^{\infty} (n-k)h(k)c(n-k) \quad (46)$$

Using the property of complex cepstrum of a minimum-phase system which states that if $H(z)$ is minimum-phase then the complex cepstrum $c(n)$ is causal, we have from (46)

$$h(n) = \frac{1}{n} \sum_{k=0}^{n-1} (n-k)h(k)c(n-k), \quad n > 0$$

since $h(n)$ is also causal and $(n-k) = 0$ for $k = n$. Thus

$$h(n) = h(0)c(n) + \frac{1}{n} \sum_{k=1}^{n-1} (n-k)h(k)c(n-k), \quad n > 0$$

Solving for $c(n)$, we obtain

$$c(n) = \frac{h(n)}{h(0)} - \frac{1}{n} \sum_{k=1}^{n-1} (n-k) \frac{h(k)}{h(0)} c(n-k), \quad n > 0 \quad (47)$$

Now for $H(z) = A(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_P z^{-P}$, we have

$$h(n) = 0, \quad n < 0; \quad h(0) = 1; \quad h(n) = a_n, \quad 1 \leq n \leq P; \quad h(n) = 0, \quad n > P \quad (48)$$

Thus in (47) we have for $1 \leq n \leq P$

$$c(n) = a_n - \frac{1}{n} \sum_{k=1}^{n-1} (n-k) a_k c(n-k), \quad 1 \leq n \leq P \quad (49)$$

Note also from (48) that for $n > P$, $h(n) = 0$ and the summation ends at P . Thus for $n > P$

$$c(n) = -\frac{1}{n} \sum_{k=1}^P (n-k) a_k c(n-k), \quad n > P \quad (50)$$

Finally, note that for $H(z) = 1/A(z)$, $c(n) \rightarrow -c(n)$. Thus in (49) and (50), we have

$$c(n) = \begin{cases} -a_n - \frac{1}{n} \sum_{k=1}^{n-1} (n-k) a_k c(n-k), & 1 \leq n \leq P \\ -\frac{1}{n} \sum_{k=1}^P (n-k) a_k c(n-k), & n > P \end{cases}$$

which completes the proof.

4.35 Consider a minimum-phase AZ (Q) model $D(z) = \sum_{k=0}^Q d_k z^{-k}$ with complex cepstrum $c(k)$. Another AZ model is created with coefficients $\tilde{d}_k = \alpha^k d_k$ and complex cepstrum $\tilde{c}(k)$.

(a) Let $0 < \alpha < 1$. Since $D(z)$ is minimum phase, we have all zeros of $D(z)$ inside the unit circle. Now

$$\tilde{D}(z) = \sum_{k=0}^Q \tilde{d}_k z^{-k} = \sum_{k=0}^Q \alpha^k d_k z^{-k} = \sum_{k=0}^Q d_k (z/\alpha)^{-k} = D(z/\alpha)$$

Thus if the zeros of $D(z)$ are at $\{z_i\}_{i=1}^Q$ then the zeros of $\tilde{D}(z)$ are at $\{\alpha z_i\}_{i=1}^Q$. Then clearly $\tilde{D}(z)$ is also minimum phase since $0 < \alpha < 1$ and furthermore, no zeros of $D(z)$ cross the unit circle to become those of $\tilde{D}(z)$. This implies that the region of convergence of $\tilde{D}(z)$ also includes the unit circle. Hence

$$\begin{aligned} \tilde{c}(k) &= \mathcal{Z}^{-1} [\log \tilde{D}(z)] = \mathcal{Z}^{-1} \left[\log \left\{ d_0 \prod_{n=1}^Q (1 - \alpha z_n z^{-1}) \right\} \right] \\ &= \begin{cases} \log |d_0|, & k = 0 \\ -\sum_{n=1}^Q \alpha^k z_n^k, & k > 0 \end{cases} = \alpha^k \begin{cases} \log |d_0|, & k = 0 \\ -\sum_{n=1}^Q z_n^k, & k > 0 \end{cases} \\ &= \alpha^k c(k) \end{aligned}$$

Note that both $c(k)$ and $\tilde{c}(k)$ are causal sequences due to minimum-phase models.

- (b) To obtain nonminimum-phase model, we need to create zeros that are inside as well as outside the unit circle. If $z_{i,\min} = \min |z_i|$ and $z_{i,\max} = \max |z_i|$, then we want

$$z_{i,\min}\alpha < 1 \text{ and } z_{i,\max}\alpha > 1$$

This gives

$$\frac{1}{z_{i,\max}} < \alpha < \frac{1}{z_{i,\min}}$$

- (c) To obtain maximum-phase model, we need to create zeros that are outside the unit circle. If $z_{i,\min} = \min |z_i|$, then we want

$$z_{i,\min}\alpha > 1$$

This gives

$$\alpha > \frac{1}{z_{i,\min}}$$

4.36 Prove Equation 4.6.27, which determines the cepstral distance in the frequency and time domains.

The cepstral distance is defined, in (4.6.27), as

$$\text{CD} \equiv \frac{1}{2\pi} \int_{-\pi}^{\pi} |\log R_1(e^{j\omega}) - \log R_2(e^{j\omega})|^2 d\omega$$

Using the power series expansion, the cepstral distance can be shown as

$$\begin{aligned} \text{CD} &= \frac{1}{2\pi} \int_{-\pi}^{\pi} \left| \sum_{n=-\infty}^{\infty} c_1(n)e^{-jn\omega} - \sum_{n=-\infty}^{\infty} c_2(n)e^{-jn\omega} \right|^2 d\omega = \frac{1}{2\pi} \int_{-\pi}^{\pi} \left| \sum_{n=-\infty}^{\infty} (c_1(n) - c_2(n))e^{-jn\omega} \right|^2 d\omega \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} (c_1(n) - c_2(n))(c_1^*(m) - c_2^*(m))e^{-j(n-m)\omega} d\omega \\ &= \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} (c_1(n) - c_2(n))(c_1^*(m) - c_2^*(m)) \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{-j(n-m)\omega} d\omega \\ &= \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} (c_1(n) - c_2(n))(c_1^*(m) - c_2^*(m))\delta(n-m) = \sum_{n=-\infty}^{\infty} |c_1(n) - c_2(n)|^2 \end{aligned}$$

Therefore, $\text{CD} = \sum_{n=-\infty}^{\infty} (c_1(n) - c_2(n))^2$, which proves (4.6.27).

Nonparametric Power Spectrum Estimation

5.1 Let $x_c(t)$, $-\infty < t < \infty$, be a continuous-time signal with Fourier transform $X_c(F)$, $-\infty < F < \infty$, and let $x(n)$ be obtained by sampling $x_c(t)$ every T per sampling interval with its DTFT $X(e^{j\omega})$.

(a) Show that the DTFT $X(e^{j\omega})$ is given by

$$X(e^{j\omega}) = F_s \sum_{l=-\infty}^{\infty} X_c(fF_s - lF_s) \quad \omega = 2\pi f \quad F_s = \frac{1}{T}$$

Start with the definition of the DTFT

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n} \quad (1)$$

Substitute $x(n) = \int_{-\infty}^{\infty} x_c(t)\delta(t - nT) dt$ into (1) results in

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} x_c(t)\delta(t - nT)e^{-j\omega n} dt \quad (2)$$

Since the delta train is zero everywhere except the point where $n = \frac{t}{T}$, (2) can be written as

$$X(e^{j\omega}) = \int_{-\infty}^{\infty} [x_c(t)] \left[\sum_{n=-\infty}^{\infty} \delta(t - nT) \right] e^{-j2\pi f F_s t} dt$$

This is the Fourier Transform between the product of two signals. The Fourier Transform of the impulse train is

$$\sum_{n=-\infty}^{\infty} \delta(t - nT) \xleftrightarrow{\mathcal{F}} 2\pi F_s \sum_{l=-\infty}^{\infty} \delta(fF_s - lF_s)$$

Therefore, the DTFT $X(e^{j\omega})$ is

$$X(e^{j\omega}) = X_c(fF_s) * F_s \sum_{l=-\infty}^{\infty} \delta(fF_s - lF_s)$$

Bringing $X_c(fF_s)$ inside the summation and using the fact that convolution with a delta result in a shift in the signal, the above equation results in

$$X(e^{j\omega}) = F_s \sum_{l=-\infty}^{\infty} X_c(fF_s - lF_s) \quad (3)$$

(b) Let

$$\tilde{X}_p(k) = X(e^{j2\pi k/N}) = F_s \sum_{l=-\infty}^{\infty} X_c\left(\frac{kF_s}{N} - lF_s\right) \quad (4)$$

Start with the definition of the IDFT($\tilde{X}_p(k)$) which is

$$x_p(n) = \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}_p(k) e^{j \frac{2\pi kn}{N}} \quad (5)$$

Substituting the equation for $\tilde{X}_p(k)$ from (4) into (5) is

$$x_p(n) = \frac{F_s}{N} \sum_{k=0}^{N-1} \sum_{l=-\infty}^{\infty} X_c\left(\frac{kF_s}{N} - lF_s\right) e^{j \frac{2\pi kn}{N}}$$

Substitute $X_c\left(\frac{kF_s}{N} - lF_s\right) = \int_{-\infty}^{\infty} X_c(f F_s - lF_s) \delta(f F_s - \frac{kF_s}{N}) df$ into the above equation and bringing the sum on k inside the integral results in

$$\frac{F_s}{N} \sum_{l=-\infty}^{\infty} \int_{-\infty}^{\infty} [X_c(f F_s - lF_s)] \left[\sum_{k=0}^{N-1} \delta(f F_s - \frac{kF_s}{N}) \right] e^{j 2\pi n f} df = \sum_{m=-\infty}^{\infty} x_c(nT - mNT) \quad (6)$$

since we have a multiplication in the frequency domain with a delta train.

5.2 MATLAB has two built in functions `bartlett` and `triang`. They each generate triangular windows with slightly different coefficients.

- (a) Figure 5.2a shows plots of both the Bartlett Window and the Triangular Window for $N = 11, 31, 51$.

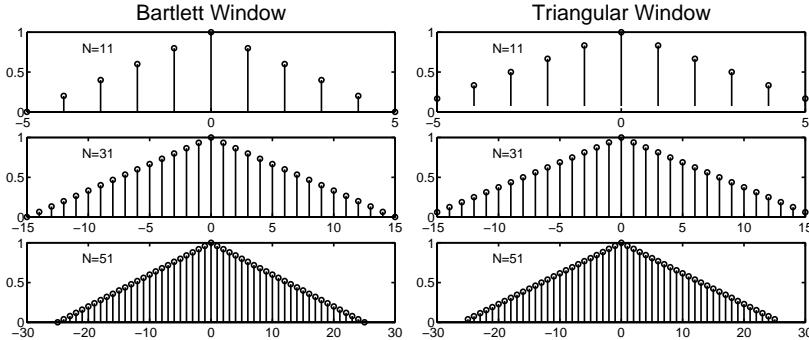


Figure 5.2a: Bartlett and Triangular Windows

- (b) Figure 5.2b show the magnitude plots for the windows given in Figure 5.2a. The mainlobe of the Triangular windows are smaller than the mainlobes of the corresponding Bartlett Windows. For each N value, the higher order lobes are spaced evenly apart for both the Bartlett and Triangular Windows.
- (c) A Bartlett Window with $N = 100$ has approximately the same mainlobe width as a rectangular window with $N = 51$. This can be seen in Figure 5.2c. This stands to reason, since a Bartlett Window is triangular in shape, and two rectangular windows convolved together results in a triangular window. Convolution in the time domain is multiplication in the frequency domain, which means that the spectrum of a triangular window is equivalent to the squared spectrum of an appropriate sized rectangular window. Lastly, when two signals of length N are convolved, the resulting output has a length of $2N - 1$. Therefore, a rectangular window of length N and a triangular window of length $2N - 1$ will have the same size mainlobe width.

5.3 This problem deals with the maximum sidelobe heights of the rectangular, Hanning, Hamming, Kaiser and Dolph-Chebyshev windows. The maximum sidelobe height is an indicator of how much spectral leakage a certain window will contribute.

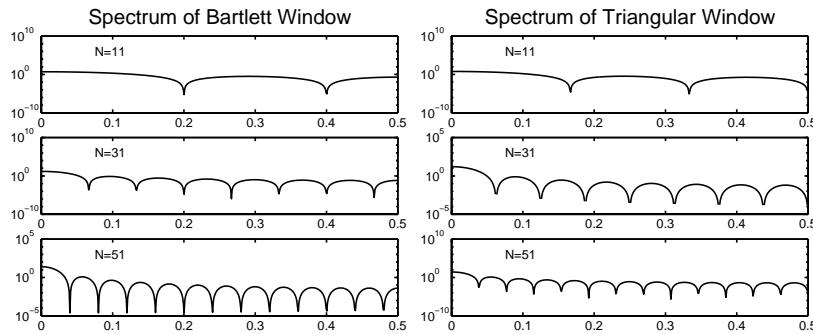


Figure 5.2b: Spectrum of Bartlett and Triangular Windows

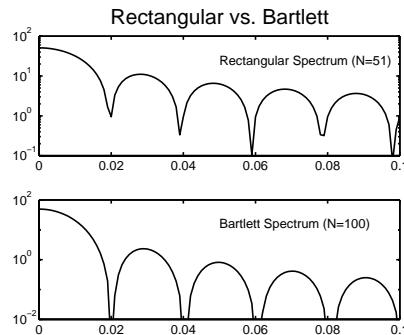


Figure 5.2c: Mainlobe Width of Bartlett and Rectangular Windows

- (a) Figure 5.3a shows magnitude spectrum plots of the rectangular, Hamming and Hanning windows for $N = 11, 31, 51$. Clearly the sidelobe height is independent of the window size.
- (b) The maximum sidelobe height of the Kaiser window is controlled by β the shape parameter. Figure 5.3b shows three plots for three different β values. Each plot has the magnitude spectrum of the Kaiser window for $N = 11, 31, 51$ shown. It is seen that the window length N does not effect the height of the sidelobe, however, the shape parameter β does.
- (c) A Kaiser window can be computed that has the same sidelobe height as the Hamming. Since the sidelobe height of the Hamming window is fixed, and from Figure 5.3a the maximum sidelobe height for a Hamming window is approximately $-42.5 dB$. The Kaiser window of $\beta = 5.8$ and length $N = 51$ has a maximum sidelobe height of approximately $-42.5 dB$. This is seen in Figure 5.3c.
- (d) The Dolph-Chebyshev has the same height A in decibels for all of the sidelobes. Figure 5.3d shows the $-3 dB$ point for three Dolph-Chebyshev windows for sidelobe height $A = 40, 50, 60 dB$.

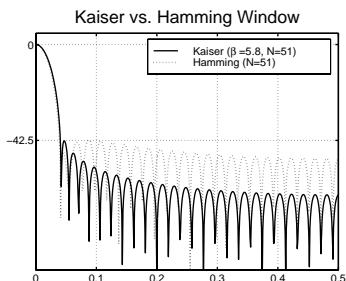


Figure 5.3c: Kaiser and Hamming Windows

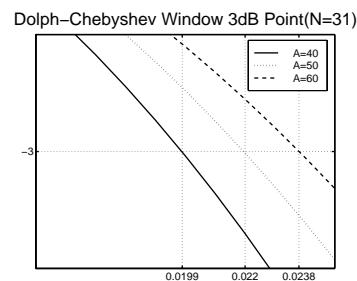
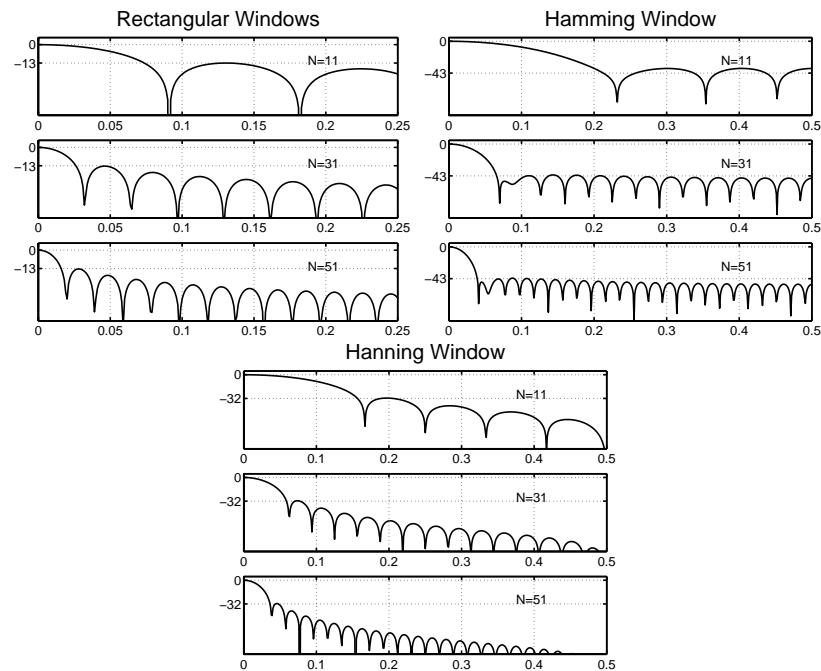
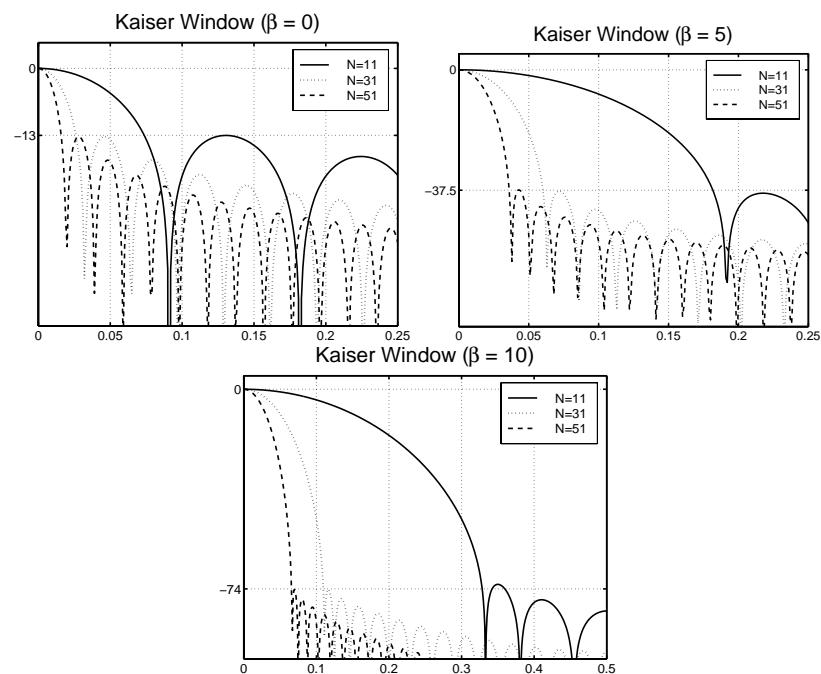


Figure 5.3d: Dolph-Chebyshev Windows

**Figure 5.3a:** Rectangular, Hamming, and Hanning Windows**Figure 5.3b:** Kaiser Windows

5.4 For $x(n) = y(n)w(n)$ where $y(n)$ is

$$y(n) = \cos \omega_1 n + \cos(\omega_2 n + \phi)$$

and $w(n)$ is either a rectangular, Hamming, or Blackman window, the goal is to determine the smallest window length that will allow the two frequencies to be separable in the $|X(e^{j\omega})|^2$ plots.

(a) Figure 5.4a shows the plot of $y(n)$ as well as its true spectrum on the left hand side. The right hand side show six different rectangular windowed spectra. The top row in Figure 5.4a corresponds to a $\phi = 0$ and the bottom row corresponds to $\phi = 0.8\pi$. It is seen that a minimum rectangular window of length $N = 27$ is required to differentiate between the two frequencies for $\phi = 0$ and $N = 55$ is required to separate the frequencies for $\phi = 0.8\pi$.

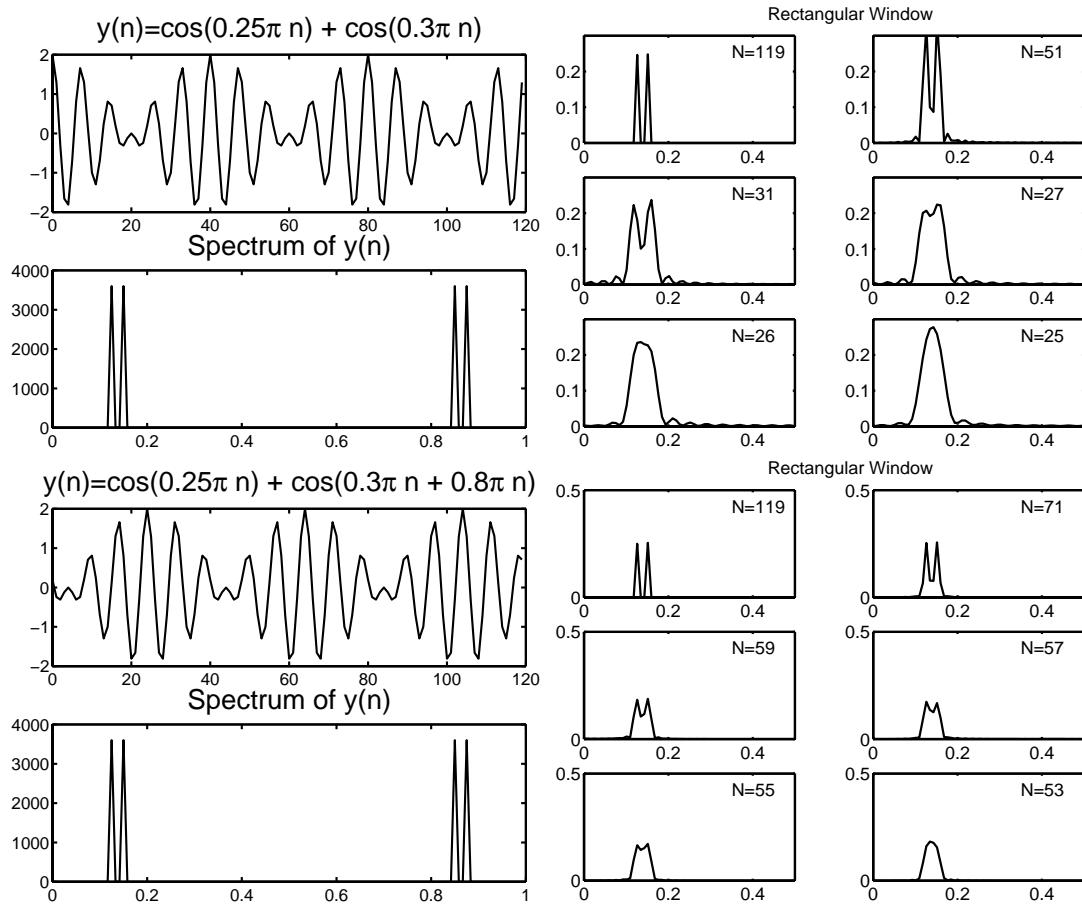


Figure 5.4a: Rectangular Window

(b) Part (a) above is repeated for the $\phi = 0.8\pi$ signal, and Figure 5.4bc shows the window spectra for the Hamming and Blackman windows. For Figure 5.4bc it is seen that the Hamming window requires an approximate minimum length $N = 65$ and the Blackman window length $N = 69$ is required.

5.5 The goal of this problem is to prove that the autocorrelation matrix $\hat{\mathbf{R}}_x$ is a nonnegative definite matrix, that is,

$$\mathbf{z}^H \hat{\mathbf{R}}_x \mathbf{z} \geq 0 \quad \forall z \neq 0$$

(a) Show that $\hat{\mathbf{R}}_x$ can be decomposed into the product $\mathbf{X}^H \mathbf{X}$, where \mathbf{X} is called a data matrix.

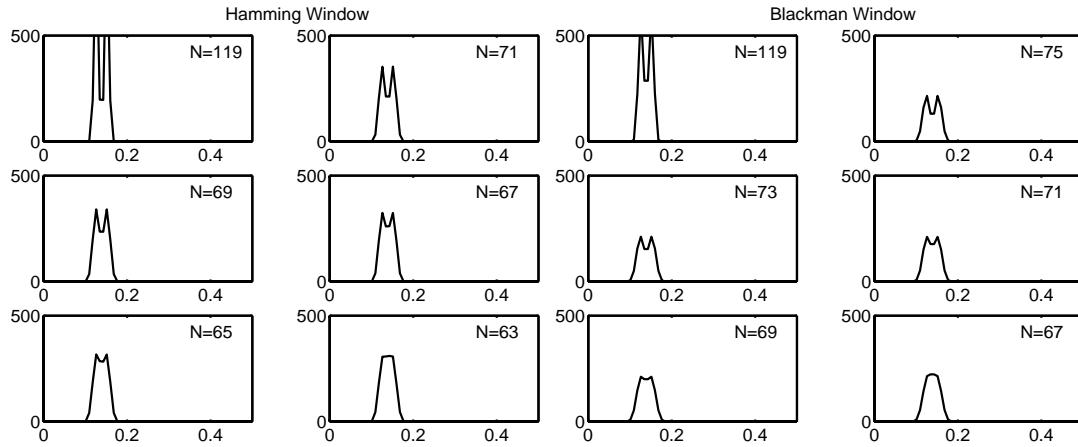


Figure 5.4bc: Hamming and Blackman Windows

Start with the definition of $\hat{\mathbf{R}}_x$ which is

$$\hat{\mathbf{R}}_x = \begin{bmatrix} \hat{r}_x(0) & \hat{r}_x^*(1) & \cdots & \hat{r}_x^*(N-1) \\ \hat{r}_x(1) & \hat{r}_x(0) & \ddots & \hat{r}_x^*(N-2) \\ \vdots & \ddots & \ddots & \vdots \\ \hat{r}_x(N-1) & \hat{r}_x(N-2) & \cdots & \hat{r}_x(0) \end{bmatrix}$$

and

$$\hat{r}_x(l) = \begin{cases} \frac{1}{N} \sum_{n=0}^{N-l-1} x(n+l)x^*(n) & 0 \leq l \leq N-1 \\ \frac{1}{N} \sum_{n=0}^{N-l-1} x(n-l)x^*(n) & -(N-1) \leq l \leq 0 \\ 0 & otherwise \end{cases}$$

Let \mathbf{X} be

$$\mathbf{X} = \frac{1}{\sqrt{N}} \begin{bmatrix} x(0) & 0 & 0 & \cdots \\ x(1) & x(0) & 0 & \cdots \\ \vdots & \vdots & x(0) & \ddots \\ \vdots & \vdots & \vdots & \ddots \\ x(N-1) & x(N-2) & x(N-3) & \ddots \\ 0 & x(N-1) & x(N-2) & \ddots \\ \vdots & 0 & x(N-1) & \ddots \\ \vdots & \vdots & 0 & \ddots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

then $\mathbf{X}^H \mathbf{X}$ is

$$\mathbf{X}^H \mathbf{X}(k, l) = \frac{1}{N} \sum_{n=0}^{N-1} x(n-k)x^*(n-l) = \frac{1}{N} \hat{r}_x(k-l)$$

Clearly, from the above equation $\mathbf{X}^H \mathbf{X} = \hat{\mathbf{R}}_x$

(b) Show that $\mathbf{z}^H \hat{\mathbf{R}}_x \mathbf{z} \geq 0$, for every $\mathbf{z} \neq 0$.

Since $\hat{\mathbf{R}}_x = \mathbf{X}^H \mathbf{X}$ then $\mathbf{z}^H \hat{\mathbf{R}}_x \mathbf{z} = \mathbf{z}^H \mathbf{X}^H \mathbf{X} \mathbf{z} = \|\mathbf{Xz}\|^2$. Where $\|\cdot\|^2$ is the 2-norm.

Clearly $\|\mathbf{Xz}\|^2 \geq 0$ therefore $\mathbf{z}^H \hat{\mathbf{R}}_x \mathbf{z} \geq 0$.

5.6 An alternate autocorrelation estimate $\check{r}_x(l)$ is given as

$$\check{r}_x(l) = \begin{cases} \frac{1}{N-l} \sum_{n=0}^{N-l-1} x(n+l)x^*(n) & 0 \leq l \leq L < N \\ \check{r}_x^*(-l) & -N < -L \leq l < 0 \\ 0 & \text{elsewhere} \end{cases}$$

This is an unbiased estimator, but it is not used in spectral estimation due to its negative definiteness.

(a) Show that the mean of $\check{r}_x(l)$ is equal to $r_x(l)$ and determine an approximate expression for the variance of $\check{r}_x(l)$.

The mean can be shown in a straight forward manner as

$$\begin{aligned} E[\check{r}_x(l)] &= E\left[\frac{1}{N-l} \sum_{n=0}^{N-l-1} x(n+l)x^*(n)\right] = \frac{1}{N-l} \sum_{n=0}^{N-l-1} E[x(n+l)x^*(n)] \\ &= \frac{1}{N-l} \sum_{n=0}^{N-l-1} r_x(l) = r_x(l) \end{aligned}$$

The variance of $\check{r}_x(l)$ is $\text{var}[\check{r}_x(l)] = E\{\check{r}_x(l)\check{r}_x^*(l)\} - |r_x(l)|^2$. Consider

$$\begin{aligned} E[\check{r}_x(l)\check{r}_x^*(l)] &= E\left[\left\{\frac{1}{N-l} \sum_{n=0}^{N-l-1} x(n+l)x^*(n)\right\} \left\{\frac{1}{N-l} \sum_{m=0}^{N-l-1} x^*(m+l)x(m)\right\}\right] \\ &= \frac{1}{(N-l)^2} \sum_{n=0}^{N-l-1} \sum_{m=0}^{N-l-1} E[x(n+l)x^*(n)x^*(m+l)x(m)] \end{aligned}$$

Thus

$$\text{var}[\check{r}_x(l)] = \frac{1}{(N-l)^2} \sum_{n=0}^{N-l-1} \sum_{m=0}^{N-l-1} E[x(n+l)x^*(n)x^*(m+l)x(m)] - |r_x(l)|^2$$

which under mild conditions can be shown to be

$$\text{var}[\check{r}_x(l)] \simeq \frac{1}{N-l} \sum_{k=-\infty}^{\infty} [r_x(k)^2 + r_x(k+l)r_x^*(k-l)]$$

(b) Show that the mean of the corresponding periodogram is given by

$$E\{\check{R}_x(e^{j\omega})\} = \frac{1}{2\pi} \int_{-\infty}^{\infty} R_x(e^{j\varphi}) W_R(e^{j\omega-\varphi}) d\varphi$$

where $W_R(e^{j\omega})$ is the DTFT of the rectangular window and is something called the *Dirichlet kernel*.

Start with

$$\begin{aligned} E\{\check{R}_x(e^{j\omega})\} &= E\left\{\sum_{l=-\infty}^{\infty} \check{r}_x(l)e^{-j\omega l}\right\} = E\left\{\sum_{l=-\infty}^{\infty} \check{r}_x(l)e^{-j\omega l}\right\} = \sum_{l=-N+1}^{N-1} E\{\check{r}_x(l)\}e^{-j\omega l} \\ &= \sum_{l=-N+1}^{N-1} r_x(l)e^{-j\omega l} \end{aligned}$$

Then if we define $w_R(l)$ as

$$w_R(l) = \begin{cases} 1 & -N + 1 \leq l \leq N - 1 \\ 0 & \text{elsewhere} \end{cases}$$

Then the mean is equivalent to $\sum_{l=-\infty}^{\infty} w_R(l)r_x(l)e^{-j\omega l}$ which is the DTFT of the product of two signals. Multiplication in the time domain is convolution in the frequency domain, therefore

$$E\{\check{R}_x(e^{j\omega})\} = \sum_{l=-\infty}^{\infty} w_R(l)r_x(l)e^{-j\omega l} = \frac{1}{2\pi} \int_{-\infty}^{\infty} R_x(e^{j\varphi})W_R(e^{j\omega-\varphi})d\varphi$$

5.7 Let $\check{r}_x(l)$ be the unbiased autocorrelation estimator (5.2.13) of a zero-mean white Gaussian process with variance σ_x^2 . For simplicity, we will assume a real-valued Gaussian process.

(a) The variance of $\check{r}_x(l)$ is $\text{var}[\check{r}_x(l)] = E[\check{r}_x(l)\check{r}_x^*(l)] - r_x^2(l)$ which from Problem 5.6 is

$$\text{var}[\check{r}_x(l)] = \frac{1}{(N-l)^2} \sum_{m=0}^{N-l-1} \sum_{n=0}^{N-l-1} E[x(n+l)x(n)x(m+l)x(m)] - r_x^2(l) \quad (1)$$

Using (3.2.53), we have

$$\begin{aligned} E[x(n+l)x(n)x(m+l)x(m)] &= E[x(n+l)x(n)]E[x(m+l)x(m)] \\ &\quad + E[x(n+l)x(m+l)]E[x(n)x(m)] \\ &\quad + E[x(n+l)x(m)]E[x(n)x(m+l)] \\ &= r_x^2(l) + r_x^2(m-n) + r_x(n-m+l)r_x(n-m-l) \end{aligned} \quad (2)$$

Substituting (2) in (1),

$$\text{var}[\check{r}_x(l)] = \frac{1}{(N-l)^2} \sum_{m=0}^{N-l-1} \sum_{n=0}^{N-l-1} [r_x^2(m-n) + r_x(m-n-l)r_x(m-n+l)]$$

Assuming stationarity, we can reduce the double summation to one, that is,

$$\begin{aligned} \text{var}[\check{r}_x(l)] &= \frac{1}{(N-l)^2} \sum_{m=0}^{N-l-1} \sum_{n=0}^{N-l-1} [r_x^2(m-n) + r_x(m-n-l)r_x(m-n+l)] \\ &= \frac{1}{N-l} \sum_{k=-(N-l)}^{N-l} \left(1 - \frac{|k|}{N-l}\right) [r_x^2(k) + r_x(k-l)r_x(k+l)] \end{aligned}$$

Now for large values of l and N , $\left(1 - \frac{|k|}{N-l}\right) \rightarrow 1$ for small enough values of k . Hence

$$\text{var}[\check{r}_x(l)] \simeq \frac{1}{N-l} \sum_{k=-\infty}^{\infty} [r_x^2(k) + r_x(k-l)r_x(k+l)]$$

which does not become small as $N, l \rightarrow \infty$.

(b) For the biased estimator $\hat{r}_x(l)$, under the same conditions as those in part (a), we have

$$\text{var}[\hat{r}_x(l)] \simeq \frac{1}{N} \sum_{k=-\infty}^{\infty} [r_x^2(k) + r_x(k-l)r_x(k+l)]$$

which becomes zero as $N \rightarrow \infty$.

5.8 To be completed.

5.9 The periodogram $\hat{R}_x(e^{j\omega})$ can also be expressed as a DTFT of the autocorrelation estimate $\hat{r}_x(l)$ given in (5.2.1).

(a) Let $v(n) = x(n)w_R(n)$, where $w_R(n)$ is a rectangular window of length N . Show that

$$\hat{r}_x(l) = \frac{1}{N} v(l) * v^*(-l)$$

This can be shown directly as follows

$$\begin{aligned}\hat{r}_x(l) &= \frac{1}{N} \sum_{n=0}^{N-l-1} x(n+l)x^*(n) = \frac{1}{N} \sum_{n=-\infty}^{\infty} x(n+l)w_R(n+l)x^*(n)w_R^*(n) \\ &= \frac{1}{N} \sum_{n=-\infty}^{\infty} v(n+l)v^*(n)\end{aligned}$$

which is the convolution summation between $v(l)$ and $v^*(-l)$. Therefore, $\hat{r}_x(l) = \frac{1}{N} v(l) * v^*(-l)$.

(b) Show that

$$\hat{R}_x(e^{j\omega}) = \sum_{l=-N+1}^{N-1} \hat{r}_x(l)e^{-j\omega l}$$

The definition of the DTFT is

$$\hat{R}_x(e^{j\omega}) = \sum_{l=-\infty}^{\infty} \hat{r}_x(l)e^{-j\omega l}$$

but $\hat{r}_x(l) = 0$ if $l \leq -N$ or $l \geq N$. Therefore

$$\hat{R}_x(e^{j\omega}) = \sum_{l=-N+1}^{N-1} \hat{r}_x(l)e^{-j\omega l}$$

5.10 Consider the following simple windows over $0 \leq n \leq N-1$: rectangular, Bartlett, Hanning, and Hamming.

(a) Determine the DTFT of the following window analytically.

i. Rectangular: The shape of the rectangular window is

$$w_R(n) = \begin{cases} 1 & n = 0, 1, \dots, N-1 \\ 0 & \text{otherwise} \end{cases}$$

Inserting this directly into the equation for the DTFT gives the following

$$W_R(e^{j\omega}) = \sum_{n=-\infty}^{\infty} w_R(n)e^{-j\omega n} = \sum_{n=0}^{N-1} e^{-j\omega n} = \frac{1 - e^{-j\omega N}}{1 - e^{-j\omega}}$$

Using Euler's relation $\sin(\omega) = \frac{1}{2}(e^{j\omega} - e^{-j\omega})$, the above equation can be simplified to

$$W_R(e^{j\omega}) = e^{-j\frac{\omega}{2}(N-1)} \frac{\sin(\omega N/2)}{\sin(\omega/2)}$$

ii. Bartlett:

iii. Hanning: The Hanning window is defined as

$$w_{\text{Hn}}(n) = \begin{cases} 0.5 - 0.5 \cos \frac{2\pi n}{N-1} & 0 \leq n \leq N-1 \\ 0 & \text{otherwise} \end{cases}$$

Computing the DTFT directly follows

$$\begin{aligned} W_{\text{Hn}}(e^{j\omega}) &= \sum_{n=-\infty}^{\infty} w_{\text{Hn}}(n)e^{-j\omega n} = \sum_{n=0}^{N-1} \left(0.5 - 0.5 \cos \frac{2\pi n}{N-1} \right) e^{-j\omega n} \\ &= 0.5 \sum_{n=0}^{N-1} e^{-j\omega n} - 0.5 \sum_{n=0}^{N-1} \cos \left(\frac{2\pi n}{N-1} \right) e^{-j\omega n} \end{aligned}$$

The first part of the above equation is just 0.5 times the spectrum of the rectangular window. The second part $-0.5 \sum_{n=0}^{N-1} \cos \left(\frac{2\pi n}{N-1} \right) e^{-j\omega n}$ can be computed by first replacing the $\cos(\omega)$ with Eulers relation $\cos \omega = \frac{1}{2}(e^{j\omega} + e^{-j\omega})$ which is

$$-\frac{1}{4} \sum_{n=0}^{N-1} (e^{\frac{j2\pi n}{N-1}} - e^{-\frac{j2\pi n}{N-1}}) e^{-j\omega n}$$

where the above equation can be separated as follows

$$-\frac{1}{4} \sum_{n=0}^{N-1} e^{-j(\omega - \frac{2\pi}{N-1})n} + \frac{1}{4} \sum_{n=0}^{N-1} e^{-j(\omega + \frac{2\pi}{N-1})n}$$

Each side can be computed independently as

$$\pm \frac{1}{4} \frac{1 - \exp -j(\omega \pm \frac{2\pi}{N-1})N}{1 - \exp -j(\omega \pm \frac{2\pi}{N-1})}$$

and using Eulers relation, the above equation is simplified to

$$\pm \frac{1}{4} e^{-j\frac{1}{2}(\omega \pm \frac{2\pi}{N-1}(N-1))} \frac{\sin(\frac{1}{2}(\omega \pm \frac{2\pi}{N-1})N)}{\sin(\frac{1}{2}(\omega \pm \frac{2\pi}{N-1}))}$$

Combining all of the above pieces, the spectrum for the Hanning window $W_{\text{Hn}}(e^{j\omega})$ is

$$\begin{aligned} W_{\text{Hn}}(e^{j\omega}) &= \frac{1}{4} e^{-j\frac{\omega}{2}(N-1)} \frac{\sin(\omega N/2)}{\sin(\omega/2)} - \frac{1}{4} e^{-j\frac{1}{2}(\omega - \frac{2\pi}{N-1})(N-1)} \frac{\sin(\frac{1}{2}(\omega - \frac{2\pi}{N-1})N)}{\sin(\frac{1}{2}(\omega - \frac{2\pi}{N-1}))} \\ &\quad + \frac{1}{4} e^{-j\frac{1}{2}(\omega + \frac{2\pi}{N-1})(N-1)} \frac{\sin(\frac{1}{2}(\omega + \frac{2\pi}{N-1})N)}{\sin(\frac{1}{2}(\omega + \frac{2\pi}{N-1}))} \end{aligned}$$

iv. Hamming: The Hamming window is defined as

$$w_{\text{Hm}}(n) = \begin{cases} 0.54 - 0.46 \cos \frac{2\pi n}{N-1} & 0 \leq n \leq N-1 \\ 0 & \text{otherwise} \end{cases}$$

It is easily seen that the only difference between the Hamming and Hanning windows are the multiplicative constants. The spectrum for the Hamming window is the same as for the Hanning window

except for the change in the constant values. Therefore, the spectrum for the Hamming window $W_{Hm}(e^{j\omega})$ is

$$\begin{aligned} W_{Hm}(e^{j\omega}) &= (0.27)e^{-j\frac{\omega}{2}}(N-1)\frac{\sin(\omega N/2)}{\sin(\omega/2)} - (0.23)e^{-j\frac{1}{2}}(\omega - \frac{2\pi}{N-1})(N-1)\frac{\sin(\frac{1}{2}(\omega - \frac{2\pi}{N-1})N)}{\sin(\frac{1}{2}(\omega - \frac{2\pi}{N-1}))} \\ &\quad + (0.23)e^{-j\frac{1}{2}}(\omega + \frac{2\pi}{N-1})(N-1)\frac{\sin(\frac{1}{2}(\omega + \frac{2\pi}{N-1})N)}{\sin(\frac{1}{2}(\omega + \frac{2\pi}{N-1}))} \end{aligned}$$

- (b) Figure 5.10bc shows the DTFT plots calculated by Matlab for the rectangular, Bartlett, Hanning and Hamming windows of length $N = 31$. On the same plots for each window are the DTFT plots determined analytically above.

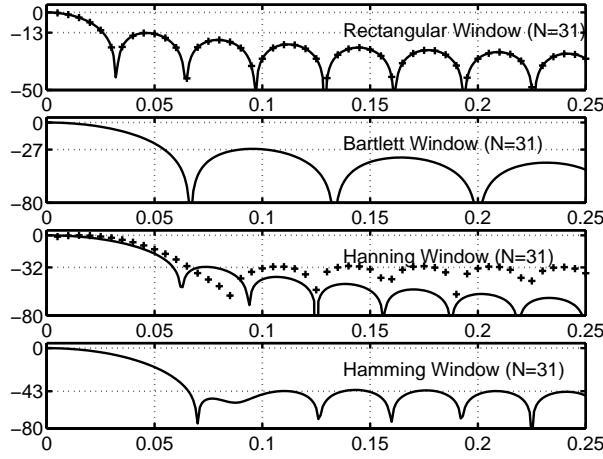


Figure 5.10bc: DTFT of simple windows

5.11 The Parzen window is given by

$$w_P(l) = \begin{cases} 1 - 6\left(\frac{l}{L}\right)^2 + 6\left(\frac{l}{L}\right)^3 & 0 \leq |l| \leq L \\ 2\left(1 - \frac{l}{L}\right)^3 & \frac{L}{2} \leq |l| \leq L \\ 0 & \text{elsewhere} \end{cases}$$

- (a) Show that its DTFT is given by

$$W_P(e^{j\omega}) \simeq \left[\frac{\sin(\omega L/4)}{\sin(\omega/4)} \right]^4 \geq 0$$

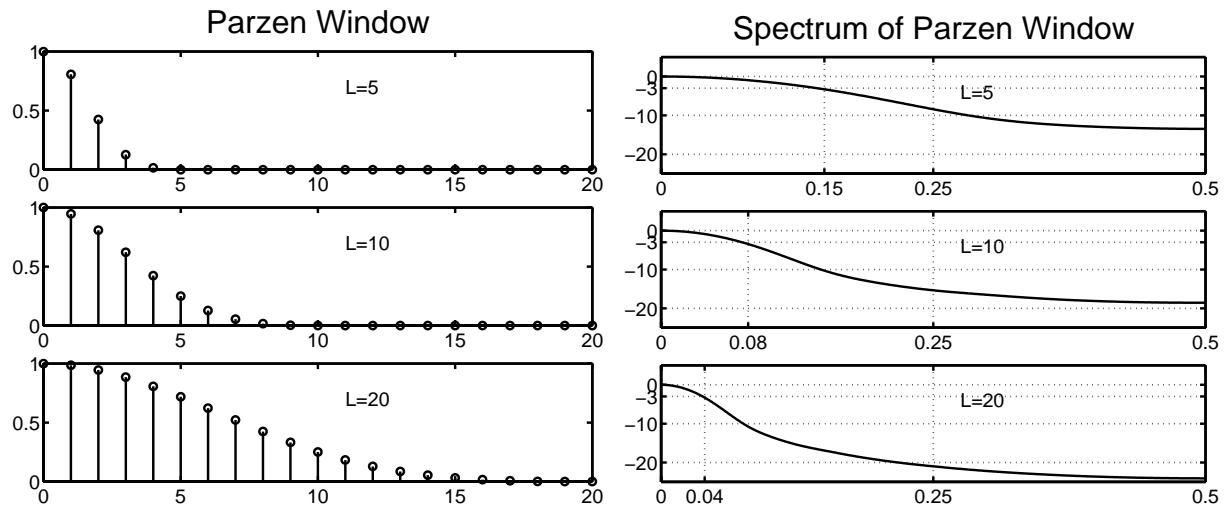
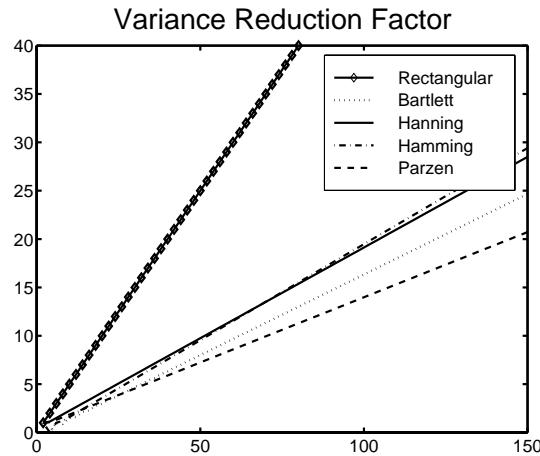
- (b) The left hand side of Figure 5.11bc shows the Parzen window at length $L = 5, 10, 20$. The right hand side of Figure 5.11bc has the magnitude spectrum of the Parzen windows with the -3 dB points noted.

5.12 The variance reduction ratio of a correlation window $\omega_a(l)$ is defined as

$$\frac{\text{var}\{\hat{R}_x^{(PS)}(e^{j\omega})\}}{\text{var}\{\hat{R}_x(e^{j\omega})\}} \simeq \frac{E_\omega}{N} \quad 0 < \omega < \pi$$

where

$$E_\omega = \frac{1}{2\pi} \int_{-\pi}^{\pi} W_a^2(e^{j\omega}) d\omega = \sum_{l=-(L-1)}^{L-1} \omega_a^2(l)$$

**Figure 5.11bc:** Parzen Window**Figure 5.12a:** Variance Reduction Ratio

- (a) Figure 5.12a shows the plot of E_ω as a function of L for the rectangular, Bartlett, Hanning, Hamming, and Parzen windows.
- (b) From Figure 5.12a, the slope of E_ω for each window for $L \gg 1$ can be computed. The slopes for each window approach the values in the following table:

Window Name	Variance Reduction Ratio
Rectangular	$2L/N$
Bartlett	$0.667L/N$
Hanning	$0.75L/N$
Hamming	$0.7948L/N$
Parzen	$0.539L/N$

5.13 Matlab function to compute autocorrelations using FFT.

```
function rx = autocfft(x,L);
% Autocorrelation sequence computation using FFT
% rx = autocfft(x,L)
% Inputs:
```

```

%
%           x : data vector; L : maximum correlation distance
% Output:
%           rx : estimate of autocorrelation over 0 <= l <= L-1
%
%-----
% Copyright 2000, by Dimitris G. Manolakis, Vinay K. Ingle,
% and Stephen M. Kogon. For use with the book
%
% "Statistical and Adaptive Signal Processing"
%
% McGraw-Hill Higher Education.
%-----

```

```

N = length(x); x = x - mean(x);
if L > (N-1); error('*** L must be <= N-1 ***'); end;
x = reshape(x,N,1); x = [x; zeros(N,1)];
Nfft = 2^ceil(log10(2*N-1)/log10(2));
Rx = (1/N)*(abs(fft(x,Nfft)).^2);
rx = ifft(Rx);
%rx = [flipud(rx(end:-1:end-L+1));rx(1:1:L+1)];
rx = rx(1:1:L);
if isreal(x); rx = real(rx); end;

```

For comparison between and autocfft.m execution time consider the following Matlab script.

```

x = rand(1000,1); flops(0); rx = autocfft(x,200); flops
ans =
    160532
x = rand(1000,1); flops(0); rx = autoc(x,200); flops
ans =
    482603
x = rand(1000,1); flops(0); rx = autocfft(x,500); flops
ans =
    160532
x = rand(1000,1); flops(0); rx = autoc(x,500); flops
ans =
    1503503

```

Clearly the autocfft.m function is more efficient for larger values of lag l .

5.14 The Welch-Bartlett estimate $\hat{R}_x^{(PA)}(k)$ is given by

$$\hat{R}_x^{(PA)}(k) = \frac{1}{KL} \sum_{i=0}^{K-1} |X_i(k)|^2$$

Let $x(n)$ be real valued. Then two real-valued sequences can be combined into one complex-valued sequence

$$g_r(n) = x_{2r}(n) + jx_{2r+1}(n) \quad n = 0, 1, \dots, L-1, r = 0, 1, \dots, \frac{K}{2}-1$$

and the L-point DFT of $g_r(n)$ is

$$\tilde{G}_r(k) = \tilde{X}_{2r}(k) + j\tilde{X}_{2r+1}(k) \quad k = 0, 1, \dots, L-1 \quad r = 0, 1, \dots, \frac{K}{2}-1$$

(a) Show that

$$|\tilde{G}_r(k)|^2 + |\tilde{G}_r(L-k)|^2 = 2[|\tilde{X}_{2r}(k)|^2 + |\tilde{X}_{2r+1}(k)|^2] \quad k = r = 0, 1, \dots, \frac{K}{2}-1$$

Start with

$$|\tilde{G}_r(k)|^2 + |\tilde{G}_r(L-k)|^2 = |\tilde{X}_{2r}(k) + j\tilde{X}_{2r+1}(k)|^2 + |\tilde{X}_{2r}(L-k) + j\tilde{X}_{2r+1}(L-k)|^2$$

The right hand side can be expanded into

$$\begin{aligned} r.h.s. &= (\tilde{X}_{2r}(k) + j\tilde{X}_{2r+1}(k))(\tilde{X}_{2r}^*(k) + j\tilde{X}_{2r+1}^*(k)) \\ &\quad + (\tilde{X}_{2r}(L-k) + j\tilde{X}_{2r+1}(L-k))(\tilde{X}_{2r}^*(L-k) + j\tilde{X}_{2r+1}^*(L-k)) \end{aligned}$$

Working out this multiplication directly results in the following cross terms

$$\begin{aligned} r.h.s. &= |\tilde{X}_{2r}(k)|^2 + |\tilde{X}_{2r+1}(k)|^2 - j\tilde{X}_{2r}(k)\tilde{X}_{2r+1}^*(k) + j\tilde{X}_{2r+1}(k)\tilde{X}_{2r}^*(k) \\ &\quad + |\tilde{X}_{2r}(L-k)|^2 + |\tilde{X}_{2r+1}(L-k)|^2 \\ &\quad - j\tilde{X}_{2r}(L-k)\tilde{X}_{2r+1}^*(L-k) + j\tilde{X}_{2r+1}(L-k)\tilde{X}_{2r}^*(L-k) \end{aligned}$$

This can be simplified using the fact that for real-valued x_{2r} , $\tilde{X}_{2r}(L-k) = \tilde{X}_{2r}^*(k)$. Most of the cross terms cancel, leaving

$$= |\tilde{X}_{2r}(k)|^2 + |\tilde{X}_{2r+1}(k)|^2 + |\tilde{X}_{2r}(L-k)|^2 + |\tilde{X}_{2r+1}(L-k)|^2$$

This can be further simplified since for real-valued x_{2r} the following is true

$$|\tilde{X}_{2r}(L-k)|^2 = (\tilde{X}_{2r}(L-k))(\tilde{X}_{2r}^*(L-k)) = (\tilde{X}_{2r}^*(k))(\tilde{X}_{2r}(k)) = |\tilde{X}_{2r}(k)|^2$$

Similarly $|\tilde{X}_{2r+1}(L-k)|^2 = |\tilde{X}_{2r+1}(k)|^2$. The final result is shown as

$$|\tilde{G}_r(k)|^2 + |\tilde{G}_r(L-k)|^2 = 2[|\tilde{X}_{2r}(k)|^2 + |\tilde{X}_{2r+1}(k)|^2] \quad k = r = 0, 1, \dots, \frac{K}{2}-1$$

(b) The expression for $\hat{R}_x^{(PA)}(k)$ in terms of $\tilde{G}(k)$ is shown below

$$\begin{aligned} \hat{R}_x^{(PA)}(k) &= \frac{1}{KL} \sum_{i=0}^{K-1} |X_i(k)|^2 = \frac{1}{KL} \sum_{i=0}^{\frac{K}{2}-1} (|\tilde{X}_{2i}(k)|^2 + |\tilde{X}_{2i+1}(k)|^2) \\ &= \frac{1}{KL} \sum_{i=0}^{\frac{K}{2}-1} (|\tilde{G}_r(k)|^2 + |\tilde{G}_r(L-k)|^2) \end{aligned}$$

(c) If k is odd, then there is one DFT which is unpaired.

$$\hat{R}_x^{(PA)}(k) = \frac{1}{KL} \sum_{i=0}^{\frac{(K-1)}{2}-1} (|\tilde{X}_{2i}(k)|^2 + |\tilde{X}_{2i+1}(k)|^2) + \frac{1}{KL} |\tilde{X}_{K-1}(k)|^2$$

5.15 The autocorrelation estimate from a Welch's method PSD estimate is

$$\hat{r}_x^{(\text{PA})}(l) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \hat{R}_x^{(\text{PA})}(e^{j\omega}) e^{j\omega l} d\omega$$

Let $\tilde{\hat{R}}_x^{(\text{PA})}(k)$ be samples of $\hat{R}_x^{(\text{PA})}(k)$ according to

$$\tilde{\hat{R}}_x^{(\text{PA})}(k) = \hat{R}_x^{(\text{PA})}(e^{j2\pi k/N_{\text{FFT}}}) \quad 0 \leq k \leq N_{\text{FFT}} - 1$$

(a) Show that the IDFT $\tilde{\hat{r}}_x^{(\text{PA})}(l)$ of $\tilde{\hat{R}}_x^{(\text{PA})}(k)$ is an aliased version of the autocorrelation estimate $\hat{r}_x^{(\text{PA})}(l)$

The IDFT of $\tilde{\hat{R}}_x^{(\text{PA})}(k)$ is

$$\begin{aligned} \tilde{\hat{r}}_x^{(\text{PA})}(l) &= \frac{1}{N_{\text{FFT}}} \sum_{k=0}^{N_{\text{FFT}}-1} \tilde{\hat{R}}_x^{(\text{PA})}(k) e^{j2\pi kl/N_{\text{FFT}}} = \frac{1}{N_{\text{FFT}}} \sum_{k=0}^{N_{\text{FFT}}-1} \hat{R}_x^{(\text{PA})}(e^{j2\pi k/N_{\text{FFT}}}) e^{j2\pi kl/N_{\text{FFT}}} \\ &= \frac{1}{N_{\text{FFT}}} \int_0^{2\pi} \left[\hat{R}_x^{(\text{PA})}(e^{j\omega}) e^{j\omega l} \right] \left[\sum_{k=0}^{N_{\text{FFT}}-1} \delta\left(\omega - \frac{2\pi k}{N_{\text{FFT}}}\right) \right] d\omega \end{aligned}$$

This is the convolution between $\hat{r}_x^{(\text{PA})}(l)$ and an impulse train. Using the identity that convolution with an impulse train results in aliased version of the signal, therefore $\tilde{\hat{r}}_x^{(\text{PA})}(l)$ is

$$\tilde{\hat{r}}_x^{(\text{PA})}(l) = \sum_{k=0}^{N_{\text{FFT}}-1} \hat{r}_x^{(\text{PA})}(l - kN_{\text{FFT}})$$

(b) Since $\hat{R}_x^{(\text{PA})}(e^{j\omega})$ is obtained by averaging periodograms, each of which, according to (5.3.6), is a DTFT of length $(2L - 1)$ autocorrelation estimates $\hat{r}_{x,i}^{(\text{PA})}(l)$. Clearly then $\hat{R}_x^{(\text{PA})}(e^{j\omega})$ is also a DTFT of length $(2L - 1)$ autocorrelation estimates $\hat{r}_x^{(\text{PA})}(l)$. Hence to avoid aliasing, $N_{\text{FFT}} \geq (2L - 1)$.

5.16 The coherence function \mathcal{G}_{xy}^2 is

$$\mathcal{G}_{xy}^2 = \frac{|R_{xy}(e^{j\omega})|^2}{R_x(e^{j\omega}) R_y(e^{j\omega})}$$

Show that it is invariant under linear transformation $x_1 = h_1(n) * x(n)$ and $y_1 = h_2(n) * y(n)$.

Start with the definition of the coherence function

$$\mathcal{G}_{x_1 y_1}^2 = \frac{|R_{x_1 y_1}(e^{j\omega})|^2}{R_{x_1}(e^{j\omega}) R_{y_1}(e^{j\omega})}$$

Substitute into the above equation the following appropriate filtered autocorrelation relations

$$\begin{aligned} R_{x_1}(e^{j\omega}) &= H_1(e^{j\omega}) H_1^*(e^{j\omega}) R_x(e^{j\omega}), \quad R_{y_1}(e^{j\omega}) = H_2(e^{j\omega}) H_2^*(e^{j\omega}) R_y(e^{j\omega}) \\ |R_{x_1 y_1}(e^{j\omega})|^2 &= R_{x_1 y_1}(e^{j\omega}) R_{x_1 y_1}^*(e^{j\omega}) = [H_1^*(e^{j\omega}) H_2(e^{j\omega}) R_{xy}(e^{j\omega})] [H_1(e^{j\omega}) H_2^*(e^{j\omega}) R_{xy}^*(e^{j\omega})] \end{aligned}$$

Therefore

$$\begin{aligned} \mathcal{G}_{x_1 y_1}^2 &= \frac{|R_{x_1 y_1}(e^{j\omega})|^2}{R_{x_1}(e^{j\omega}) R_{y_1}(e^{j\omega})} = \frac{[H_1^*(e^{j\omega}) H_2(e^{j\omega}) R_{xy}(e^{j\omega})] [H_1(e^{j\omega}) H_2^*(e^{j\omega}) R_{xy}^*(e^{j\omega})]}{H_1(e^{j\omega}) H_1^*(e^{j\omega}) R_x(e^{j\omega}) H_2(e^{j\omega}) H_2^*(e^{j\omega}) R_y(e^{j\omega})} \\ &= \frac{|R_{xy}(e^{j\omega})|^2}{R_x(e^{j\omega}) R_y(e^{j\omega})} \end{aligned}$$

5.17 Bartlett's Method as a special case of Welch's method:

- (a) Consider the i th periodogram:
- (b) Let $\mathbf{u}(e^{j\omega}) = [1 \ e^{j\omega} \ \dots \ e^{j(L-1)\omega}]^T$ and

$$\hat{\mathbf{R}}_{x,i} = \begin{bmatrix} \hat{r}_{x,i}(0) & \hat{r}_{x,i}(-1) & \dots & \hat{r}_{x,i}(1-L) \\ \hat{r}_{x,i}(1) & \hat{r}_{x,i}(0) & \dots & \hat{r}_{x,i}(2-L) \\ \vdots & \vdots & \ddots & \vdots \\ \hat{r}_{x,i}(L-1) & \hat{r}_{x,i}(L-2) & \dots & \hat{r}_{x,i}(0) \end{bmatrix}$$

We will first show (P.6) using explicit calculations. First note that

$$w_B(l) = \left(\frac{L-|l|}{L} \right), \quad |l| \leq L \quad (1)$$

Consider $L = 1$: Then

$$\begin{aligned} \frac{1}{1} \mathbf{u}^H(e^{j\omega}) \hat{\mathbf{R}}_{x,i} \mathbf{u}(e^{j\omega}) &= [1] [\hat{r}_{x,i}(0)] [1] = (0) \hat{r}_{x,i}(-1)e^{j\omega} + (1) \hat{r}_{x,i}(0) + (0) \hat{r}_{x,i}(1)e^{-j\omega} \\ &= \sum_{l=-1}^1 \hat{r}_{x,i}(l) (1-|l|) e^{-j\omega l} = \sum_{l=-1}^1 \hat{r}_{x,i}(l) \left(\frac{1-|l|}{1} \right) e^{-j\omega l} \\ &= \sum_{l=-1}^1 \hat{r}_{x,i}(l) w_B(l) e^{-j\omega l} \end{aligned}$$

Consider $L = 2$:

$$\begin{aligned} \frac{1}{2} \mathbf{u}^H(e^{j\omega}) \hat{\mathbf{R}}_{x,i} \mathbf{u}(e^{j\omega}) &= \frac{1}{2} [1 \ e^{-j\omega}] \begin{bmatrix} \hat{r}_{x,i}(0) & \hat{r}_{x,i}(-1) \\ \hat{r}_{x,i}(1) & \hat{r}_{x,i}(0) \end{bmatrix} \begin{bmatrix} 1 \\ e^{j\omega} \end{bmatrix} \\ &= \frac{1}{2} \{ \hat{r}_{x,i}(0) + e^{-j\omega} \hat{r}_{x,i}(1) + [\hat{r}_{x,i}(-1) + e^{-j\omega} \hat{r}_{x,i}(0)] e^{j\omega} \} \\ &= \frac{1}{2} \{ (0) \hat{r}_{x,i}(-2)e^{2j\omega} + (1) \hat{r}_{x,i}(-1)e^{j\omega} + (2) \hat{r}_{x,i}(0) \\ &\quad + (1) \hat{r}_{x,i}(1)e^{-j\omega} + (0) \hat{r}_{x,i}(2)e^{-3j\omega} \} \\ &= \sum_{l=-2}^2 \hat{r}_{x,i}(l) \left(\frac{2-|l|}{2} \right) e^{-j\omega l} = \sum_{l=-2}^2 \hat{r}_{x,i}(l) w_B(l) e^{-j\omega l} \end{aligned}$$

Consider $L = 3$:

$$\begin{aligned} \frac{1}{3} \mathbf{u}^H(e^{j\omega}) \hat{\mathbf{R}}_{x,i} \mathbf{u}(e^{j\omega}) &= \frac{1}{3} [1 \ e^{-j\omega} \ e^{-j2\omega}] \begin{bmatrix} \hat{r}_{x,i}(0) & \hat{r}_{x,i}(-1) & \hat{r}_{x,i}(-2) \\ \hat{r}_{x,i}(1) & \hat{r}_{x,i}(0) & \hat{r}_{x,i}(-1) \\ \hat{r}_{x,i}(2) & \hat{r}_{x,i}(1) & \hat{r}_{x,i}(0) \end{bmatrix} \begin{bmatrix} 1 \\ e^{j\omega} \\ e^{j2\omega} \end{bmatrix} \\ &= \frac{1}{3} \{ [\hat{r}_{x,i}(0) + e^{-j\omega} \hat{r}_{x,i}(1) + e^{-2j\omega} \hat{r}_{x,i}(2)] + [\hat{r}_{x,i}(-1) + \hat{r}_{x,i}(0)e^{-j\omega} + \hat{r}_{x,i}(1)e^{-j2\omega}] e^{j\omega} \\ &\quad + [\hat{r}_{x,i}(-2) + e^{-j\omega} \hat{r}_{x,i}(-1) + e^{-2j\omega} \hat{r}_{x,i}(0)] e^{2j\omega} \} \\ &= \frac{1}{3} \{ (0) \hat{r}_{x,i}(-3)e^{3j\omega} + \hat{r}_{x,i}(-2)e^{2j\omega} + 2\hat{r}_{x,i}(-1)e^{j\omega} + 3\hat{r}_{x,i}(0) + 2\hat{r}_{x,i}(1)e^{-j\omega} \\ &\quad + \hat{r}_{x,i}(2)e^{-2j\omega} + (0) \hat{r}_{x,i}(2)e^{-3j\omega} \} \\ &= \sum_{l=-3}^3 \hat{r}_{x,i}(l) \left(\frac{3-|l|}{3} \right) e^{-j\omega l} = \sum_{l=-3}^3 \hat{r}_{x,i}(l) w_B(l) e^{-j\omega l} \end{aligned}$$

Now consider the general expression

$$\begin{aligned}\hat{R}_{x,i}(e^{j\omega}) &= \sum_{l=-L}^L \hat{r}_{x,i}(l) w_B(l) e^{-j\omega l} = \sum_{l=-L}^L \hat{r}_{x,i}(l) \left(\frac{L-|l|}{L} \right) e^{-j\omega l} \\ &= \frac{1}{L} \sum_{l=-L}^L \hat{r}_{x,i}(l) (L-|l|) e^{-j\omega l}\end{aligned}$$

Using the identity

$$\sum_{l=-L}^L (L-|l|) f(l) = \sum_{m=0}^{L-1} \sum_{n=0}^{L-1} f(m-n)$$

we have

$$\begin{aligned}\hat{R}_{x,i}(e^{j\omega}) &= \frac{1}{L} \sum_{l=-L}^L \hat{r}_{x,i}(l) (L-|l|) e^{-j\omega l} = \frac{1}{L} \sum_{m=0}^{L-1} \sum_{n=0}^{L-1} \hat{r}_{x,i}(m-n) e^{-j\omega(m-n)} \\ &= \frac{1}{L} \mathbf{u}^H(e^{j\omega}) \hat{\mathbf{R}}_{x,i} \mathbf{u}(e^{j\omega})\end{aligned}$$

(c) Finally, using (5.3.55)

$$\hat{R}_x^{(B)}(e^{j\omega}) = \frac{1}{K} \sum_{i=1}^K \hat{R}_{x,i}(e^{j\omega}) = \frac{1}{KL} \sum_{i=1}^K \mathbf{u}^H(e^{j\omega}) \hat{\mathbf{R}}_{x,i} \mathbf{u}(e^{j\omega})$$

5.18 Combined data and correlation-lag windowing (Carter-Nuttall Approach):

Let $\hat{R}_x^{(PA)}(e^{j\omega})$ be the Welch-Bartlett PSD estimator of $R_x(e^{j\omega})$. Then the Carter-Nuttall PSD estimator $\hat{R}_x^{(CN)}(e^{j\omega})$ is given by

$$\begin{aligned}\hat{R}_x^{(CN)}(e^{j\omega}) &= \mathcal{F}[\hat{r}_x^{(CN)}(l)] \triangleq \mathcal{F}[\hat{r}_x^{(PA)}(l) w_a(l)] \triangleq \hat{R}_x^{(PA)}(e^{j\omega}) \circledast W_a(e^{j\omega}) \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} \hat{R}_x^{(PA)}(e^{j\theta}) W_a(e^{j(\omega-\theta)}) d\theta\end{aligned}\quad (1)$$

where \circledast denotes the periodic convolution and $W_a(e^{j\omega})$ is the spectrum of the correlation lag window $w_a(l)$, $-L \leq l \leq L$.

(a) Bias of $\hat{R}_x^{(CN)}(e^{j\omega})$: Note that from (5.3.57) the mean of $\hat{R}_x^{(PA)}(e^{j\omega})$ is given by

$$E[\hat{R}_x^{(PA)}(e^{j\omega})] = \frac{1}{L} \hat{R}_x^{(PA)}(e^{j\omega}) \circledast R_w(e^{j\omega})$$

where $R_w(e^{j\omega})$ is the spectrum of the data window $w(n)$, $0 \leq n \leq L-1$. Thus from (1)

$$\begin{aligned}E[\hat{R}_x^{(CN)}(e^{j\omega})] &= \frac{1}{2\pi} \int_{-\pi}^{\pi} E[\hat{R}_x^{(PA)}(e^{j\theta})] W_a(e^{j(\omega-\theta)}) d\theta \\ &= E[\hat{R}_x^{(PA)}(e^{j\omega})] \circledast W_a(e^{j\omega})\end{aligned}\quad (2)$$

$$= \frac{1}{L} \hat{R}_x^{(PA)}(e^{j\omega}) \circledast R_w(e^{j\omega}) \circledast W_a(e^{j\omega})\quad (3)$$

Thus $\hat{R}_x^{(CN)}(e^{j\omega})$ is a biased estimator.

- (b) From (1), the Carter-Nuttall PSD estimator, $\hat{R}_x^{(\text{CN})}(e^{j\omega})$, is obtained by performing additional windowing on the Welch-Bartlett PSD estimator, $\hat{R}_x^{(\text{PA})}(e^{j\omega})$. Thus the variance of $\hat{R}_x^{(\text{PA})}(e^{j\omega})$ is further reduced while broadening frequency peaks (or decreasing resolution). The larger the maximum lag-length L , the lesser are these effects.

- (c) Matlab function CN_psd:

```

function [Rx,Freq] = CN_psd(x,L,Lagwindow,Fs)
% Spectrum estimation using Carter and Nuttall approach that
% combines data and lag windows.
% Rx = CN_psd(x,L,Window,Fs)
%
% Inputs:
%       x : data values
%       L : maximum Lag distance
%   LagWindow : Lag window function (must be previously defined)
%       Fs : sampling rate
%
% Output:
%       Rx : Spectral estimates over 0 -- Fs/2
%       Freq : Frequency samples between 0 and Fs/2
%-----
% Copyright 2000, by Dimitris G. Manolakis, Vinay K. Ingle,
% and Stephen M. Kogon. For use with the book
%
% "Statistical and Adaptive Signal Processing"
%
% McGraw-Hill Higher Education.
%-----

Nfft = 2^ceil(log10(2*L-1)/log10(2));      % Nfft : power of 2 & > (2L-1)
datawin = hamming(L);                      % Hamming window for Welch PSD
Rx = psd(x,Nfft,Fs,datawin,L/2,'none');    % Welch PSD with 50% overlap
Rx = [Rx;flipud(Rx(2:end-1))];            % Assemble full period for IFFT
rx = real(ifft(Rx));                      % IFFT -> correlation estimates
rx = [rx(end:-1:end-L+1);rx(1:L+1)];      % extract over -L to L
win = feval(Lagwindow,2*L+1);              % Lag window of length 2*L+1
rx = rx.*win;                            % Window correlation estimates
rx = fftshift(rx(1:end-1));                % assemble for FFT for
rx = [rx(1:L+1);zeros(Nfft-2*L,1);rx(L+2:end)]; % Nfft length DFT
Rx = real(fft(rx));                      % Carter Nuttall estimates
Rx = Rx(1:Nfft/2+1);                     % extract over 0 to Fs/2
Freq = [0:Nfft/2]*Fs/Nfft;                % Frequencies between 0 and Fs/2

```

5.19 The scaling factor $\sum_{n=0}^{L-1} w^2(n) = L$ normalizes the data window. Hence using the Parseval's theorem

$$\frac{1}{L} \sum_{n=0}^{L-1} w^2(n) = \frac{1}{2\pi L} \int_{-\pi}^{\pi} |W(e^{j\omega})|^2 d\omega = 1$$

This makes the estimate $\hat{R}_x^{(\text{PA})}(e^{j\omega})$ asymptotically unbiased.

5.20 Consider the basic periodogram estimator

$$\hat{R}(e^{j\omega}) = \frac{1}{N} \left| \sum_{n=0}^{N-1} x(n) e^{-j\omega n} \right|^2$$

(a) At $\omega = 0$, we have

$$\hat{R}(e^{j0}) = \frac{1}{N} \left| \sum_{n=0}^{N-1} x(n) e^{-jn0} \right|^2 = \frac{1}{N} \left| \sum_{n=0}^{N-1} x(n) \right|^2$$

(b) Assume that $x(n)$ is a real-valued zero-mean white Gaussian process with variance σ_x^2 . Then $r_x(l) = \sigma_x^2$. The mean of $\hat{R}(e^{j0})$ is given by

$$\begin{aligned} E[\hat{R}(e^{j0})] &= E\left[\frac{1}{N} \left| \sum_{n=0}^{N-1} x(n) \right|^2\right] = \frac{1}{N} E\left[\left\{\sum_{n=0}^{N-1} x(n)\right\} \left\{\sum_{m=0}^{N-1} x(m)\right\}\right] \\ &= \frac{1}{N} E\left[\sum_{n=0}^{N-1} \sum_{m=0}^{N-1} x(n)x(m)\right] = \frac{1}{N} \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} E[x(n)x(m)] = \frac{1}{N} \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} r_x(n-m) \\ &= \frac{1}{N} \sigma_x^2 \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} \delta(n-m) = \sigma_x^2 \end{aligned}$$

The variance of $\hat{R}(e^{j0})$ is given by

$$\text{var}[\hat{R}(e^{j0})] = E[\hat{R}^2(e^{j0})] - E^2[\hat{R}(e^{j0})] = E[\hat{R}^2(e^{j0})] - \sigma_x^4$$

Consider

$$\begin{aligned} E[\hat{R}^2(e^{j0})] &= \frac{1}{N^2} E\left[\left\{\sum_{m=0}^{N-1} x(m)\right\} \left\{\sum_{n=0}^{N-1} x(n)\right\} \left\{\sum_{p=0}^{N-1} x(p)\right\} \left\{\sum_{q=0}^{N-1} x(q)\right\}\right] \\ &= \frac{1}{N^2} E\left[\sum_{m=0}^{N-1} \sum_{n=0}^{N-1} \sum_{p=0}^{N-1} \sum_{q=0}^{N-1} x(q)x(p)x(n)x(m)\right] \\ &= \frac{1}{N^2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} \sum_{p=0}^{N-1} \sum_{q=0}^{N-1} E[x(q)x(p)x(n)x(m)] \end{aligned}$$

Now using the property (3.2.53) of real-valued Gaussian random variables, that is,

$$E[x_1x_2x_3x_4] = E[x_1x_2]E[x_3x_4] + E[x_1x_3]E[x_2x_4] + E[x_1x_4]E[x_2x_3]$$

we obtain

$$\begin{aligned} E[x(q)x(p)x(n)x(m)] &= E[x(q)x(p)]E[x(n)x(m)] + E[x(q)x(n)]E[x(p)x(m)] \\ &\quad + E[x(q)x(m)]E[x(p)x(n)] \\ &= \sigma_x^2 \delta(q-p) \sigma_x^2 \delta(n-m) + \sigma_x^2 \delta(q-n) \sigma_x^2 \delta(p-m) + \sigma_x^2 \delta(q-m) \sigma_x^2 \delta(p-n) \\ &= \sigma_x^4 [\delta(q-p)\delta(n-m) + \delta(q-n)\delta(p-m) + \delta(q-m)\delta(p-n)] \end{aligned}$$

Hence

$$\begin{aligned} \mathbb{E} [\hat{R}^2(e^{j0})] &= \frac{\sigma_x^4}{N^2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} \sum_{p=0}^{N-1} \sum_{q=0}^{N-1} [\delta(q-p)\delta(n-m) + \delta(q-n)\delta(p-m) + \delta(q-m)\delta(p-n)] \\ &= 3 \frac{\sigma_x^4}{N^2} \left[\sum_{m=0}^{N-1} \sum_{n=0}^{N-1} \delta(n-m) \right] \left[\sum_{p=0}^{N-1} \sum_{q=0}^{N-1} \delta(q-p) \right] = 3\sigma_x^4 \end{aligned}$$

Therefore,

$$\text{var} [\hat{R}(e^{j0})] = 3\sigma_x^4 - \sigma_x^4 = 2\sigma_x^4$$

(c) Clearly the estimator $\hat{R}^2(e^{j0})$ is not consistent since $\text{var} [\hat{R}(e^{j0})]$ is independent of N .

5.21 Consider Bartlett's method for estimating $\hat{R}_x(e^{j0})$ using $L = 1$. The periodogram of one sample $x(n)$ is simply $|x(n)|^2$. Thus

$$\hat{R}_x^{(B)}(e^{j0}) = \frac{1}{N} \sum_{n=0}^{N-1} \hat{R}_{x,n}(e^{j0}) = \frac{1}{N} \sum_{n=0}^{N-1} |x(n)|^2$$

Assume that $x(n)$ is a real-valued white Gaussian process with variance σ_x^2 . Then $r_x(l) = \mathbb{E}[x^2(n)] = \sigma_x^2$.

(a) The mean of $\hat{R}_x^{(B)}(e^{j0})$ is given by

$$\mathbb{E} [\hat{R}_x^{(B)}(e^{j0})] = \mathbb{E} \left[\frac{1}{N} \sum_{n=0}^{N-1} |x(n)|^2 \right] = \frac{1}{N} \sum_{n=0}^{N-1} \mathbb{E}[x^2(n)] = \sigma_x^2$$

The variance of $\hat{R}_x^{(B)}(e^{j0})$ is given by

$$\text{var} [\hat{R}_x^{(B)}(e^{j0})] = \mathbb{E} \left[\left\{ \hat{R}_x^{(B)}(e^{j0}) \right\}^2 \right] - \mathbb{E}^2 [\hat{R}_x^{(B)}(e^{j0})] = \mathbb{E} \left[\left\{ \hat{R}_x^{(B)}(e^{j0}) \right\}^2 \right] - \sigma_x^4$$

Consider

$$\begin{aligned} \mathbb{E} \left[\left\{ \hat{R}_x^{(B)}(e^{j0}) \right\}^2 \right] &= \mathbb{E} \left[\left\{ \frac{1}{N} \sum_{m=0}^{N-1} x^2(m) \right\} \left\{ \frac{1}{N} \sum_{n=0}^{N-1} x^2(n) \right\} \right] \\ &= \frac{1}{N^2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} \mathbb{E}[x^2(n)x^2(m)] \end{aligned}$$

Using the property (3.2.53) of real-valued Gaussian random variables, that is,

$$\mathbb{E}[x_1x_2x_3x_4] = \mathbb{E}[x_1x_2]\mathbb{E}[x_3x_4] + \mathbb{E}[x_1x_3]\mathbb{E}[x_2x_4] + \mathbb{E}[x_1x_4]\mathbb{E}[x_2x_3]$$

we obtain

$$\begin{aligned} \mathbb{E}[x^2(n)x^2(m)] &= \mathbb{E}[x^2(n)]\mathbb{E}[x^2(m)] + \mathbb{E}[x(n)x(m)]\mathbb{E}[x(n)x(m)] + \mathbb{E}[x(n)x(m)]\mathbb{E}[x(n)x(m)] \\ &= \sigma_x^4 + 2\sigma_x^4\delta(n-m) \end{aligned}$$

Hence

$$\begin{aligned} E\left[\left\{\hat{R}_x^{(B)}(e^{j0})\right\}^2\right] &= \frac{1}{N^2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} E[x^2(n)x^2(m)] = \frac{1}{N^2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} [\sigma_x^4 + 2\sigma_x^4\delta(n-m)] \\ &= \sigma_x^4 + \frac{1}{N^2} 2\sigma_x^4 N = \sigma_x^4 + \frac{2\sigma_x^4}{N} \end{aligned}$$

Finally

$$\text{var}\left[\hat{R}_x^{(B)}(e^{j0})\right] = E\left[\left\{\hat{R}_x^{(B)}(e^{j0})\right\}^2\right] - \sigma_x^4 = \frac{2\sigma_x^4}{N}$$

- (b) Comparison with the results of Problem 5.20: The means and variances of $\hat{R}_x^{(B)}(e^{j0})$ are

Problem	Mean	Variance
5.20	σ_x^2	$2\sigma_x^4$
5.21	σ_x^2	$2\sigma_x^4/N$

Clearly, the variance estimator of Problem 5.21 is consistent.

5.22 Design of a lag window using an arbitrary data window.

- (a) Let $v(n)$, $0 \leq n \leq L - 1$ be a data window. Let the lag window $w(n)$ be defined by

$$w(n) \triangleq \mathcal{F}^{-1}[|\mathcal{F}\{v(n)\}|^2] = \mathcal{F}^{-1}[V(e^{j\omega})V^*(e^{j\omega})]$$

Then using the convolution property of the DTFT, the lag window is

$$w(n) = v(n) * v^*(n), \quad 0 \leq n \leq 2L - 2$$

Thus $w(n)$ has the desired property that its DTFT is nonnegative. We also the length of the window equal to $2L + 1$.

- (b) Matlab script to design a lag window with The Hanning window as the prototype is given below and the plots of the window and its DTFT are shown in Figure P5.22.

```
% (b) Design of a lag window of length 31 using the
%      Hanning window prototype
N = 31; L = (N+1)/2; n = 0:N-1;
v = hanning(L);
w = conv(v,v);
% DTFT of w(n);
W = fft(w,1024); magW = abs(W(1:513));
omg = [0:512]/512;

% Plots
subplot('position',[0.1,0.65,0.85,0.3])
stem(n,w,'g'); axis([-1,N,0,8]);
set(gca,'xtick',[0,15,30],'ytick',[0:2:8]);
xlabel('sample index n','fontsize',label_fontsize);
ylabel('x(n)','fontsize',label_fontsize);
title('Lag Window','fontsize',title_fontsize);
```

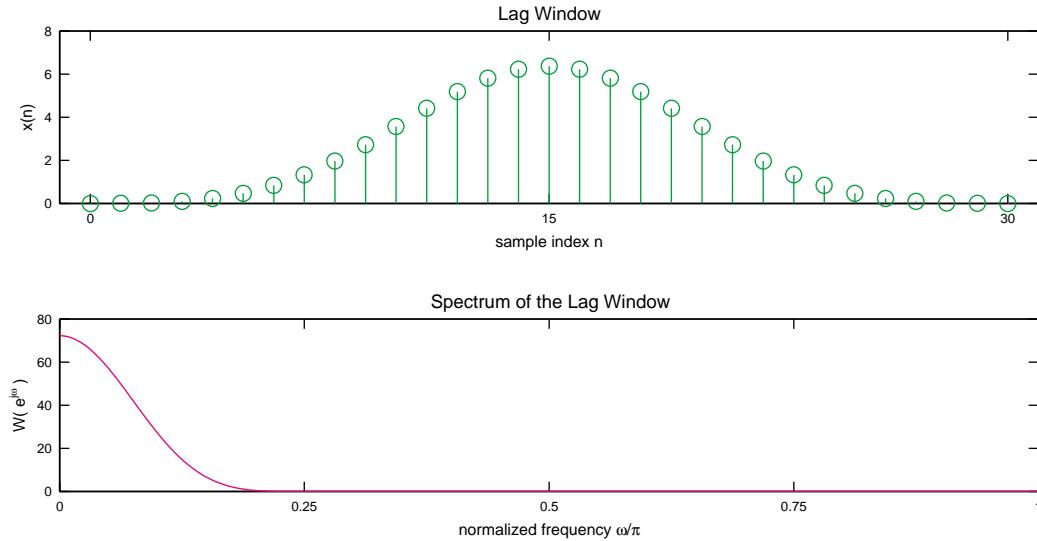


Figure 5.22: Plots of the lag window in the time and frequency domains.

```

subplot('position',[0.1,0.15,0.85,0.3])
plot(omg,magW,'m'); axis([0,1,0,80]);
set(gca,'xtick',[0:0.25:1], 'ytick',[0:20:80]);
xlabel('normalized frequency \omega/\pi','fontsize',label_fontsize);
ylabel('W( e^{j\omega} )','fontsize',label_fontsize);
title('Spectrum of the Lag Window','fontsize',title_fontsize);

```

5.23 Consider the following random process

$$x(n) = \sum_{k=1}^4 A_k \sin(\omega_k n + \phi_k) + v(n)$$

where $A_1 = 1$ $A_2 = 0.5$ $A_3 = 0.5$ $A_4 = 0.25$ $\omega_1 = 0.1\pi$ $\omega_2 = 0.6\pi$ $\omega_3 = 0.65\pi$ $\omega_4 = 0.8\pi$ and the phases $\{\phi_i\}_{i=1}^4$ are IID random variables uniformly distributed over $[-\pi, \pi]$. The Blackman-Tukey estimates will be computed from fifty realizations of $x(n)$ for $0 \leq n \leq 256$

- (a) The top two row of Figure 5.23ab shows the Blackman-Tukey estimate using a Bartlett window for $L = 32, 64, 128$ in dB. The left sided plots are the Blackman-Tukey estimates for each realization, while the right sided plots are the average of the Blackman-Tukey estimates.
- (b) The bottom row of Figure 5.23ab shows the Blackman-Tukey estimate using a Parzen window for $L = 32, 64, 128$ in dB. The left sided plots are the B-T estimates for each realization, while the right sided plots are the average of the B-T estimates.
- (c) It can be seen in Figure 5.23ab that neither window does a good job of estimating the true spectrum with a window length less than $L = 128$. At this length, all four frequency components are visible. For lengths less than $L = 128$, the Parzen window is more tightly bound around the true spectrum, with the Bartlett window estimate having a larger envelope. This stands to reason, the Bartlett window used for part (a) has higher sidelobes than the Parzen window used in part (b). The higher the sidelobe the greater the spectral leakage from the true spectra.

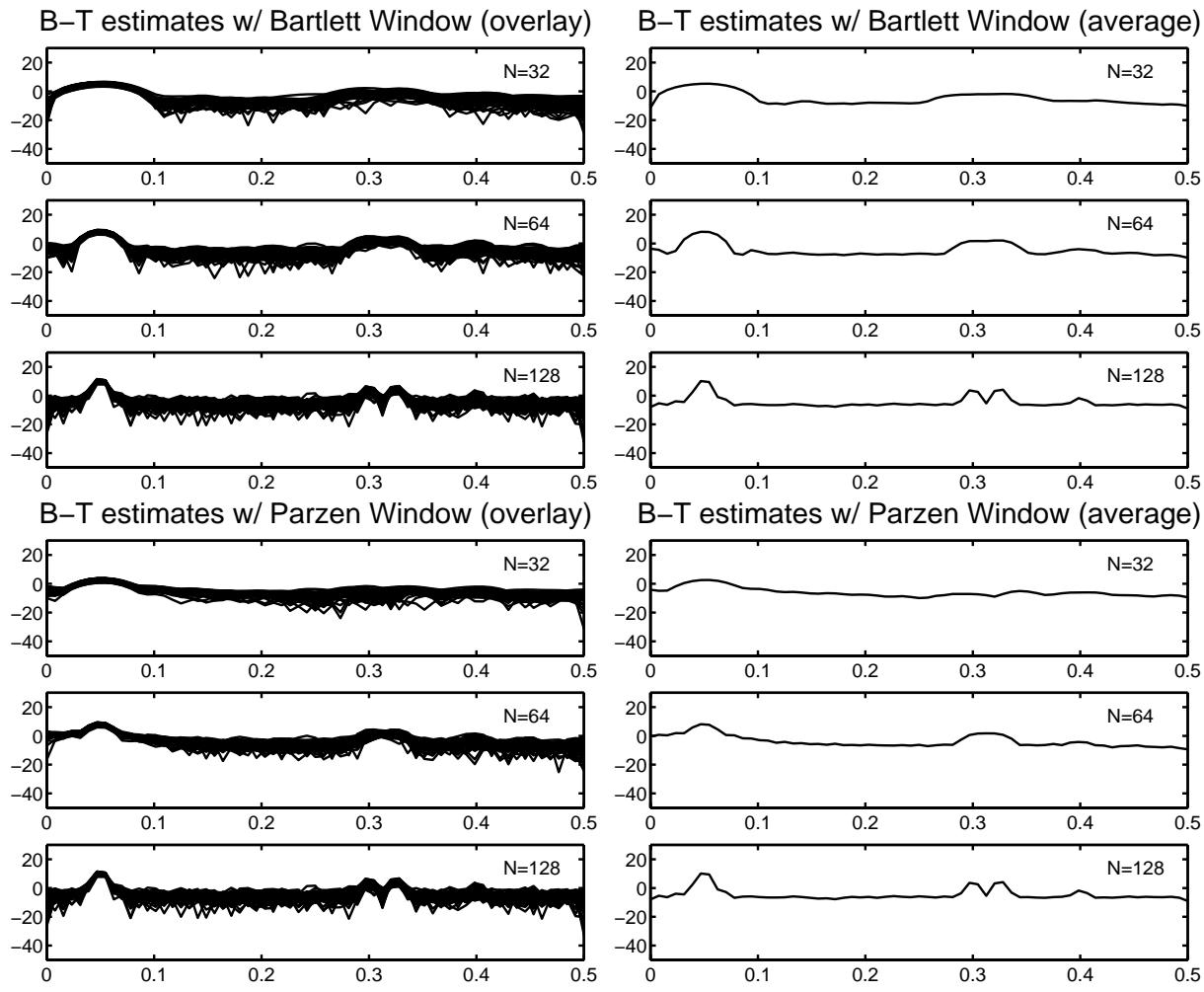


Figure 5.23ab:

5.24 Consider the random process given in Problem 5.23

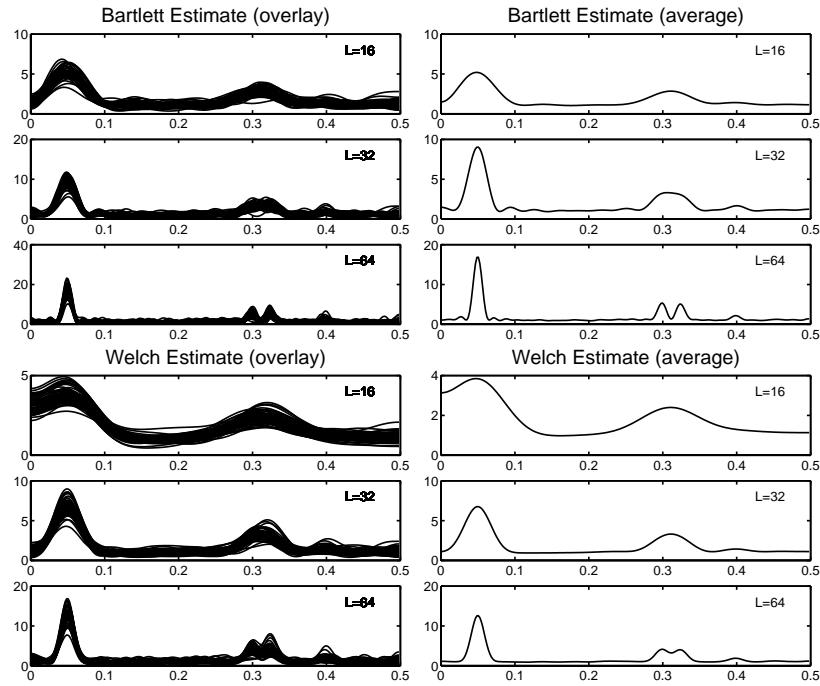


Figure 5.24ab:

- (a) The top row of Figure 5.24ab shows the Bartlett estimate in dB, which uses no overlap, a rectangular window, and $L = 32, 64, 128$. The left sided plots are the Bartlett estimates for each realization, while the right sided plots are the average of the Bartlett estimates.
- (b) The bottom row of Figure 5.24ab shows the Welch estimate in dB using 50% overlap, a Hamming window, and $L = 32, 64, 128$. The left sided plots are the Welch estimates for each realization, while the right sided plots are the average of the Welch estimates.
- (c) The Bartlett estimates have better resolution but slightly higher variance.

5.25 Consider the random process given in Problem 5.23.

- (a) Figure 5.25a shows the multitaper spectrum estimate, using $K = 3, 5, 7$ Slepian tapers. The left sided plot is the estimates for each realizations while the right sided plot is the average of the estimates.
- (b) As expected the multitaper spectral estimates have wider (and almost rectangular) peak and hence less resolution.

5.26 This problem deals with spectral estimation using three different techniques. A 1000 sample AR(1) process is generated with $a = -0.9$.

- (a) The top row of Figure 5.26 shows a periodogram estimate for the AR(1) process. It is not very accurate over the entire spectrum and has a high variance.
- (b) The bottom left side plot of Figure 5.26 shows a Blackman-Tukey estimate for $L = 10, 20, 50, 100$. Clearly, as L increases the estimated spectrum more closely matches the true spectrum.
- (c) The bottom right side plot of Figure 5.26 shows a Welch estimate, with 50% overlap, using a Hamming window, for $L = 10, 20, 50, 100$. Clearly, as L increases the estimated spectrum more closely matches the true spectrum. This appears to be the best estimator of the three.

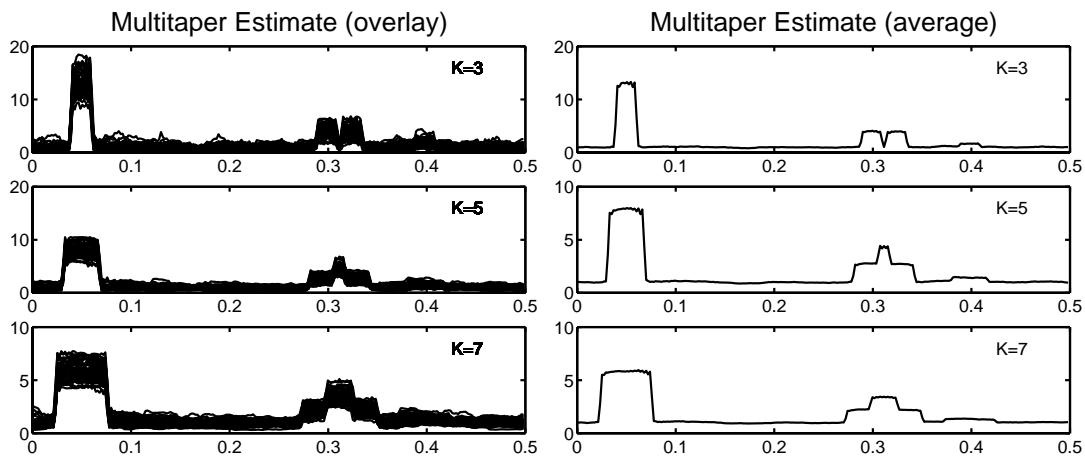


Figure 5.25a:

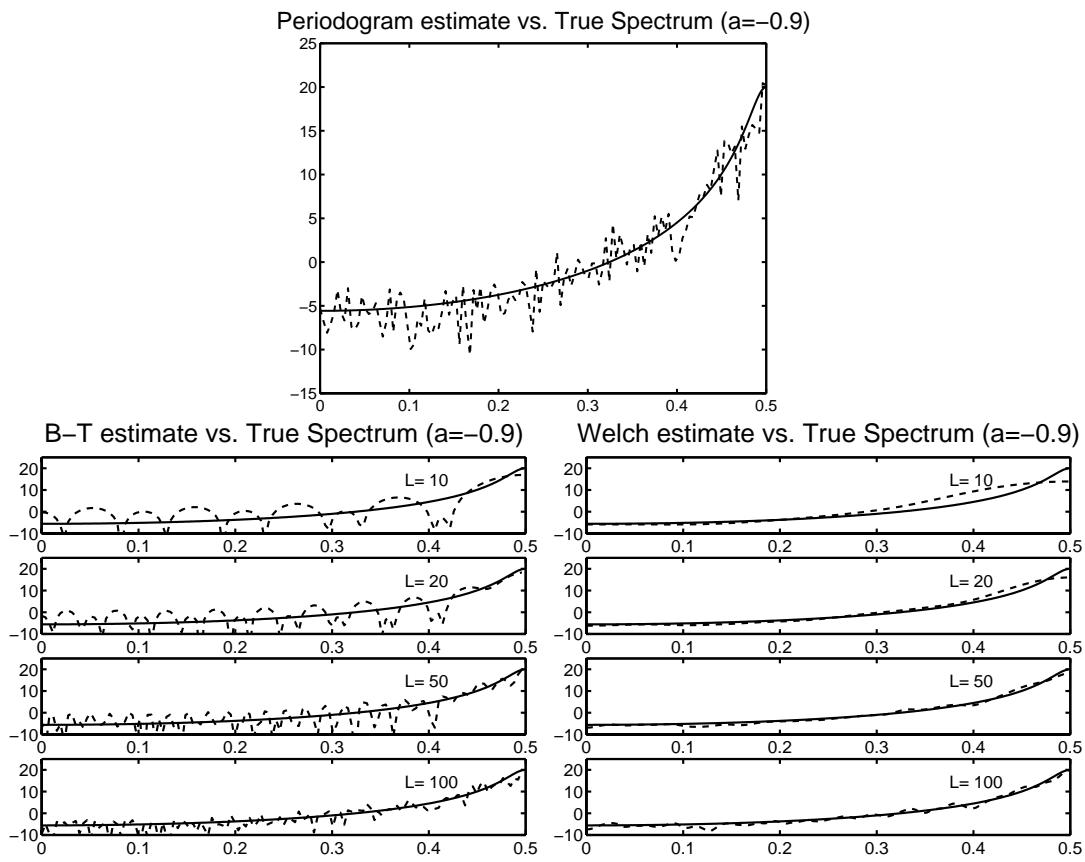


Figure 5.26:

5.27 This problem deals with spectral estimation using three different techniques. A 1000 sample AR(1) process is generated with $a = 0.9$.

- (a) The top row of Figure 5.27 shows a periodogram estimate for the AR(1) process. It is not very accurate over the entire spectrum and has a high variance.
- (b) The bottom left side plot of Figure 5.27 shows a Blackman-Tukey estimate for $L = 10, 20, 50, 100$. Clearly, as L increases the estimated spectrum more closely matches the true spectrum.
- (c) The bottom right side plot of Figure 5.27 shows a Welch estimate, with 50% overlap, using a Hamming window, for $L = 10, 20, 50, 100$. Clearly, as L increases the estimated spectrum more closely matches the true spectrum. This appears to be the best estimator of the three.

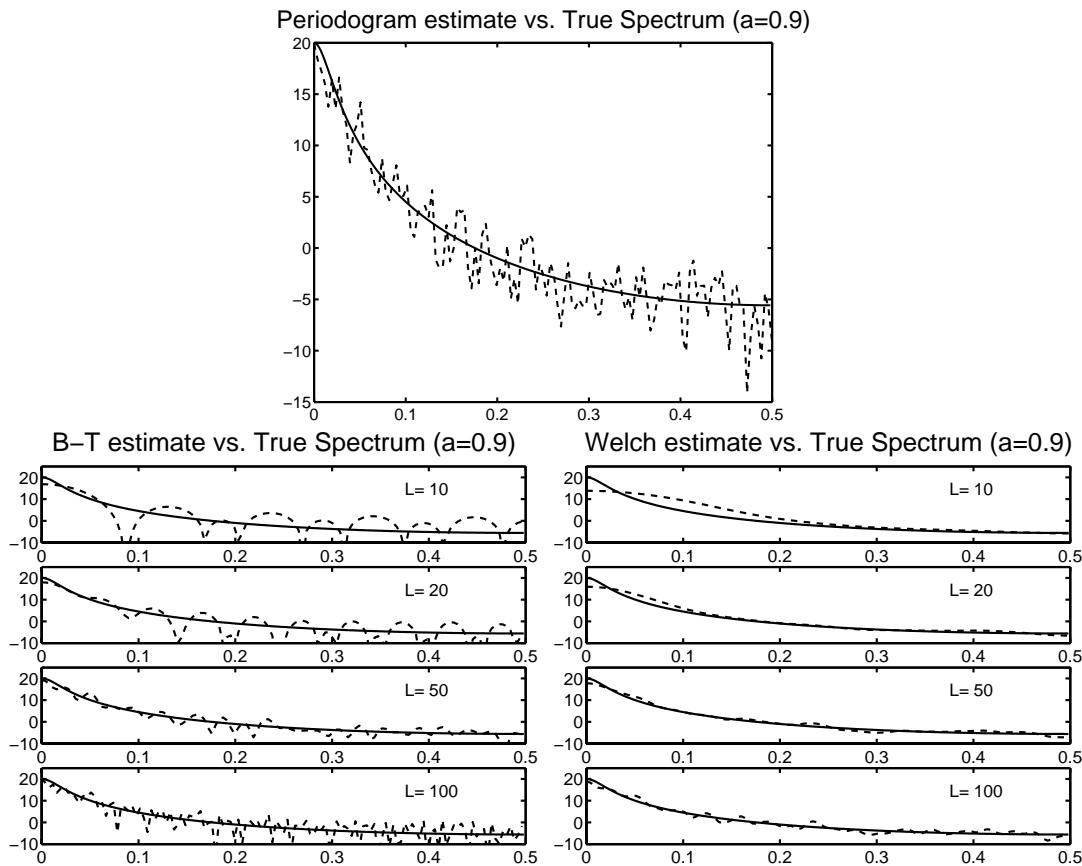


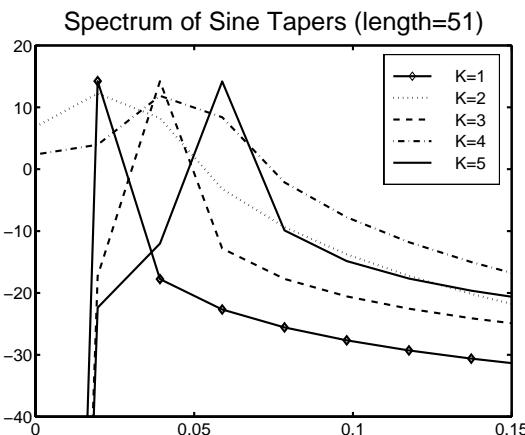
Figure 5.27:

5.28 Multitaper estimation technique requires a properly designed orthonormal set of tapers for proper performance. One set discussed in the chapter is that of harmonically related sinusoids given by

$$\omega_k(n) = \sqrt{\frac{2}{N-1}} \sin \frac{\pi(k+1)(n+1)}{N+1} \quad n = 0, 1, \dots, N-1$$

- (a) Listed below is a MATLAB functions that will generate $K < N$ sinusoidal tapers of length N .

```
function w = sine_tapers(N, K)
k = [1:K]'; n = [0:N-1]; w = sqrt(2/(N-1))* sin((pi*(k+1)*(n+1))/(N+1));
```

**Figure 5.28b:**

(b) Figure 5.28b plots the first five tapers of length 51.

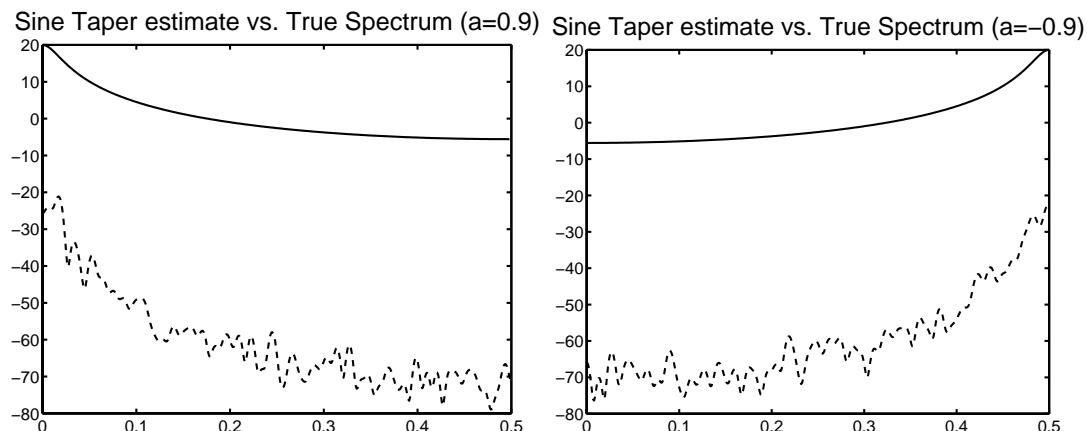
5.29 The following MATLAB function determines the multitaper estimate using the sine tapers generated above

```
function Pxx = psd_sinetaper(x, K)

N = length(x); w = sine_tapers(N,K);
for k = 1:K
    x_w(k,:) = x.*w(1,:);
    S_x(:,k) = psd(x_w(k,:),N , 1, ones(N/10,1),0);
end
Pxx = mean(S_x');
```

(a) The left side of Figure 5.29 shows the estimate of the AR(1) process given in Problem 5.26

(b) The right side of Figure 5.29 shows the estimate of the AR(1) process given in Problem 5.27

**Figure 5.29:**

Chapter 6
Optimum Linear Filters

- 6.1** Show that the linear MMSE estimate \hat{y} of a random variable y using the data vector \mathbf{x} is given by $\hat{y} = y_o + \mathbf{c}^H \mathbf{x}$, where $y_o = E\{y\} - \mathbf{c}^H E\{\mathbf{x}\}$, $\mathbf{c} = \mathbf{R}^{-1} \mathbf{d}$, $\mathbf{R} = E\{\mathbf{x}\mathbf{x}^H\}$, and $\mathbf{d} = E\{\mathbf{x}y^*\}$.

Let

$$\varepsilon^2 = |\hat{y} - y|^2 = \hat{y}\hat{y}^* - \hat{y}y^* - yy^* + yy^*$$

Now, \hat{y} is a linear estimate of y , therefore

$$\hat{y} = y_o + \mathbf{c}^H \mathbf{x}$$

Substituting the expression for \hat{y} into the above expression for ε^2 , computing its expected value and taking the derivative with respect to y_o results in

$$\frac{dE\varepsilon^2}{dy_o} = 2y_o + 2\mu_x^H c - 2\mu_y$$

Setting the above expression equal to zero yields $y_o = \mu_y - c^H \mu_x$

- 6.2** Consider an optimum FIR filter specified by the input correlation matrix \mathbf{R} = Toeplitz $\{1, \frac{1}{4}\}$ and cross-correlation vector $\mathbf{d} = [1 \ \frac{1}{2}]^T$.

- (a) Determine the optimum impulse response \mathbf{c}_o and the MMSE P_o

Using (6.2.12), the optimum impulse response \mathbf{c}_o is

$$R\mathbf{c}_o = \mathbf{d} \Rightarrow \mathbf{c}_o = R^{-1}\mathbf{d} = [0.933 \ 0.267]^T$$

and, using (6.2.17), the MMSE P_o is

$$P_o = P_y - d^H c_o = P_y - [1 \ \frac{1}{2}] \begin{bmatrix} 0.933 \\ 0.267 \end{bmatrix} = P_y - 1.067$$

- (b) Express \mathbf{c}_o and the MMSE P_o in terms of the eigenvalues and eigenvectors of \mathbf{R}

The eigenvalues and eigenvectors of \mathbf{R} are

$$\mathbf{R} = Q\Lambda Q^H = \begin{bmatrix} 0.7071 & 0.7071 \\ -0.7071 & 0.7071 \end{bmatrix} \begin{bmatrix} 0.75 & 0 \\ 0 & 1.25 \end{bmatrix} \begin{bmatrix} 0.7071 & -0.7071 \\ 0.7071 & 0.7071 \end{bmatrix}$$

Using (6.2.35) and (6.2.32), c_o and P_o are found using the eigenvalues and eigenvectors as is shown below

$$\begin{aligned} c_o &= Q\Lambda Q^H d = \sum_{i=1}^2 \frac{q_i^H d}{\lambda_i} q_i \\ &= \frac{1}{0.75} [0.7071 \ -0.7071] \begin{bmatrix} 1 \\ \frac{1}{2} \end{bmatrix} \begin{bmatrix} 0.7071 \\ -0.7071 \end{bmatrix} + \frac{1}{1.25} [0.7071 \ 0.7071] \begin{bmatrix} 1 \\ \frac{1}{2} \end{bmatrix} \begin{bmatrix} 0.7071 \\ 0.7071 \end{bmatrix} \\ &= \begin{bmatrix} 0.333 \\ -0.333 \end{bmatrix} + \begin{bmatrix} 0.6 \\ 0.6 \end{bmatrix} = \begin{bmatrix} 0.933 \\ 0.267 \end{bmatrix} \end{aligned}$$

$$P_o = P_y - \sum_{i=1}^2 \frac{|d'_i|^2}{\lambda} = P_y - \frac{(0.3536)^2}{0.75} - \frac{(1.0607)^2}{1.25} = P_y - 1.067$$

where $d' = Q^H d = [0.3536 \ 1.0607]^T$

- 6.3** Repeat Problem 6.2 for a third-order optimum FIR filter, with \mathbf{R} = Toeplitz $\{1, \frac{1}{4}, 0\}$ and cross-correlation vector $\mathbf{d} = [1 \ \frac{1}{2} \ 0]^T$.

(a) Solving the same way as Problem 6.2, using (6.2.12) and (6.2.17) results in

$$c_o = R^{-1}d = \begin{bmatrix} 0.9286 \\ 0.2857 \\ -0.0714 \end{bmatrix}$$

and

$$P_o = P_y - d^H c_o = P_y - 1.071$$

(b) The eigenvalues and eigenvectors of \mathbf{R} are

$$Q = \begin{bmatrix} 0.5 & -0.7071 & -0.5 \\ 0.7071 & 0 & 0.7071 \\ 0.5 & 0.7071 & -0.5 \end{bmatrix}$$

and

$$\Lambda = \begin{bmatrix} 1.3536 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.6464 \end{bmatrix}$$

Using (6.2.35) and (6.2.32), c_o and P_o are

$$c_o = \begin{bmatrix} 0.315 \\ 0.446 \\ 0.315 \end{bmatrix} + \begin{bmatrix} 0.5 \\ 0 \\ -0.5 \end{bmatrix} + \begin{bmatrix} 0.113 \\ -0.160 \\ 0.113 \end{bmatrix} = \begin{bmatrix} 0.9286 \\ 0.2857 \\ -0.0714 \end{bmatrix}$$

and

$$P_o = P_y - \frac{(0.8536)^2}{1.3536} - \frac{(0.7071)^2}{1} - \frac{(0.1464)^2}{0.6464} = P_y - 1.071$$

- 6.4** A process $y(n)$ with the autocorrelation $r_y(l) = a^{|l|}$, $-1 < a < 1$, is corrupted by additive, uncorrelated white noise $v(n)$ with variance σ_v^2 . To reduce the noise in the observed process $x(n) = y(n) + v(n)$, we use a first-order Wiener filter.

(a) Express the coefficients $c_{o,1}$ and $c_{o,2}$ and the MMSE P_o in terms of the parameters a and σ_v^2
The autocorrelation matrix for $x(n)$ is

$$R_x = E\{x(n)x^H(n)\} = R_y + \sigma_v^2 I = \begin{bmatrix} 1 + \sigma_v^2 & a \\ a & 1 + \sigma_v^2 \end{bmatrix}$$

and the cross-correlation matrix is

$$d = E\{x(n)y^*(n)\} = E \begin{bmatrix} y(n)y^*(n) + v(n)y^*(n) \\ y(n+1)y^*(n) + v(n+1)y^*(n) \end{bmatrix} = \begin{bmatrix} 1 \\ a \end{bmatrix}$$

Solving for the filter coefficients using $c_o = R^{-1}d$ yields the following

$$c_o = ((1 + \sigma_v^2)^2 - a^2) \begin{bmatrix} 1 + \sigma_v^2 & -a \\ -a & 1 + \sigma_v^2 \end{bmatrix} \begin{bmatrix} 1 \\ a \end{bmatrix} = \begin{bmatrix} \frac{1 + \sigma_v^2 - a^2}{((1 + \sigma_v^2)^2 - a^2)} \\ \frac{\sigma_v^2 a}{((1 + \sigma_v^2)^2 - a^2)} \end{bmatrix}$$

The MMSE P_o is found using $P_o = P_y - d^H c_o$ where $P_y = E|y|^2 = 1$. After simplifying, the MMSE P_o is

$$P_o = 1 - [1 \ a] \begin{bmatrix} \frac{1 + \sigma_v^2 - a^2}{((1 + \sigma_v^2)^2 - a^2)} \\ \frac{\sigma_v^2 a}{((1 + \sigma_v^2)^2 - a^2)} \end{bmatrix} = \sigma_v^2(1 + a^2 + \sigma_v^2)$$

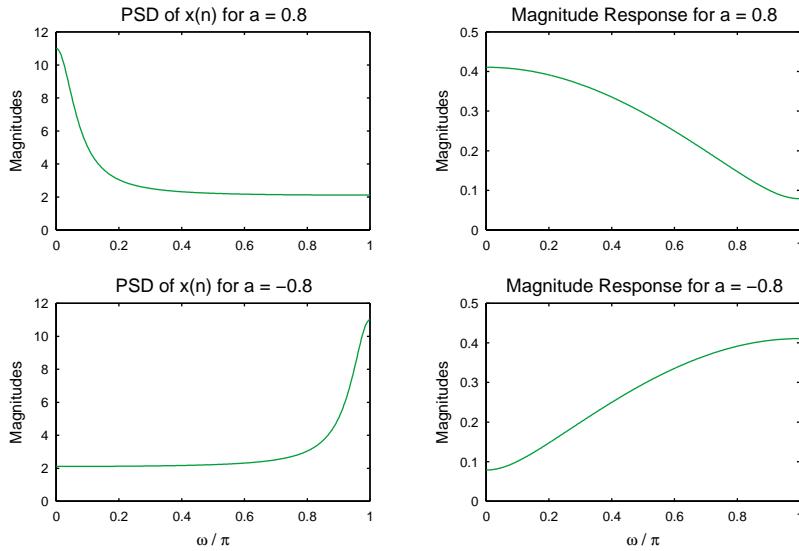
- (b) Compute and plot the PSD of $x(n)$ and the magnitude response $|C_o(e^{j\omega})|$ of the filter when $\sigma_v^2 = 2$, for both $a = 0.8$ and $a = -0.8$, and compare the results.

```
% (b) Plot of the PSD of x(n) and the Mag Res of the optimal filter
Hf_1 = figure('units','SCRUN','position',SCRPOS,...
    'paperunits','PAPUN','paperposition',[0,0,5,3]);
set(Hf_1,'NumberTitle','off','Name','Pr0604b');

% Given parameters
var_v = 2;
% 1. a = 0.8
a = 0.8;
% Computed Parameters
c_o1 = (1+var_v-a*a)/((1+var_v)*(1+var_v)+a*a);
c_o2 = (var_v*a)/((1+var_v)*(1+var_v)+a*a);
% PSD of x(n)
omg = [0:500]*pi/500;
Rx = (2-(1+a*a))./((1+a*a)-2*a*cos(omg)) + var_v;
% Magnitude plot of the optimal filter
C_o = freqz([c_o1,c_o2],1,omg);
magC_o = abs(C_o);

subplot(2,2,1);
plot(omg/pi,Rx,'g'); axis([0,1,0,12]);
% xlabel('\omega / \pi','fontsize',label_fontsize);
ylabel('Magnitudes','fontsize',label_fontsize);
set(gca,'xtick',[0:0.2:1],'ytick',[0:2:12]);
title('PSD of x(n) for a = 0.8','fontsize',title_fontsize);

subplot(2,2,2);
plot(omg/pi,magC_o,'g'); axis([0,1,0,0.5]);
% xlabel('\omega / \pi','fontsize',label_fontsize);
ylabel('Magnitudes','fontsize',label_fontsize);
```

**Figure 6.4b:** Spectral plots

```

set(gca,'xtick',[0:0.2:1], 'ytick',[0:.1:0.5]);
title('Magnitude Response for a = 0.8','fontsize',title_fontsize);

% 2. a = -0.8
a = -0.8;
% Computed Parameters
c_o1 = (1+var_v-a*a)/((1+var_v)*(1+var_v)+a*a);
c_o2 = (var_v*a)/((1+var_v)*(1+var_v)+a*a);
% PSD of x(n)
omg = [0:500]*pi/500;
Rx = (2-(1+a*a))./((1+a*a)-2*a*cos(omg)) + var_v;
% Magnitude plot of the optimal filter
C_o = freqz([c_o1,c_o2],1,omg);
magC_o = abs(C_o);

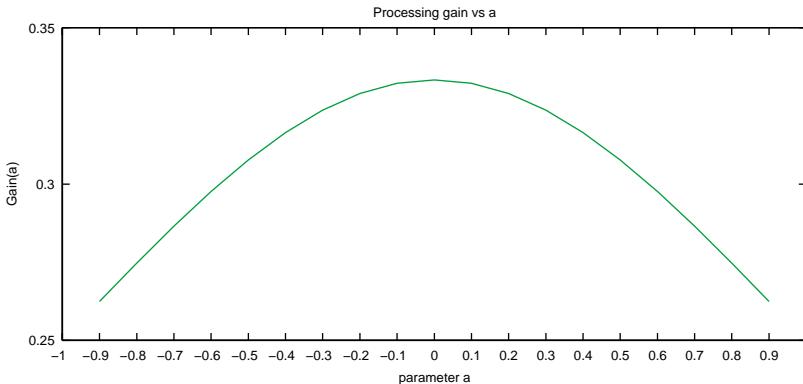
subplot(2,2,3);
plot(omg/pi,Rx,'g'); axis([0,1,0,12]);
xlabel('\omega / \pi','fontsize',label_fontsize);
ylabel('Magnitudes','fontsize',label_fontsize);
set(gca,'xtick',[0:0.2:1], 'ytick',[0:2:12]);
title('PSD of x(n) for a = -0.8','fontsize',title_fontsize);

subplot(2,2,4);
plot(omg/pi,magC_o,'g'); axis([0,1,0,0.5]);
xlabel('\omega / \pi','fontsize',label_fontsize);
ylabel('Magnitudes','fontsize',label_fontsize);
set(gca,'xtick',[0:0.2:1], 'ytick',[0:.1:0.5]);
title('Magnitude Response for a = -0.8','fontsize',title_fontsize);

```

The plot is shown in Figure 6.4b.

- (c) Compute and plot the processing gain of the filter for $a = -0.9, -0.8, -0.7, \dots, 0.9$ as a function of a

**Figure 6.4c:** Plot of processing gain

and comment on the results.

```
% (c) Processing Gain
a = -0.9:0.1:0.9;
PG = 1./(1+var_v+a.*a);
Hf_1 = figure('units','SCRUN','position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',[0,0,5,2]);
set(Hf_1,'NumberTitle','off','Name','Pr0604c');
plot(a,PG,'g'); axis([-1,1,0.25,0.35]);
xlabel('parameter a','fontsize',label_fontsize);
ylabel('Gain(a)','fontsize',label_fontsize);
set(gca,'xtick',[-1:0.1:1],'ytick',[0.25:.05:0.35]);
title('Processing gain vs a','fontsize',title_fontsize);
```

The plot is shown in Figure 6.4c.

6.5 Consider the harmonic process $y(n)$ and its noise observation $x(n)$ given in Example 6.4.1

From Example 6.4.1, $y(n) = A \cos(\omega_o n + \phi)$ where ϕ is uniformly distributed on $[0, 2\pi]$. The noise observation is $x(n) = y(n) + v(n)$ with $v(n) \sim N(0, \sigma_v^2)$

(a) Show that $r_y(l) = \frac{1}{2}A^2 \cos \omega_o l$

Solving directly, it is shown as follows

$$\begin{aligned} r_y(l) &= E\{y(n)y^*(n+l)\} \\ &= E\{A^2 \cos(\omega_o n + \phi) \cos(\omega_o(n+l) + \phi)\} \\ &= A^2 E\{\frac{1}{2} \cos(\omega_o l) + \frac{1}{2} \cos(2\omega_o(n+l) + 2\phi)\} \\ &= \frac{1}{2} A^2 [\cos(\omega_o l) + \cos(2\omega_o(n+l)) E[\cos \phi] - \sin(2\omega_o(n+l)) E[\sin \phi]] \\ &= \frac{1}{2} A^2 \cos \omega_o l \end{aligned}$$

(b) The following MATLAB function will compute an Mth-order optimum FIR filter impulse response $h(n)$

```
function [h] = opt_fir(A,f0,var_v,M);
%
% Designs the Mth order optimum filter h(n)
% where y(n) = A cos(2 pi f0 n + phi)
```

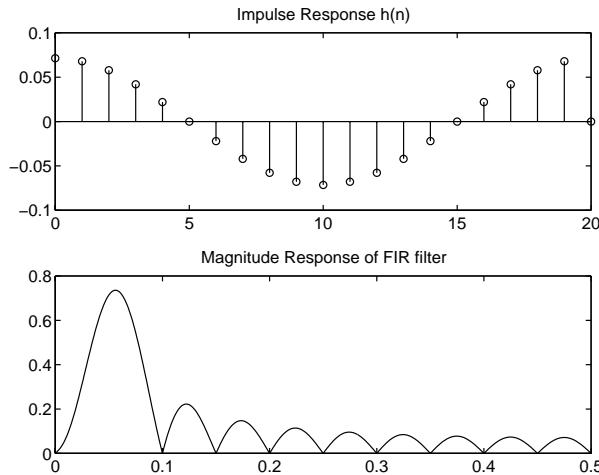


Figure 6.5: 20th-order optimum FIR filter

```
% x(n) = y(n) + nu(n)
% nu(n) = Normal(0, var_v)

l=[0:M-1]';

ry=((A^2)/2)*cos(2*pi*f0.*l); rv=var_v.*[1 ; zeros(M-1,1)];
rx=ry+rv;

Rxx=toeplitz(rx);

h=inv(Rxx)*ry;
```

- (c) Figure 6.5 shows the impulse response of the 20th-order optimum FIR filter for $A = 0.5$, $f_o = 0.05$, and $\sigma_v^2 = 0.5$.
- (d) Figure 6.5 shows the impulse and magnitude response. Note that this matches Example 6.4.1

- 6.6** Consider a “desired” signal $s(n)$ generated by the process $s(n) = -0.8w(n-1) + w(n)$, where $w(n) \sim WN(0, \sigma_w^2)$. This signal is passed through the causal system $H(z) = 1 - 0.9z^{-1}$ whose output $y(n)$ is corrupted by additive white noise $v(n) \sim WN(0, \sigma_v^2)$. The processes $w(n)$ and $v(n)$ are uncorrelated with $\sigma_w^2 = 0.3$ and $\sigma_v^2 = 0.1$.

- (a) Design a second-order optimum FIR filter that estimates $s(n)$ from the signal $x(n) = y(n) + v(n)$ and determine \mathbf{c}_o and P_o :

```
% Given parameters
var_w = 0.3;
h1 = [1,-0.8]; lh1 = length(h1);
h = [ 1,-0.9];
var_v = 0.1;

% Computed Parameters
% Autocorrelation of s(n): rs(l) = var_w*h1(l)*h1(-l)
```

```

rh1 = conv(h1,fliplr(h1)); lrh1 = length(rh1);
rs = var_w*rh1; rs = rs(lh1:end); Ps = rs(1);
% Signal x(n) = h(n)*h1(n)*w(n) + v(n); h2 def h1*h;
h2 = conv(h,h1); lh2 = length(h2);
% Autocorrelation of x(n): rx(1) = var_w*h2(1)*h2(-1)+var_v*delta(1)
rx = var_w*conv(h2,fliplr(h2))+var_v*[zeros(1,lh2-1),1,zeros(1,lh2-1)];
rx = rx(lh2:end);
% Crosscorrelation between x(n) and s(n): rxs(1) = var_w*h(1)*h1(1)*h1(-1)
lrh1c = round((lrh1-1)/2)+1;
rxs = var_w*conv(h,rh1); rxs = rxs(lrh1c:end);

% (a) Second-order (i.e., length M = 2) optimal FIR filter design
M = 2;
% Optimal filter
Rx = toeplitz(rx(1:M),rx(1:M));
dxs = rxs(1:M)';
co = Rx\dxs;
% Optimal error
Po = Ps - dxs'*co;
% Printout
Second-order (i.e., length M = 2) optimal FIR filter design
Optimal FIR filter coefficients:  0.3255, -0.2793
Optimal error = 0.0709

```

- (b) Plot the error performance surface, and verify that it is quadratic and that the optimum filter points to its minimum.

```

% (b) Error performance surface
Hf_1 = figure('units','SCRUN','position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',[0,0,5,3]);
set(Hf_1,'NumberTitle','off','Name','Pr0606');

[co0,co1] = meshgrid([co(1)-1:0.05:co(1)+1],[co(2)-1:0.01:co(2)+1]);
P = Ps - 2*dxs(1)*co0 - 2*dxs(2)*co1 + rx(1)*co0.^2 + 2*rx(2)*co0.*co1...
    + rx(1)*co1.^2;
V = [0:0.05:1]+Po;
contour(co0,co1,P,V);
xlabel('c_o(1)', 'fontsize',label_fontsize);
ylabel('c_o(2)', 'fontsize',label_fontsize);
set(gca,'xtick',[co(1)-1:1:co(1)+1],'ytick',[co(2)-1:1:co(2)+1]);
title('P6.6 : Error Performance Surface', 'fontsize',title_fontsize);
grid;

```

The plot is shown in Figure 6.6.

- (c) Repeat part (a) for a third-order filter, and see whether there is any improvement.

```

% (c) Third-order (i.e., length M = 3) optimal FIR filter design
M = 3;
% Optimal filter
Rx = toeplitz(rx(1:M),rx(1:M));
dxs = rxs(1:M)';

```

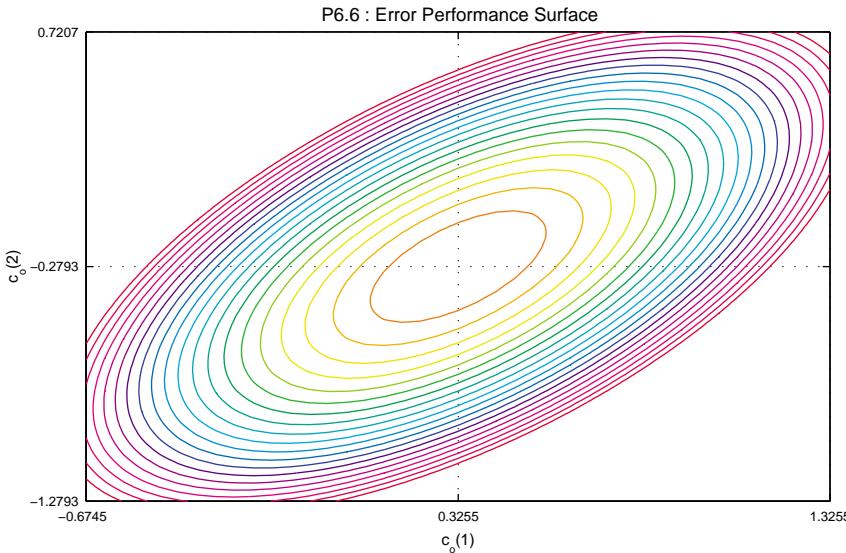


Figure 6.6: Plot of error surface

```

co = Rx\dxs;
% Optimal error
Po = Ps - dxs'*co;
% Printout
Third-order (i.e., length M = 3) optimal FIR filter design
Optimal FIR filter coefficients: 0.2775, -0.3893, -0.1303
Optimal error = 0.0579

```

6.7 Repeat Problem 6.6, assuming that the desired signal is generated by $s(n) = -0.8s(n-1) + w(n)$.

- (a) Design a second-order optimum FIR filter that estimates $s(n)$ from the signal $x(n) = y(n) + v(n)$ and determine \mathbf{c}_o and P_o :

```

% Given parameters
var_w = 0.3;
n = 0:20;
h1 = (-0.8).^n; lh1 = length(h1);
h = [ 1,-0.9];
var_v = 0.1;

% Computed Parameters
% Autocorrelation of s(n): rs(1) = var_w*h1(1)*h1(-1)
rh1 = conv(h1,fliplr(h1)); lrh1 = length(rh1);
rs = var_w*rh1; rs = rs(lh1:end); Ps = rs(1);
% Signal x(n) = h(n)*h1(n)*w(n) + v(n); h2 def h1*h;
h2 = conv(h,h1); lh2 = length(h2);
% Autocorrelation of x(n): rx(1) = var_w*h2(1)*h2(-1)+var_v*delta(1)
rx = var_w*conv(h2,fliplr(h2))+var_v*[zeros(1,lh2-1),1,zeros(1,lh2-1)];
rx = rx(lh2:end);
% Crosscorrelation between x(n) and s(n): rxs(1) = var_w*h(1)*h1(1)*h1(-1)
lrh1c = round((lrh1-1)/2)+1;
rxs = var_w*conv(h,rh1); rxs = rxs(lrh1c:end);

```

```
% (a) Second-order (i.e., length M = 2) optimal FIR filter design
M = 2;
% Optimal filter
Rx = toeplitz(rx(1:M),rx(1:M));
dxs = rxs(1:M)';
co = Rx\dxs;
% Optimal error
Po = Ps - dxs'*co;
% Printout
Second-order (i.e., length M = 2) optimal FIR filter design
Optimal FIR filter coefficients:  0.2941, -0.2493
Optimal error = 0.0587
```

- (b) Plot the error performance surface, and verify that it is quadratic and that the optimum filter points to its minimum.

```
% (b) Error performance surface
Hf_1 = figure('units','SCRUN','position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',[0,0,5,3]);
set(Hf_1,'NumberTitle','off','Name','Pr0607');

[co0,co1] = meshgrid([co(1)-1:0.05:co(1)+1],[co(2)-1:0.01:co(2)+1]);
P = Ps - 2*dxs(1)*co0 - 2*dxs(2)*co1 + rx(1)*co0.^2 + 2*rx(2)*co0.*co1...
    + rx(1)*co1.^2;
V = [0:0.05:1]+Po;
contour(co0,co1,P,V);
xlabel('c_o(1)', 'fontsize',label_fontsize);
ylabel('c_o(2)', 'fontsize',label_fontsize);
set(gca,'xtick',[co(1)-1:1:co(1)+1],'ytick',[co(2)-1:1:co(2)+1]);
title('P6.7 : Error Performance Surface','fontsize',title_fontsize);
grid;
```

The plot is shown in Figure 6.7.

- (c) Repeat part (a) for a third-order filter, and see whether there is any improvement.

```
% (c) Third-order (i.e., length M = 3) optimal FIR filter design
M = 3;
% Optimal filter
Rx = toeplitz(rx(1:M),rx(1:M));
dxs = rxs(1:M)';
co = Rx\dxs;
% Optimal error
Po = Ps - dxs'*co;
% Printout
Third-order (i.e., length M = 3) optimal FIR filter design
Optimal FIR filter coefficients:  0.2769, -0.3269, -0.0723
Optimal error = 0.0552
```

- 6.8** Repeat Problem 6.6, assuming that $H(z) = 1$.

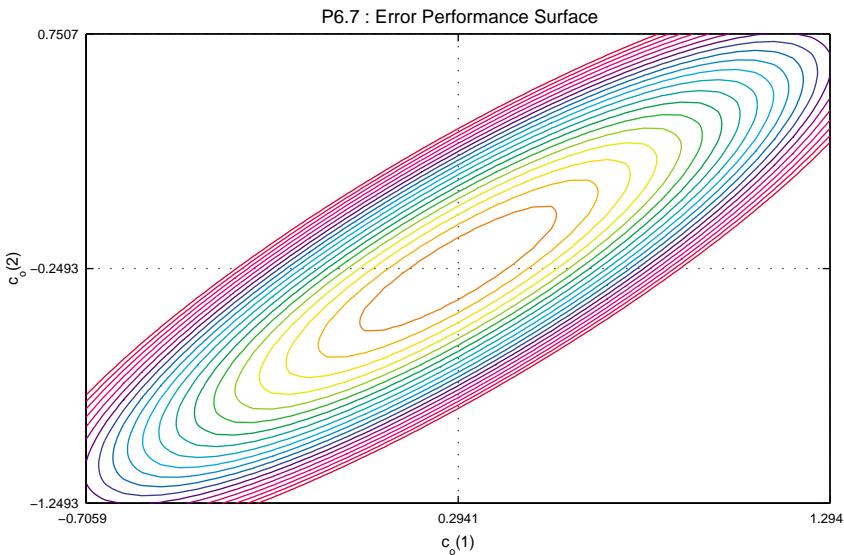


Figure 6.7: Plot of error surface

- (a) Design a second-order optimum FIR filter that estimates $s(n)$ from the signal $x(n) = y(n) + v(n)$ and determine \mathbf{c}_o and P_o :

```
% Given parameters
var_w = 0.3;
h1 = [1,-0.8]; lh1 = length(h1);
h = [ 1];
var_v = 0.1;

% Computed Parameters
% Autocorrelation of s(n): rs(1) = var_w*h1(1)*h1(-1)
rh1 = conv(h1,fliplr(h1)); lrh1 = length(rh1);
rs = var_w*rh1; rs = rs(lh1:end); Ps = rs(1);
% Signal x(n) = h(n)*h1(n)*w(n) + v(n); h2 def h1*h;
h2 = conv(h,h1); lh2 = length(h2);
% Autocorrelation of x(n): rx(1) = var_w*h2(1)*h2(-1)+var_v*delta(1)
rx = var_w*conv(h2,fliplr(h2))+var_v*[zeros(1,lh2-1),1,zeros(1,lh2-1)];
rx = [rx(lh2:end),0,0];
% Crosscorrelation between x(n) and s(n): rxs(1) = var_w*h(1)*h1(1)*h1(-1)
lrh1c = round((lrh1-1)/2)+1;
rxs = var_w*conv(h,rh1); rxs = [rxs(lrh1c:end),0,0];

% (a) Second-order (i.e., length M = 2) optimal FIR filter design
M = 2;
% Optimal filter
Rx = toeplitz(rx(1:M),rx(1:M));
dxs = rxs(1:M)';
co = Rx\dxs;
% Optimal error
Po = Ps - dxs'*co;
% Printout
```

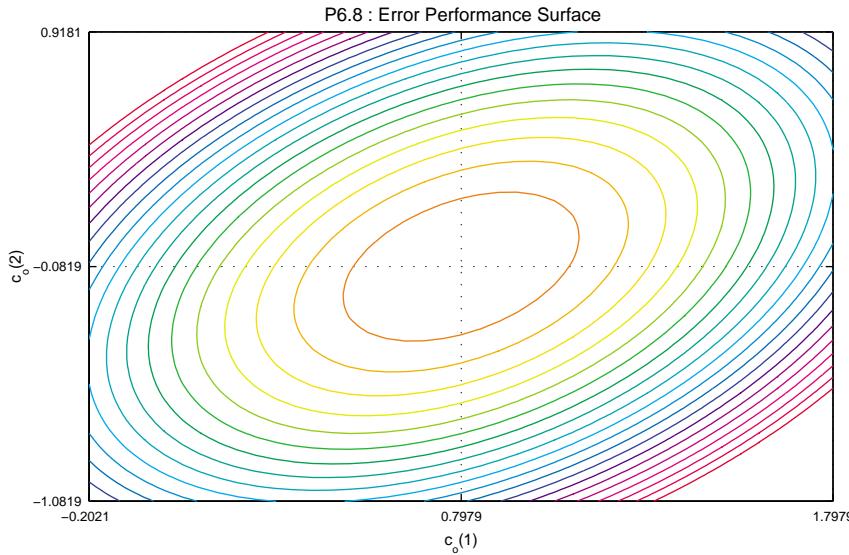


Figure 6.8: Plot of error surface

Second-order (i.e., length $M = 2$) optimal FIR filter design
 Optimal FIR filter coefficients: 0.7979, -0.0819
 Optimal error = 0.0798

- (b) Plot the error performance surface, and verify that it is quadratic and that the optimum filter points to its minimum.

```
% (b) Error performance surface
Hf_1 = figure('units','SCRUN','position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',[0,0,5,3]);
set(Hf_1,'NumberTitle','off','Name','Pr0608');

[co0,co1] = meshgrid([co(1)-1:0.05:co(1)+1],[co(2)-1:0.01:co(2)+1]);
P = Ps - 2*dxs(1)*co0 - 2*dxs(2)*co1 + rx(1)*co0.^2 + 2*rx(2)*co0.*co1...
    + rx(1)*co1.^2;
V = [0:0.05:1]+Po;
contour(co0,co1,P,V);
xlabel('c_o(1)', 'fontsize',label_fontsize);
ylabel('c_o(2)', 'fontsize',label_fontsize);
set(gca,'xtick',[co(1)-1:1:co(1)+1],'ytick',[co(2)-1:1:co(2)+1]);
title('P6.8 : Error Performance Surface', 'fontsize',title_fontsize);
grid;
```

The plot is shown in Figure 6.8.

- (c) Repeat part (a) for a third-order filter, and see whether there is any improvement.

```
% (c) Third-order (i.e., length M = 3) optimal FIR filter design
M = 3;
% Optimal filter
Rx = toeplitz(rx(1:M),rx(1:M));
dxs = rxs(1:M)';
co = Rx\dxs;
% Optimal error
```

```

Po = Ps - dxs'*co;
% Printout
Third-order (i.e., length M = 3) optimal FIR filter design
Optimal FIR filter coefficients: 0.7897, -0.1020, -0.0414
Optimal error = 0.0790

```

- 6.9** A stationary process $x(n)$ is generated by the difference equation $x(n) = \rho x(n-1) + w(n)$, where $w(n) \sim \text{WN}(0, \sigma_w^2)$

- (a) Show that the correlation matrix of $x(n)$ is given by

$$R_x = \frac{\sigma_w^2}{1-\rho^2} \text{Toeplitz}\{1, \rho, \rho^2, \dots, \rho^{M-1}\}$$

Computing the correlation directly

$$\begin{aligned} r_x(l) &= E\{x(n)x^*(n-l)\} = E\{(\rho x(n-1) + w(n))x^*(n-l)\} \\ &= \rho E\{x(n-1)x^*(n-l)\} = \rho r_x(l-1) = \rho^2 r_x(l-2) = \dots \\ &= \rho^l r_x(0) \end{aligned}$$

Solving for $r_x(0)$

$$\begin{aligned} r_x(0) &= E\{\rho^2 x(n-1)x^*(n-1) + \rho x(n-1)w^*(n) + \rho^* x^*(n)w(n-1) + w(n-1)w^*(n-1)\} \\ &= \rho^2 r_x(0) + \sigma_w^2 \end{aligned}$$

Simplifying, and combining with the result above yields

$$r_x(l) = \frac{\sigma_w^2}{1-\rho^2} \rho^l$$

Therefore, the correlation matrix is

$$R_x = \frac{\sigma_w^2}{1-\rho^2} \text{Toeplitz}\{1, \rho, \rho^2, \dots, \rho^{M-1}\}$$

- (b) Show that the Mth-order FLP is given by $a_1^{(M)} = -\rho$, $a_k^{(M)} = 0$ for $k > 1$ and the MMSE is $P_M^f = \sigma_w^2$.

The FLP, using (6.5.18), is $R(n-1)a_o(n) = -r(n)$. Expanding this relation below

$$\frac{\sigma_w^2}{1-\rho^2} \begin{bmatrix} 1 & \rho & \cdots \\ \rho & 1 & \ddots \\ \vdots & \ddots & \ddots \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \end{bmatrix} = -\frac{\sigma_w^2}{1-\rho^2} \begin{bmatrix} \rho \\ \rho^2 \\ \vdots \end{bmatrix}$$

It is clearly seen, that the right-hand side of the above equation is simply the first column of the autocorrelation matrix scaled by $-\rho$. Therefore $a = [-\rho 0 \cdots 0]^T$, or in other words, $a_1^{(M)} = -\rho$, $a_k^{(M)} = 0$ for $k > 1$.

The MMSE, using (6.5.19), is

$$P_o^f(n) = P_x(n) + r^{fH} a_o^{(M)} = \frac{\sigma_w^2}{1-\rho^2} - \frac{\sigma_w^2 \rho^2}{1-\rho^2} = \sigma_w^2$$

- 6.10** Using Parseval's theorem, show that (6.4.18) can be written as (6.4.21) in the frequency domain
Equation (6.4.18) is

$$P = E\{|e(n)|^2\} = r_y(0) - \sum_{k=0}^{M-1} h(k)r_{yx}^*(k) - \sum_{k=0}^{M-1} h^*(k)r_{yx}(k) - \sum_{k=0}^{M-1} \sum_{l=0}^{M-1} h(k)r_{yx}(l-k)h^*(l)$$

Parseval's relation (6.4.20) is

$$\sum_{n=-\infty}^{\infty} x_1(n)x_2^*(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X_1(e^{j\omega})X_2^*(e^{j\omega})d\omega$$

Applying Parseval's relation to (6.4.18) and using the fact that $h(k)$ is non-zero over $[0, M-1]$, we have

$$\begin{aligned} \sum_0^{M-1} h(k)r_{yx}^*(k) &= \frac{1}{2\pi} \int_{-\infty}^{\infty} H(e^{j\omega})R_{yx}^*(e^{j\omega})d\omega \\ \sum_0^{M-1} h^*(k)r_{yx}(k) &= \frac{1}{2\pi} \int_{-\infty}^{\infty} H^*(e^{j\omega})R_{yx}(e^{j\omega})d\omega \end{aligned}$$

and

$$\begin{aligned} \sum_{k=0}^{M-1} \sum_{l=0}^{M-1} h(k)r_{yx}(l-k)h^*(k) &= \sum_{k=0}^{M-1} h(k) \sum_{l=0}^{M-1} r_{yx}(l-k)h^*(l) \\ &= \sum_{k=0}^{M-1} h(k) \frac{1}{2\pi} \int_{-\infty}^{\infty} R_{yx}(e^{j\omega})e^{-j\omega k} H^*(e^{j\omega})d\omega \\ &= \frac{1}{2\pi} \int_{-\infty}^{\infty} \left\{ \sum_{k=0}^{M-1} h(k)e^{-j\omega k} \right\} R_{yx}(e^{j\omega})H^*(e^{j\omega})d\omega \\ &= \frac{1}{2\pi} \int_{-\infty}^{\infty} H(e^{j\omega})R_{yx}(e^{j\omega})H^*(e^{j\omega})d\omega \end{aligned}$$

Substituting these relations in (6.4.18), we obtain (6.4.21).

- 6.11** By differentiating (6.4.21) with respect to $H(e^{j\omega})$, derive the frequency response function $H_o(e^{j\omega})$ of the optimum filter in terms of $R_{yx}(e^{j\omega})$ and $R_x(e^{j\omega})$.

Differentiating with respect to $H(e^{j\omega})$ or equivalently with respect to $H^*(e^{j\omega})$ [see Appendix B].

$$\frac{\partial P}{\partial H^*(e^{j\omega})} = 0$$

$$0 = \frac{\partial}{\partial H^*(e^{j\omega})} \left[r_y(0) - \frac{1}{2\pi} \int_{-\pi}^{\pi} [H(e^{j\omega})R_{yx}^*(e^{j\omega}) + H^*(e^{j\omega})R_{yx}(e^{j\omega}) - H(e^{j\omega})H^*(e^{j\omega})R_{yx}^*(e^{j\omega})]d\omega \right]$$

Which implies that

$$\frac{\partial}{\partial H^*(e^{j\omega})} [H(e^{j\omega})R_{yx}^*(e^{j\omega}) + H^*(e^{j\omega})R_{yx}(e^{j\omega}) - H(e^{j\omega})H^*(e^{j\omega})R_{yx}^*(e^{j\omega})] = 0$$

Using (B.13), we obtain

$$R_{yx}(e^{j\omega}) - H_o(e^{j\omega})R_x(e^{j\omega}) = 0$$

or

$$H_o(e^{j\omega}) = \frac{R_{yx}(e^{j\omega})}{R_x(e^{j\omega})}$$

- 6.12** A conjugate symmetric linear smoother is obtained from (6.5.12) when $M = 2L$ and $i = L$. If the process $x(n)$ is stationary, then, using $\bar{R}J = J\bar{R}^*$, show that $\bar{c} = J\bar{c}^*$.

Using (6.5.12)

$$\bar{R}\bar{c} = \begin{bmatrix} 0 \\ P_o^{(i)} \\ 0 \end{bmatrix}$$

Multiply both sides by the exchange matrix J

$$J\bar{R}\bar{c} = J \begin{bmatrix} 0 \\ P_o^{(i)} \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ P_o^{(i)} \\ 0 \end{bmatrix}$$

With the second equality due to the symmetry of the MMSE $P_o^{(i)}(n)$. Using $\bar{R}J = J\bar{R}^*$ the above equation can be written

$$J\bar{R}\bar{c} = R^*Jc = \begin{bmatrix} 0 \\ P_o^{(i)} \\ 0 \end{bmatrix}$$

Resulting in $J\bar{c} = \bar{c}^*$

- 6.13** Let \bar{Q} and $\bar{\Lambda}$ be the matrices from the eigen-decomposition of \bar{R} , that is, $\bar{R} = \bar{Q}\bar{\Lambda}\bar{Q}^H$

- (a) Substitute R into (6.5.20) and (6.5.27) to prove (6.5.43) and (6.5.44).

Starting with (6.5.20), and solving directly

$$\begin{aligned} \bar{R}(n) \begin{bmatrix} 1 \\ a_o(n) \end{bmatrix} &= \begin{bmatrix} P_o^f(n) \\ 0 \end{bmatrix} \\ \bar{Q}\bar{\Lambda}\bar{Q}^H \begin{bmatrix} 1 \\ a_o(n) \end{bmatrix} &= \begin{bmatrix} P_o^f(n) \\ 0 \end{bmatrix} \\ \begin{bmatrix} 1 \\ a_o(n) \end{bmatrix} &= \bar{Q}\bar{\Lambda}^{-1}\bar{Q}^H \begin{bmatrix} P_o^f(n) \\ 0 \end{bmatrix} = P_o^f(n)\bar{Q}\bar{\Lambda}^{-1} \begin{bmatrix} q_{1,1}^* \\ \vdots \\ q_{M-1,1}^* \end{bmatrix} = P_o^f(n) \sum_{i=1}^{M+1} \frac{1}{\bar{\lambda}_i} \bar{q}_i \bar{q}_{i,1}^* \end{aligned}$$

A similar proof for (6.5.44).

- (b) Generalize the above result for a j th-order linear signal estimator $c^{(j)}(n)$; that is, prove that

$$c^{(j)}(n) = P_o^{(j)} \sum_{i=1}^{M+1} \frac{1}{\bar{\lambda}_i} \bar{q}_i \bar{q}_{i,j}^*$$

Starting with (6.5.12), and solving directly

$$\begin{aligned} \bar{R}\bar{c}^{(j)}(n) &= \begin{bmatrix} 0 \\ P_o^{(i)} \\ 0 \end{bmatrix} \\ \bar{c}^{(j)}(n) &= \bar{Q}\bar{\Lambda}^{-1}\bar{Q}^H \begin{bmatrix} 0 \\ P_o^{(i)} \\ 0 \end{bmatrix} = P_o^f(n)\bar{Q}\bar{\Lambda}^{-1} \begin{bmatrix} q_{1,j}^* \\ \vdots \\ q_{M-1,j}^* \end{bmatrix} = P_o^{(j)} \sum_{i=1}^{M+1} \frac{1}{\bar{\lambda}_i} \bar{q}_i \bar{q}_{i,j}^* \end{aligned}$$

6.14 Let $\tilde{R}(n)$ be the inverse of the correlation matrix $\bar{R}(n)$ given in (6.5.11).

(a) Using (6.5.12), show that the diagonal elements of $\tilde{R}(n)$ are given by

$$\langle \tilde{R}(n) \rangle_{i,i} = \frac{1}{P^{(i)}(n)} \quad 1 \leq i \leq M+1$$

Given

$$\bar{R}(n) = E\{\bar{x}(n)\bar{x}^H(n)\} = \begin{bmatrix} R_{11}(n) & r_1(n) & R_{12}(n) \\ r_1^H(n) & P_x(n-1) & r_2^H(n) \\ R_{12}^H(n) & r_2(n) & R_{22}(n) \end{bmatrix}$$

Let $\tilde{R}(n) = \bar{R}^{-1}(n)$. Consider (6.5.12)

$$\bar{R}(n) \bar{c}_o^{(i)}(n) = \begin{bmatrix} 0 \\ P_o^{(i)}(n) \\ 0 \end{bmatrix}$$

and divide by $P_o^{(i)}(n)$ to obtain

$$\bar{R}(n) \frac{\bar{c}_o^{(i)}(n)}{P_o^{(i)}(n)} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

or

$$\frac{\bar{c}_o^{(i)}(n)}{P_o^{(i)}(n)} = \bar{R}(n) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \langle \tilde{R}(n) \rangle_{1,i} \\ \vdots \\ \langle \tilde{R}(n) \rangle_{i,i} \\ \vdots \\ \langle \tilde{R}(n) \rangle_{M+1,i} \end{bmatrix}$$

Since the i th element of $\bar{c}_o^{(i)}(n) = 1$, comparing the i th row of the above equation, we obtain

$$\langle \tilde{R}(n) \rangle_{i,i} = \frac{1}{P^{(i)}(n)} \quad 1 \leq i \leq M+1$$

(b) Again consider (6.5.12) above

$$\bar{c}_o^{(i)}(n) = \bar{R}(n) \begin{bmatrix} 0 \\ P_o^{(i)}(n) \\ 0 \end{bmatrix} = P_o^{(i)}(n) \begin{bmatrix} \langle \tilde{R}(n) \rangle_{1,i} \\ \vdots \\ \langle \tilde{R}(n) \rangle_{i,i} \\ \vdots \\ \langle \tilde{R}(n) \rangle_{M+1,i} \end{bmatrix} = \frac{\tilde{r}_i(n)}{\langle \tilde{R}(n) \rangle_{i,i}}$$

6.15 The first five samples of the autocorrelation sequence of a signal $x(n)$ are $r(0) = 1$, $r(1) = 0.8$, $r(2) = 0.6$, $r(3) = 0.4$, and $r(4) = 0.3$. Compute the FLP, the BLP, the optimum symmetric smoother, and the corresponding MMSE.

(a) Normal equations method: The matrix $\bar{\mathbf{R}}$ is

$$\bar{\mathbf{R}} = \begin{bmatrix} 1 & 0.8 & 0.6 & 0.4 & 0.3 \\ 0.8 & 1 & 0.8 & 0.6 & 0.4 \\ 0.6 & 0.8 & 1 & 0.8 & 0.6 \\ 0.4 & 0.6 & 0.8 & 1 & 0.8 \\ 0.3 & 0.4 & 0.6 & 0.8 & 1 \end{bmatrix}$$

Following the method outlined in Table 6.3, we obtain the following systems of equations:
FLB:

$$\begin{bmatrix} 1 & 0.8 & 0.6 & 0.4 \\ 0.8 & 1 & 0.8 & 0.6 \\ 0.6 & 0.8 & 1 & 0.8 \\ 0.4 & 0.6 & 0.8 & 1 \end{bmatrix} \begin{bmatrix} c_1^f \\ c_2^f \\ c_3^f \\ c_4^f \end{bmatrix} = - \begin{bmatrix} 0.8 \\ 0.6 \\ 0.4 \\ 0.3 \end{bmatrix} \Rightarrow \begin{bmatrix} c_1^f \\ c_2^f \\ c_3^f \\ c_4^f \end{bmatrix} = \begin{bmatrix} -0.89286 \\ 0 \\ 0.25 \\ -0.14286 \end{bmatrix}$$

$$P_o^f = r_x(0) + \mathbf{d}^{(1)\top} c_o^f = 1 + [0.8 \ 0.6 \ 0.4 \ 0.3] \begin{bmatrix} -0.89286 \\ 0 \\ 0.25 \\ -0.14286 \end{bmatrix} = 0.34285$$

BLP:

$$\begin{bmatrix} 1 & 0.8 & 0.6 & 0.4 \\ 0.8 & 1 & 0.8 & 0.6 \\ 0.6 & 0.8 & 1 & 0.8 \\ 0.4 & 0.6 & 0.8 & 1 \end{bmatrix} \begin{bmatrix} c_1^b \\ c_2^b \\ c_3^b \\ c_4^b \end{bmatrix} = - \begin{bmatrix} 0.3 \\ 0.4 \\ 0.6 \\ 0.8 \end{bmatrix} \Rightarrow \begin{bmatrix} c_1^b \\ c_2^b \\ c_3^b \\ c_4^b \end{bmatrix} = \begin{bmatrix} -0.14286 \\ 0.25 \\ 0 \\ -0.89286 \end{bmatrix}$$

$$P_o^b = r_x(0) + \mathbf{d}^{(5)\top} c_o^b = 1 + [0.3 \ 0.4 \ 0.6 \ 0.8] \begin{bmatrix} -0.14286 \\ 0.25 \\ 0 \\ -0.89286 \end{bmatrix} = 0.34285$$

SLS:

$$\begin{bmatrix} 1 & 0.8 & 0.4 & 0.3 \\ 0.8 & 1 & 0.6 & 0.4 \\ 0.4 & 0.6 & 1 & 0.8 \\ 0.3 & 0.4 & 0.8 & 1 \end{bmatrix} \begin{bmatrix} c_1^s \\ c_2^s \\ c_3^s \\ c_4^s \end{bmatrix} = - \begin{bmatrix} 0.6 \\ 0.8 \\ 0.8 \\ 0.6 \end{bmatrix} \Rightarrow \begin{bmatrix} c_1^s \\ c_2^s \\ c_3^s \\ c_4^s \end{bmatrix} = \begin{bmatrix} 0 \\ -0.5 \\ -0.5 \\ 0 \end{bmatrix}$$

$$P_o^s = r_x(0) + \mathbf{d}^{(3)\top} c_o^s = 1 + [0.6 \ 0.8 \ 0.8 \ 0.6] \begin{bmatrix} 0 \\ -0.5 \\ -0.5 \\ 0 \end{bmatrix} = 0.2$$

(b) Inverse of normal equations method: The inverse of matrix $\bar{\mathbf{R}}$ is

$$\bar{\mathbf{R}}^{-1} = \begin{bmatrix} 1 & 0.8 & 0.6 & 0.4 & 0.3 \\ 0.8 & 1 & 0.8 & 0.6 & 0.4 \\ 0.6 & 0.8 & 1 & 0.8 & 0.6 \\ 0.4 & 0.6 & 0.8 & 1 & 0.8 \\ 0.3 & 0.4 & 0.6 & 0.8 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 2.9167 & -2.6042 & 0 & .72917 & -.41667 \\ -2.6042 & 5.1823 & -2.5 & -.65104 & .72917 \\ 0 & -2.5 & 5.0 & -2.5 & 0 \\ .72917 & -.65104 & -2.5 & 5.1823 & -2.6042 \\ -.41667 & .72917 & 0 & -2.6042 & 2.9167 \end{bmatrix}$$

Using the Property 6.5.4, we obtain the following results:

FLB: Divide the first column by 2.9167 to obtain \mathbf{c}_o^f and take inverse of 2.9167 to obtain P_o^f .

$$\begin{bmatrix} c_1^f \\ c_2^f \\ c_3^f \\ c_4^f \end{bmatrix} = \frac{1}{2.9167} \begin{bmatrix} -2.6042 \\ 0 \\ .72917 \\ -.41667 \end{bmatrix} = \begin{bmatrix} -0.89286 \\ 0 \\ 0.25 \\ -0.14286 \end{bmatrix}, P_o^f = \frac{1}{2.9167} = 0.34285$$

BLP: Divide the last column by 2.9167 to obtain \mathbf{c}_o^b and take inverse of 2.9167 to obtain P_o^b .

$$\begin{bmatrix} c_1^b \\ c_2^b \\ c_3^b \\ c_4^b \end{bmatrix} = \frac{1}{2.9167} \begin{bmatrix} -.41667 \\ 0.72917 \\ 0 \\ -2.6042 \end{bmatrix} = \begin{bmatrix} -0.89286 \\ 0 \\ 0.25 \\ -0.14286 \end{bmatrix}, P_o^b = \frac{1}{2.9167} = 0.34285$$

SLS: Divide the middle column by 5 to obtain \mathbf{c}_o^s and take inverse of 5 to obtain P_o^s .

$$\begin{bmatrix} c_1^s \\ c_2^s \\ c_3^s \\ c_4^s \end{bmatrix} = \frac{1}{5} \begin{bmatrix} 0 \\ -2.5 \\ -2.5 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ -0.5 \\ -0.5 \\ 0 \end{bmatrix}, P_o^s = \frac{1}{5} = 0.2$$

- 6.16** For the symmetric, Toeplitz autocorrelation matrix $\mathbf{R} = \text{Toeplitz}\{r(0), r(1), r(2)\} = r(0) \times \text{Toeplitz}\{1, \rho_1, \rho_2\}$ with $\mathbf{R} = \mathbf{LDL}^H$ and $\mathbf{D} = \text{diag}\{\xi_1, \xi_2, \xi_3\}$, the following conditions are equivalent:

- \mathbf{R} is positive definite.
- $\xi_i > 0$ for $1 \leq i \leq 3$.
- $|k_i| < 1$ for $1 \leq i \leq 3$.

Determine the values of ρ_1 and ρ_2 for which \mathbf{R} is positive definite, and plot the corresponding area in the (ρ_1, ρ_2) plane.

- 6.17** Prove the first equation in (6.5.45) by rearranging the FLP normal equations in terms of the unknowns $P_o^f(n), a_1(n), \dots, a_M(n)$ and then solve for $P_o^f(n)$, using Cramer's rule. Repeat the procedure for the second equation

(a) Given

$$\bar{R}(n) \begin{bmatrix} 1 \\ a_o(n) \end{bmatrix} = \begin{bmatrix} P_o^f(n) \\ 0 \end{bmatrix}$$

and using Cramer's rule for the first element

$$1 = \frac{D_1}{\det \bar{R}(n)}$$

where,

$$D_1 = \det \begin{bmatrix} P_o^f(n) & r^{fH}(n) \\ 0 & R(n-1) \end{bmatrix} = P_o^f(n) \det R(n-1)$$

Hence,

$$P_o^f(n) = \frac{\det \bar{R}(n)}{\det R(n-1)}$$

(b) Given

$$\bar{R}(n) \begin{bmatrix} b_o(n) \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ P_o^b(n) \end{bmatrix}$$

Using Cramer's rule for the last element

$$1 = \frac{D_{M+1}}{\det \bar{R}(n)}$$

where,

$$D_{M+1} = \det \begin{bmatrix} R(n) & 0 \\ r^{bH}(n) & P_0^b(n) \end{bmatrix} = P_0^b(n) \det R(n)$$

Hence,

$$P_0^b(n) = \frac{\det \bar{R}(n)}{\det R(n)}$$

6.18 Consider the signal $x(n) = y(n) + v(n)$ where $y(n)$ is a useful random signal corrupted by noise $v(n)$. The processes $y(n)$ and $v(n)$ are uncorrelated with PSDs

$$R_y(e^{j\omega}) = \begin{cases} 1 & 0 \leq |\omega| \leq \frac{\pi}{2} \\ 0 & \frac{\pi}{2} < |\omega| \leq \pi \end{cases}$$

and

$$R_v(e^{j\omega}) = \begin{cases} 1 & \frac{\pi}{4} \leq |\omega| \leq \frac{\pi}{2} \\ 0 & 0 \leq |\omega| < \frac{\pi}{4} \text{ and } \frac{\pi}{2} < |\omega| \leq \pi \end{cases}$$

(a) Determine the optimum IIR filter and find MMSE

Using 6.6.9,

$$H_{nc}(z) = \frac{R_{yx}(z)}{R_x(z)}$$

Evaluating on the unit circle $z = e^{j\omega}$ results in the following

$$H_{nc}(e^{j\omega}) = \frac{R_{yx}(e^{j\omega})}{R_x(e^{j\omega})}$$

Using the fact that $y(n)$ and $v(n)$ are uncorrelated, the above can be re-written as

$$H_{nc}(e^{j\omega}) = \frac{R_y(e^{j\omega})}{R_y(e^{j\omega}) + R_v(e^{j\omega})} = \begin{cases} 1 & 0 \leq |\omega| \leq \frac{\pi}{4} \\ 1/2 & \frac{\pi}{4} < |\omega| \leq \frac{\pi}{2} \\ 0 & \frac{\pi}{2} < |\omega| \leq \pi \end{cases}$$

This can be broken into the sum of two filters as

$$H_{nc}(e^{j\omega}) = H_{nc1}(e^{j\omega}) + H_{nc2}(e^{j\omega})$$

where

$$H_{nc1}(e^{j\omega}) = \begin{cases} 1/2 & 0 \leq |\omega| \leq \frac{\pi}{2} \\ 0 & \frac{\pi}{2} < |\omega| \leq \pi \end{cases}$$

and

$$H_{nc_2}(e^{j\omega}) = \begin{cases} 1/2 & 0 \leq |\omega| \leq \frac{\pi}{4} \\ 0 & \frac{\pi}{4} < |\omega| \leq \pi \end{cases}$$

Taking the inverse Fourier transform on these two filters results in the optimum IIR filter

$$h_{nc}(n) = h_{nc_1} + h_{nc_2} = \frac{\sin \frac{\pi n}{2}}{2\pi n} + \frac{\sin \frac{\pi n}{4}}{2\pi n}$$

The MMSE is evaluated as

$$P_o = \frac{1}{2\pi} \int_{-\pi}^{\pi} [R_y(e^{j\omega}) - H(e^{j\omega})R_{yx}^*(e^{j\omega})] d\omega = \frac{1}{2\pi}(\pi) - \frac{1}{2\pi}(\pi - \pi/4) = 1/8$$

- (b) Determine the third-order optimum FIR filter and MMSE

The optimum filter can be found from

$$\sum_{k=0}^{M-1} h_o(k)r_x(n-k) = r_{yx}(n) \quad 0 \leq n \leq M-1$$

where from part(a) above

$$r_{yx}(n) = r_y(n) = \frac{\sin \frac{\pi n}{2}}{2\pi n}$$

and

$$r_x(n) = r_y(n) + r_y(n) = \frac{\sin \frac{\pi n}{2}}{2\pi n} + (\cos \frac{3\pi n}{8}) \frac{\sin \frac{\pi n}{8}}{2\pi n}$$

Solving directly

$$h_o = R_x^{-1}r_y = \begin{bmatrix} 2 & 0.3649 & -0.0796 \\ 0.3649 & 2 & 0.3649 \\ -0.0796 & 0.3649 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 0.3183 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.4877 \\ 0.0689 \\ 0.0068 \end{bmatrix}$$

The MMSE is evaluated as

$$P_o = P_y - \sum_{k=0}^{M-1} h_o(k)r_{yx}^*(k) = 1 - [1 \ 0.3183 \ 0] \begin{bmatrix} 0.4877 \\ 0.0689 \\ 0.0068 \end{bmatrix} = 0.4904$$

- (c) Determine the non-causal optimum FIR filter defined by

$$\hat{y}(n) = h(-1)x(n+1) + h(0)x(n) + h(1)x(n-1)$$

Using a simple substitution, the above equation can be re-written as

$$\hat{y}(m-1) = h(0)x(m) + h(1)x(m-1) + h(2)x(m-2)$$

resulting in an optimal filter equation of

$$\sum_{k=0}^2 h_o(k)r_x(m-k) = r_{yx}(m-1) \quad 0 \leq m \leq M-1$$

Due to the real values of $x(n)$ and $y(n)$, the autocorrelation is symmetric and $r_y(l) = r_y(-l)$. Therefore the optimal filter equation is

$$h_o = R_x^{-1}r_y = \begin{bmatrix} r_x(0) & r_x(1) & r_x(2) \\ r_x(1) & r_x(0) & r_x(1) \\ r_x(2) & r_x(1) & r_x(0) \end{bmatrix}^{-1} \begin{bmatrix} r_y(1) \\ r_y(0) \\ r_y(1) \end{bmatrix} = \begin{bmatrix} 0.0760 \\ 0.4723 \\ 0.0760 \end{bmatrix}$$

The MMSE is evaluated as

$$P_o = P_y - \sum_{k=0}^2 h_o(k)r_{yx}^*(k-1) = 1 - [0.3183 \ 1 \ 0.3183] \begin{bmatrix} 0.0760 \\ 0.4723 \\ 0.0760 \end{bmatrix} = 0.4793$$

6.19 Consider the ARMA(1,1) process $x(n) = 0.8x(n-1) + \omega(n) + 0.5\omega(n-1)$, where $\omega(n) \sim WGN(0, 1)$.

- (a) Determine the coefficients and the MMSE of (1) the one-step ahead FLP $\hat{x}(n) = a_1x(n-1) + a_2x(n-2)$ and (2) the two-step ahead FLP $\hat{x}(n+2) = a_1x(n-1) + a_2x(n-2)$.

Using (6.5.20)

$$\bar{R}(n) \begin{bmatrix} 1 \\ a_o(n) \end{bmatrix} = \begin{bmatrix} P_o^f(n) \\ 0 \end{bmatrix}$$

where

$$\bar{R}(n) = \begin{bmatrix} P_x(n) & r^{fH}(n) \\ r^f(n) & R(n-1) \end{bmatrix} = \left[\begin{array}{c} r_x(0) = 8.25 \\ \begin{bmatrix} r_x(1) \\ r_x(2) \end{bmatrix} = \begin{bmatrix} 7.1 \\ 5.68 \end{bmatrix} \\ \begin{bmatrix} r_x(1) & r_x(2) \end{bmatrix} = \begin{bmatrix} 7.1 & 5.68 \end{bmatrix} \end{array} \right]$$

Using the inverse of \bar{R} to find the filter coefficients and MMSE

$$\begin{bmatrix} 1 \\ a_o(n) \end{bmatrix} = \bar{R}(n)^{-1} \begin{bmatrix} P_o^f(n) \\ 0 \end{bmatrix}$$

Therefore, $a_o = [-1.0337 \ 0.2011]^T$ and $P_o = 2.0532$

Repeating the above step for the two-step ahead filter, where

$$\bar{R}(n) = \left[\begin{array}{c} r_x(0) = 8.25 \\ \begin{bmatrix} r_x(2) \\ r_x(3) \end{bmatrix} = \begin{bmatrix} 5.68 \\ 4.54 \end{bmatrix} \\ \begin{bmatrix} r_x(2) & r_x(3) \end{bmatrix} = \begin{bmatrix} 5.68 & 4.54 \end{bmatrix} \end{array} \right]$$

Resulting in $a_o = [-0.8269 \ 0.1609]^T$ and $P_o = 4.289$

- (b) Check if the obtained prediction error filters are minimum-phase, and explain your findings.
For the filters to be minimum phase, the zeros of the filter transfer function must fall within the unit circle.
For a second order filter, the magnitude of the zero vector is

$$\left| \frac{a_2}{a_1} \right| < 1$$

Clearly, for both filters above, the zero is within the unit circle and therefore minimum phase.

6.20 Consider a random signal $x(n) = s(n) + v(n)$, where $v(n) \sim WGN(0, 1)$ and $s(n)$ is the AR(1) process $s(n) = 0.9s(n-1) + w(n)$, where $w(n) \sim WGN(0, 0.64)$. The signals $s(n)$ and $v(n)$ are uncorrelated.

- (a) Determine and plot the autocorrelation $r_s(l)$ and the PSD $R_s(e^{j\omega})$ of $s(n)$.

Start with

$$w(n) = s(n) * h(n)$$

where,

$$h(n) = (0.9)^n u(n)$$

Hence

$$r_s(l) = h(l) * h(-l) * r_w(l) = 0.64[h(l) * h(-l)]$$

Using,

$$[a^l u(n)] * [a^{-l} u(n)] = \frac{a^{|l|}}{1 - a^2} \quad |a| < 1$$

and

$$r_s(l) = \frac{0.64}{0.19} (0.9)^{|l|} = \frac{64}{19} (0.9)^{|l|}$$

The PSD is

$$\begin{aligned} R_s(e^{j\omega}) &= \sum_{-\infty}^{\infty} r_s(l) e^{-j\omega l} = \frac{64}{19} \left\{ \sum_0^{\infty} (0.9)^l e^{-j\omega l} + \sum_{-1}^{-\infty} (0.9)^{-l} e^{-j\omega l} \right\} \\ &= \frac{64}{19} \left\{ \frac{1}{1 - 0.9e^{-j\omega}} + \frac{0.9e^{-j\omega}}{1 - 0.9e^{-j\omega}} \right\} \end{aligned}$$

Therefore,

$$R_s(e^{j\omega}) = \frac{0.64}{1.81 - 1.8 \cos \omega}$$

The plot is obtained using the Matlab script and is shown in Figure 6.20a.

```
% (a) Plot of the PSD R_s
Hf_1 = figure('units','SCRUN','position',SCRPOS,...%
    'paperunits',PAPUN,'paperposition',[0,0,4,2]);
set(Hf_1,'NumberTitle','off','Name','Pr0620b');

omg = [0:1:500]*pi/500;
Rs = (0.64)./(1.81-1.8*cos(omg));
plot(omg/pi,Rs,'g'); axis([0,1,0,70]);
xlabel('\omega / \pi','fontsize',label_fontsize);
ylabel('R_s(e^{j\omega})','fontsize',label_fontsize);
set(gca,'xtick',[0:0.2:1],'ytick',[0:10:70]);
title('PSD of s(n)','fontsize',title_fontsize);
```

- (b) Design a second-order optimum FIR filter to estimate $s(n)$ from $x(n)$. What is the MMSE?

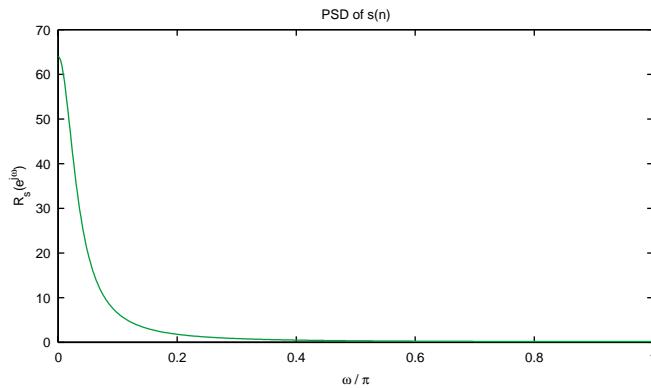


Figure 6.20a: plot of $R_s(e^{j\omega})$

```
% (b) Second-order (i.e., length M = 2) optimal FIR filter design
%% Given parameters
var_w = 0.64;
a = 0.9;
var_v = 1.0;

%% Computed Parameters
% Autocorrelation of s(n): rs(1) = (var_w/(1-a*a))*a^|1|
l = 0:100; rs = (var_w/(1-a*a))*a.^l;
Ps = rs(1);
% Signal x(n) = s(n) + v(n);
% Autocorrelation of x(n): rx(1) = rs(1)+var_v*delta(1)
rx = rs + var_v*[1,zeros(1,100)];
% Crosscorrelation between x(n) and s(n): rxs(1) = rs(1)
rxs = rs;

% Optimal filter
M = 2;
Rx = toeplitz(rx(1:M),rx(1:M));
dxs = rxs(1:M)';
co = Rx\dxs;
% Optimal error
Po = Ps - dxs'*co;
% Printout
Second-order (i.e., length M = 2) optimal FIR filter design
Optimal FIR filter coefficients:  0.5584,  0.3064
Optimal error = 0.5584
```

- (c) Design an optimum IIR filter to estimate $s(n)$ from $x(n)$. What is the MMSE?

Start with

$$H_o(e^{j\omega}) = \frac{R_{sx}(e^{j\omega})}{R_x(e^{j\omega})}$$

Now

$$\begin{aligned} R_x(e^{j\omega}) &= R_s(e^{j\omega}) + R_v(e^{j\omega}) = \frac{0.64}{1.81 - 1.8 \cos \omega} + 1 \\ &= \frac{2.45 - 1.8 \cos \omega}{1.81 - 1.8 \cos \omega} \end{aligned}$$

and

$$R_{sx}(e^{j\omega}) = R_s(e^{j\omega}) = \frac{0.64}{1.81 - 1.8 \cos \omega}$$

Thus

$$H_o(e^{j\omega}) = \frac{0.64}{2.45 - 1.8 \cos \omega}$$

or

$$h_o(n) = 0.3851(0.4377)^{|l|}$$

a non-causal IIR filter.

The optimum error is given by

$$\begin{aligned} P_o &= P_s - \sum_{-\infty}^{\infty} h_o(k) r_{sx}^*(l) = P_s - \sum_{-\infty}^{\infty} h_o(k) r_s(l) \\ &= \frac{0.64}{0.19} - \frac{0.64}{0.19} (0.3851) \sum_{-\infty}^{\infty} [(0.9)(0.4377)]^{|l|} = 0.5136 \end{aligned}$$

6.21 A useful signal $s(n)$ with PSD $R_s(z) = [(1 - 0.9z^{-1})(1 - 0.9z)]^{-1}$ is corrupted by additive uncorrelated noise $v(n) \sim \text{WN}(0, \sigma_v^2)$. Choose $\sigma_v^2 = 1$.

- (a) The resulting signal $x(n) = s(n) + v(n)$ is passed through a causal filter with system function $H(z) = (1 - 0.8z^{-1})^{-1}$. Determine (1) the SNR at the input, (2) the SNR at the output, and (3) the processing gain, that is, the improvement in SNR.

1. the SNR at the input: The autocorrelation of the input is

$$r_s(l) = \mathcal{Z}^{-1}[R_s(z)] = \frac{1}{0.19} (0.9)^{|l|} = 5.2632 (0.9)^{|l|}$$

Hence the input power is $r_s(0) = 5.2632$. The noise power is $r_v(0) = \sigma_v^2 = 1$. Hence

$$\text{SNR}_I = 5.2632$$

2. the SNR at the Output: Let the signal component at the output of the filter due to $s(n)$ be $y_s(n)$ and that due to noise be $y_v(n)$. Then

$$\begin{aligned} R_{y_s}(z) &= H(z)H(z^{-1})R_s(z) = \frac{1}{(1 - 0.8z)(1 - 0.8z^{-1})(1 - 0.9z)(1 - 0.9z^{-1})} \\ &= -\frac{79.365z}{z - .8} + \frac{169.17z}{z - .9} - \frac{169.17z}{z - 1.1111} + \frac{79.365z}{z - 1.25} \end{aligned}$$

or

$$r_{y_s}(l) = 169.17 (0.9)^{|l|} - 79.365 (0.8)^{|l|}$$

Thus the signal power at the output of the filter is $169.17 - 79.365 = 89.805$. Similarly,

$$\begin{aligned} R_{y_v}(z) &= H(z)H(z^{-1})R_v(z) = \frac{1}{(1-0.8z)(1-0.8z^{-1})} \\ &= \frac{2.7778z}{z-.8} - \frac{2.7778z}{z-1.25} \end{aligned}$$

or

$$r_{y_v}(l) = 2.7778(0.8)^{|l|}$$

Thus the noise power at the output of the filter is 2.7778. Hence $\text{SNR}_O = \frac{89.805}{2.7778} = 32.339$.

3. the processing gain:

$$\text{PG} = \frac{\text{SNR}_O}{\text{SNR}_I} = \frac{32.339}{5.263} = 6.1446$$

(b) Determine the causal optimum filter and compare its performance with that of the filter in (a).

First

$$\begin{aligned} R_x(z) &= R_s(z) + R_v(z) = \frac{1}{(1-0.9z)(1-0.9z^{-1})} + 1 \\ &= \frac{2.81 - 0.9z - 0.9z^{-1}}{(1-0.9z)(1-0.9z^{-1})} = 0.55198 \frac{(1-0.36233z)(1-0.36233z^{-1})}{(1-0.9z)(1-0.9z^{-1})} \end{aligned}$$

Hence $\sigma_x^2 = 0.55198$ and

$$H_x(z) = \frac{1-0.36233z^{-1}}{1-0.9z^{-1}}$$

Also

$$\begin{aligned} R_{sw}(z) &= \frac{R_{sx}(z)}{H_x(z)} = \frac{R_s(z)}{H_x(z^{-1})} = \frac{1}{(1-0.9z)(1-0.9z^{-1})} \frac{(1-0.9z)}{(1-0.36233z)} \\ &= \frac{1}{(1-0.9z^{-1})(1-0.36233z)} = \frac{1.4839}{1-.9z^{-1}} - \frac{1.4839}{1-0.36233z} \end{aligned}$$

Hence

$$\begin{aligned} H_c(z) &= \frac{[R_{sw}(z)]_+}{\sigma_x^2 H_x(z)} = \frac{1}{0.55198} \frac{1.4839}{1-.9z^{-1}} \frac{1-0.9z^{-1}}{1-0.36233z^{-1}} \\ &= \frac{2.6883}{1-0.36233z^{-1}} \end{aligned}$$

1. the SNR at the input: This is same as before, that is, $\text{SNR}_I = 5.2632$

2. the SNR at the Output: Let the signal component at the output of the filter due to $s(n)$ be $y_s(n)$ and that due to noise be $y_v(n)$. Then

$$\begin{aligned} R_{y_s}(z) &= H_c(z)H_c(z^{-1})R_s(z) = \frac{(2.6883)(2.6883)}{(1-0.36233z^{-1})(1-0.36233z)(1-0.9z)(1-0.9z^{-1})} \\ &= -\frac{8.3189z}{z-.36233} + \frac{94.478z}{z-.9} - \frac{94.478z}{z-1.1111} + \frac{8.3189z}{z-2.7599} \end{aligned}$$

or

$$r_{ys}(l) = 94.478 (0.9)^{|l|} - 8.3189 (0.36233)^{|l|}$$

Thus the signal power at the output of the filter is $94.478 - 8.3189 = 86.159$. Similarly,

$$\begin{aligned} R_{yv}(z) &= H_c(z)H_c(z^{-1})R_v(z) = \frac{(2.6883)(2.6883)}{(1 - 0.36233z^{-1})(1 - 0.36233z)} \\ &= \frac{8.3191}{z - .36233} - \frac{8.3191}{z - 2.7599} \end{aligned}$$

or

$$r_{yv}(l) = 8.3191 (0.36233)^{|l|}$$

Thus the noise power at the output of the filter is 8.3191. Hence $\text{SNR}_O = \frac{86.159}{8.3191} = 10.357$.

3. the processing gain:

$$\text{PG} = \frac{\text{SNR}_O}{\text{SNR}_I} = \frac{10.357}{5.263} = 1.9679$$

- 6.22** A useful signal $s(n)$ with PSD $R_s(z) = 0.36[(1 - 0.8z^{-1})(1 - 0.8z^{-1})]^{-1}$ is corrupted by additive uncorrelated noise $v(n) \sim \text{WN}(0, 1)$. Determine the optimum noncausal and causal IIR filters, and compare their performance by examining the MMSE and their magnitude response.

The optimum noncausal filter is

$$\begin{aligned} H_{nc}(z) &= \frac{R_{sx}(z)}{R_x(z)} = \frac{R_s(z)}{R_s(z) + R_v(z)} = \frac{0.36}{0.36 + (1 - 0.8z^{-1})(1 - 0.8z^{-1})} \\ &= \frac{0.225}{(1 - 0.5z^{-1})(1 - 0.5z^{-1})} \end{aligned}$$

Therefore the optimum noncausal filter is $h_{nc}(l) = (0.225)(0.5)^{|l|}$. The resulting MMSE is

$$\begin{aligned} P_o &= r_s(0) - \sum_k h_{nc}(k)r_{sx}(k) = 1 - (0.225) \sum_k (0.5)^{|k|} (0.9)^{|k|} = 1 - (0.225)(3.333) \\ &= 0.25 \end{aligned}$$

The optimum causal filter is

$$H_c(z) = \frac{1}{\sigma_x^2 H_x(z)} \left[\frac{R_{yx}(z)}{H_x(1/z)} \right]_+$$

where

$$\begin{aligned} \left[\frac{R_{yx}(z)}{H_x(1/z)} \right]_+ &= \left[\frac{R_{sx}(z)}{H_x(1/z)} \right]_+ = \left[\frac{0.36}{(1 - 0.8z^{-1})(1 - 0.8z)} \left(\frac{1 - 0.8z}{1 - 0.5z} \right) \right]_+ \\ &= \left[\frac{-0.6}{1 - 2z^{-1}} + \frac{0.6}{1 - 0.8z^{-1}} \right]_+ = \frac{0.6}{1 - 0.8z^{-1}} \end{aligned}$$

Also

$$R_x(z) = R_s(z) + R_v(z) = \sigma_x^2 H_x(z)H_x(z^{-1}) = (1.6) \left(\frac{1 - 0.5z^{-1}}{1 - 0.8z^{-1}} \right) \left(\frac{1 - 0.5z}{1 - 0.8z} \right)$$

Therefore,

$$H_x(z) = \frac{1 - 0.5z^{-1}}{1 - 0.8z^{-1}}$$

and

$$H_c(z) = \left(\frac{1 - 0.5z^{-1}}{1.6(1 - 0.8z^{-1})} \right) \left(\frac{0.6}{1 - 0.8z^{-1}} \right) = \frac{0.375}{1 - 0.5z^{-1}}$$

The optimum causal filter is

$$h_c(l) = 0.375(0.5)^l u(l)$$

The resulting MMSE is

$$\begin{aligned} P_o &= r_s(0) - \sum_k h_c(k) r_{sx}(k) = 1 - (0.375) \sum_{k=0}^{\infty} (0.5)^k (0.9)^k = 1 - (0.375)(1.667) \\ &= 0.375 \end{aligned}$$

6.23 Consider a process with PSD $R_x(z) = \sigma^2 H_x(z) H_x(z^{-1})$. Determine the D -step ahead linear predictor, and show that the MMSE is given by $P^{(D)} = \sigma^2 \sum_{n=0}^{D-1} h_{lp}^2(n)$. Check your results by using the PSD $R_x(z) = (1 - a^2)[(1 - az^{-1})(1 - az)]^{-1}$.

Consider the desired response

$$y(n) = x(n + D)$$

then

$$e^f(n + D) = x(n + 1) - \sum_{k=0}^{\infty} h_{lp}^{[D]}(k) x(n - k)$$

where $h_{lp}^{[D]}(n)$ is the impulse response of the D -step ahead predictor. Thus

$$r_{yx}(l) = E\{y(n)x^*(n - l)\} = E\{x(n + D)x^*(n - l)\} = r_x(l + D)$$

or

$$R_{yx}(z) = z^D R_x(z)$$

Thus,

$$H_{lp}^{[D]}(z) = \frac{1}{\sigma_x^2 H_x(z)} \left[\frac{z^D \sigma_x^2 H_x(z) H_x(1/z)}{H_x(1/z)} \right]_+ = \frac{[z^D H_x(z)]_+}{H_x(z)}$$

where

$$[z^D H_x(z)]_+ = h(D) + h(D + 1)z^{-1} + h(D + 2)z^{-2} + \dots$$

Consider,

$$z^D H_x(z) = z^D h(0) + h(1)z^{D-1} + h(2)z^{D-2} + \dots + h(D)z^0 + h(D + 1)z^{-1} + \dots \quad h(0) = 1$$

Hence,

$$\begin{aligned}[z^D H_x(z)]_+ &= z^D H_x(z) - [z^D h + h(1)z^{D-1} + h(2)z^{D-2} + \cdots + h(D-1)z] \\ &= z^D H_x(z) - z^D [1 + h(1)z^{-1} + \cdots + h(D-1)z^{1-D}]\end{aligned}$$

or

$$\frac{[z^D H_x(z)]_+}{H_x(z)} = z^D - z^D \left[\frac{\sum_{k=0}^{D-1} h(k)z^{-k}}{H_x(z)} \right] = z^D \left[1 - \frac{\sum_{k=0}^{D-1} h(k)z^{-1}}{H_x(z)} \right]$$

Substituting in $P_o^{[D]}$

$$\begin{aligned}P_o^{[D]} &= \frac{1}{2\pi j} \oint [R_x(z) - z^D \left\{ 1 - \frac{\sum_{k=0}^{D-1} h(k)z^{-1}}{H_x(z)} \right\} z^{-D} R_x(z)] z^{-1} dz \\ &= \frac{1}{2\pi j} \oint \left[R_x(z) \frac{\sum_{k=0}^{D-1} h(k)z^{-1}}{H_x(z)} \right] z^{-1} dz = \frac{1}{2\pi j} \oint \sigma_x^2 H_x(1/z) \sum_{k=0}^{D-1} h(k)z^{-1} z^{-1} dz \\ &= \frac{\sigma_x^2}{2\pi j} \oint H_x(1/z) \sum_{k=0}^{D-1} h(k)z^{-1} z^{-1} dz\end{aligned}$$

The inverse z-transform is given by

$$h(n) = \frac{1}{2\pi j} \oint H(z) z^{n-1} dz$$

If we substitute $z \rightarrow z^{-1} = 1/z$ then

$$H(z) \rightarrow H(z^{-1}) \quad \text{and} \quad z^{n-1} \rightarrow z^{-n+1}$$

Also $p = z^{-1}$ so $dp = 1z^{-2}dz$ or $dz = -p^{-2}dp$ and

$$h(n) = \frac{1}{2\pi j} \oint H(p^{-1}) p^{-n+1} p^{-2} dp = \frac{1}{2\pi j} \oint H(p^{-1}) p^{-n-1} dp$$

Hence

$$P_o^{[D]} = \sum_{k=0}^{D-1} h^2(k)$$

To check, consider

$$R_x(z) = \frac{1 - a^2}{(1 - az^{-1})(1 - az)}$$

with

$$\sigma_x^2 = (1 - a^2), \quad H_x(z) = \frac{1}{1 - az^{-1}}, \quad H_x(z^{-1}) = \frac{1}{1 - az}, \quad h_x(k) = a^k u(k)$$

Hence

$$[z^D H_x(z)]_+ = a^D + a^{D+1}z^{-1} + \cdots = a^D [1 + az^{-1} + \cdots] = a^D H_x(z)$$

and

$$H_0(z) = a^D \Rightarrow h_0(k) = a^D \delta(k)$$

The MMSE is

$$\begin{aligned} P_o &= r_y(0) - \sum_k h_0(k)r_{yx}^*(k) = r_x(0) - \sum_k a^D \delta(k)r_{yx}^*(k) \\ &= r_x(0) - a^D r_{yx}^*(0) = \sigma_x^2 \sum_k h_x(k)^2 - \sigma_x^2 \sum_D h_x(k)^2 \\ &= \sigma_x^2 \sum_{k=0}^{D-1} h_x(k)^2 \end{aligned}$$

6.24 Let $x(n) = s(n) + v(n)$ with $R_v(z) = 1$, $R_{sv}(z) = 0$, and

$$R_s(z) = \frac{0.75}{(1 - 0.5z^{-1})(1 - 0.5z)}$$

Determine the optimum filters for the estimation of $s(n)$ and $s(n-2)$ from $\{x(k)\}_{-\infty}^n$ and the corresponding MMSEs.

Start with

$$H_c(z) = \frac{1}{\sigma_x^2 H_x(z)} \left[\frac{R_{sx}(z)}{H_x(1/z)} \right] = \frac{1}{\sigma_x^2 H_x(z)} \left[\frac{R_s(z)}{H_x(1/z)} \right]$$

Compute the autocorrelation $R_x(z)$

$$\begin{aligned} R_x(z) &= R_s(z) + R_v(z) = \frac{0.75}{(1 - 0.5z^{-1})(1 - 0.5z)} + 1 \\ &= \sigma_x^2 H_x(z) H_x(z^{-1}) = (1.866) \left(\frac{1 - 0.2679z^{-1}}{1 - 0.5z^{-1}} \right) \left(\frac{1 - 0.2679z}{1 - 0.5z} \right) \end{aligned}$$

Therefore,

$$H_x(z) = \frac{1 - 0.2679z^{-1}}{1 - 0.5z^{-1}}$$

Substituting into the above equation for $H_c(z)$

$$\begin{aligned} H_c(z) &= \left(\frac{1}{1.866} \right) \left(\frac{1 - 0.5z^{-1}}{1 - 0.2679z^{-1}} \right) \left(\frac{0.8659}{1 - 0.5z^{-1}} \right) \\ &= 0.464 \left(\frac{1}{1 - 0.2679z^{-1}} \right) \end{aligned}$$

Therefore, $h_c(l) = 0.464(0.2679)^l u(l)$, and is shown in Figure 6.24. The resulting MMSE is

$$P_o = r_s(0) - \sum_k h_c(k)r_{sx}(k) = 1 - \sum_{k=0}^{\infty} 0.464(0.2679)^k (0.70)(0.5)^k = 0.5982$$

For $s(n-2)$,

$$\begin{aligned} R'_{yw}(z) &= z^D R_{yw}(z) = z^{-2} \frac{R_{yx}(z)}{H(1/z)} = \frac{0.75z^{-3}}{-0.2679 + 1.134z^{-1} - 0.5z^{-2}} \\ [R'_{yw}(z)]_+ &= \left[\frac{0.062 + 0.201z^{-1} + 0.75z^{-2}}{1 - 0.5z^{-1}} \right] \end{aligned}$$

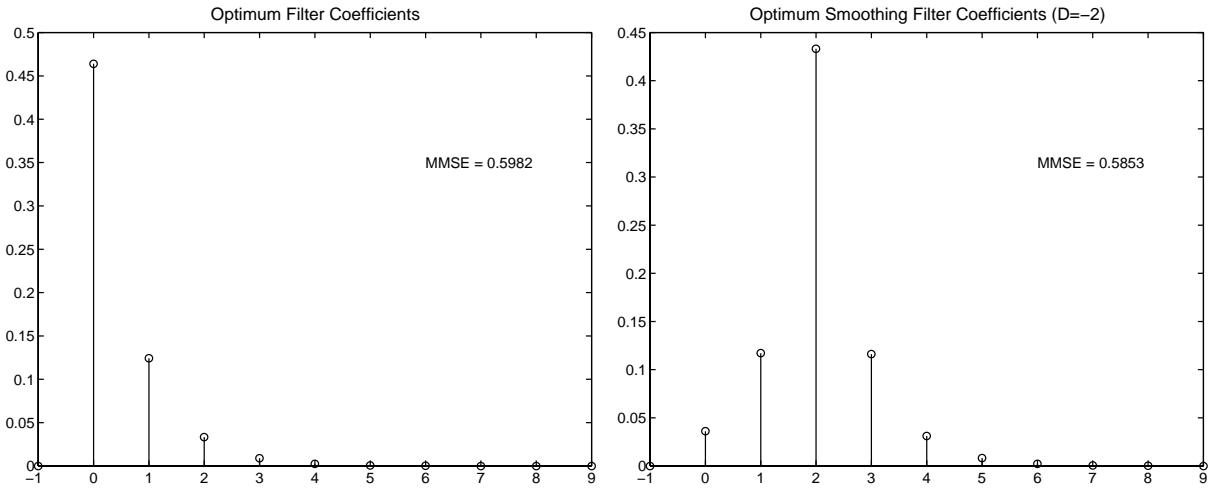


Figure 6.24: Optimum Estimation Filters

Plugging this into the above equation for $H_c(z)$

$$H_c(z) = (1.866) \left(\frac{1 - 0.2679z^{-1}}{1 - 0.5z^{-1}} \right) \left[\frac{0.062 + 0.201z^{-1} + 0.75z^{-2}}{1 - 0.5z^{-1}} \right]$$

and the impulse response is

$$h_c(l) = -1.5\delta(l-1) - 6\delta(l) + 6.036(0.2679)^l u(l)$$

The resulting MMSE is

$$P_o = r_s(0) - \sum_k h_c(k)r_{sx}(k) = 0.75 - \sum_{k=0}^{\infty} h_c(k)r_{sx}(k) = 0.5853$$

6.25 For the random signal with PSD

$$R_x(z) = \left(\frac{1 - 0.2z^{-1}}{1 - 0.9z^{-1}} \right) \left(\frac{1 - 0.2z}{1 - 0.9z} \right)$$

determine the optimum two-step ahead linear predictor and the corresponding MMSE.

First find the autocorrelation

$$\begin{aligned} R_x(z) &= \frac{1.04 - 0.2z^{-1} - 0.2z}{1.81 - 0.9z^{-1} - 0.9z} = \frac{-3.3567}{1 - 1.11z^{-1}} + \frac{3.3567}{1 - 0.9z^{-1}} + 0.222 \\ r_x(l) &= 3.3567(0.9)^{|l|} + \delta(l)(0.222) \end{aligned}$$

The two-step ahead linear predictor can be found directly using (6.5.18)

$$R_x(n-1)a_o = -r_x^f(n) \Rightarrow a_o = \{0.6301, 0.1259, 0.0252, 0.0050, 0.0010, 0.0002, 0 \dots\}$$

The corresponding MMSE is

$$P_o^f(n) = P_x(n) + r^{fH}(n)a_o(n) = 1.4897$$

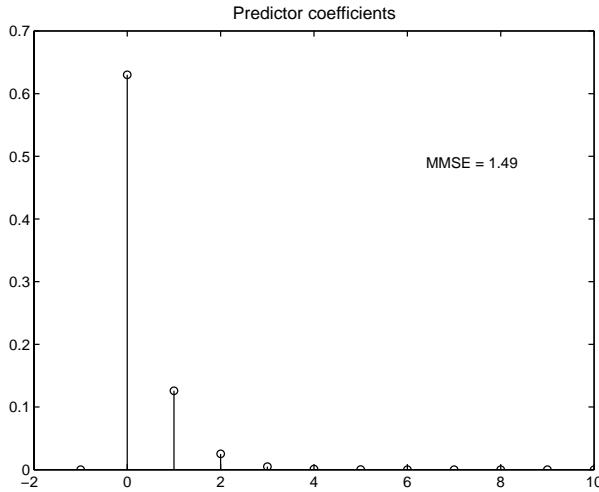


Figure 6.25: Two-Step Ahead Linear Predictor

6.26 Repeat Problem 6.25 for

$$R_x(z) = \frac{1}{(1 - 0.2z^{-1})(1 - 0.9z^{-1})(1 - 0.2z)(1 - 0.9z)}$$

First find the autocorrelation

$$\begin{aligned} R_x(z) &= \frac{1}{(1.04 - 0.2z^{-1} - 0.2z)(1.81 - 0.9z^{-1} - 0.9z)} \\ &= \frac{z^{-2}}{0.18 - 1.298z^{-1} + 2.242z^{-2} - 1.298z^{-3} + 0.18z^{-4}} \\ &= \frac{0.0653}{1 - 5z^{-1}} + \frac{-1.4854}{1 - 1.11z^{-1}} \frac{1.4854}{1 - 0.9z^{-1}} + \frac{-0.0653}{1 - 0.2z^{-1}} \\ r_x(l) &= 1.4854(0.9)^{|l|} - 0.0653(0.2)^{|l|} \end{aligned}$$

$$R_x(n-1)a_o = -r_x^f(n)$$

Figure 6.27 shows the filter coefficients

The corresponding MMSE is

$$P_o^f(n) = P_x(n) + r^{fH}(n)a_o(n) = 0.7403$$

6.27 Let $x(n) = s(n) + v(n)$ with $v(n) \sim WN(0, 1)$ and $s(n) = 0.6s(n-1) + w(n)$, where $w(n) \sim WN(0, 0.82)$. The process $s(n)$ and $v(n)$ are uncorrelated.

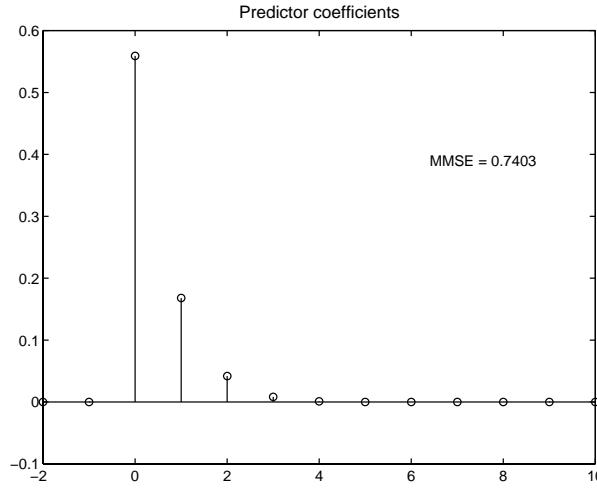
Determine the optimum filters for the estimation of $s(n)$, $s(n+2)$, and $s(n-2)$ from $\{x(k)\}_{-\infty}^n$ and the corresponding MMSEs.

The optimum filter equation is

$$C_o = R_x^{-1}d$$

where $R_x = R_s + R_v$. The autocorrelation of the signal is symmetric, and each lag is found using

$$r_s(l) = r_s(0)(0.6)^{|l|}$$

**Figure 6.26: Two-Step Ahead Linear Predictor**

where

$$r_s(0) = \frac{\sigma_w^2}{1 - \rho^2} = \frac{\sigma_w^2}{1 - (0.6)^2}$$

and $r_v(l) = \sigma_v^2 \delta(l)$. The same optimum filter equation is used for the three cases: filtering $s(n)$, estimation $s(n+2)$ or smoothing $s(n-2)$. The only change is the value of the cross-correlation vector between the desired response and the input data vector. The value for this vector is shown below for the three cases Filtering

$$s(n) : d(l) = E\{x(n)s^*(n+l)\} = r_s(l) = \begin{bmatrix} 1 \\ (0.6) \\ (0.6)^2 \\ \vdots \end{bmatrix}$$

Estimation

$$s(n+2) : d(l) = E\{x(n)s^*(n+l)\} = r_s(l+2) = \begin{bmatrix} (0.6)^2 \\ (0.6)^3 \\ (0.6)^4 \\ \vdots \end{bmatrix}$$

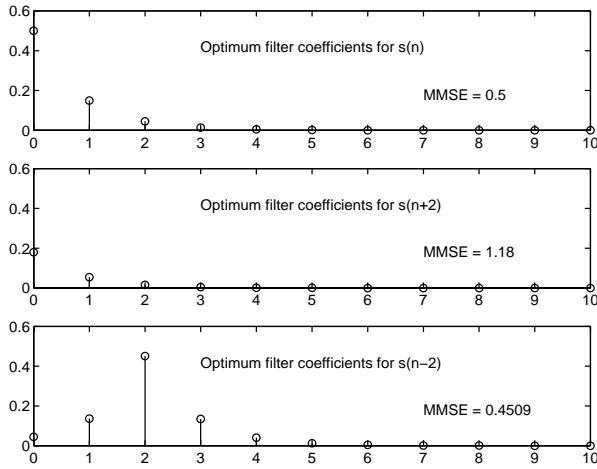
Smoothing

$$s(n-2) : d(l) = E\{x(n)s^*(n+l)\} = r_s(l-2) = \begin{bmatrix} (0.6)^2 \\ (0.6) \\ 1 \\ (0.6) \\ (0.6)^2 \\ \vdots \end{bmatrix}$$

The MMSE is found directly using the filter coefficients and the appropriate cross-correlation vector

$$P_o = P_y - d^H C_o = r_s(0) - d^H C_o$$

The filter coefficients are found, using MATLAB, and shown in Figure 6.27, along with the appropriate MMSE values.

**Figure 6.27:** Optimum Filter Coefficients

6.28 A random sequence $s(n)$ with PSD $R_s(z) = \frac{1}{(1-0.5z^{-1})(1-0.5z)}$ is corrupted by noise sequence $v(n)$ with PSD $R_v(z) = 5$ to obtain the observed sequence $x(n) = s(n) + v(n)$. Also $R_{sv}(z) = 0$.

Optimal causal filter to estimate $s(n)$ from $\{x(n)\}_{-\infty}^n$: First, $r_s(l) = z^{-l} \left[\frac{1}{(1-0.5z^{-1})(1-0.5z)} \right] = \frac{4}{3} (0.5)^{|l|}$. Now

$$R_x(z) = \frac{1}{(1-0.5z^{-1})(1-0.5z)} + 5 = 6.25 \frac{(1-0.4z^{-1})(1-0.4z)}{(1-0.5z^{-1})(1-0.5z)}$$

and $R_{sx}(z) = R_s(z)$. Thus after spectral factorization

$$\sigma_x^2 = 6.25 \text{ and } H_x(z) = \frac{(1-0.4z^{-1})}{(1-0.5z^{-1})}, |z| > 0.5$$

Hence

$$\begin{aligned} R_{sw}(z) &= \frac{R_{sx}(z)}{H_x(z^{-1})} = \frac{1}{(1-0.5z^{-1})(1-0.5z)(1-0.4z)} = \frac{1}{(1-0.5z^{-1})(1-0.4z)} \\ &= \frac{1.25}{1-0.5z^{-1}} - \frac{1.25}{1-2.5z^{-1}} \end{aligned}$$

or $[R_{sw}(z)]_+ = \frac{1.25}{1-0.5z^{-1}}$. Finally

$$H_c(z) = \frac{[R_{sw}(z)]_+}{\sigma_x^2 H_x(z)} = \frac{1}{6.25} \frac{1.25}{1-0.5z^{-1}} \frac{(1-0.5z^{-1})}{(1-0.4z^{-1})} = \frac{0.2}{(1-0.4z^{-1})}, |z| > 0.4$$

and $h_c(n) = 0.2 (0.4)^n u(n)$. The MMSE is given by

$$P_c = r_s(0) - \sum_{k=0}^{\infty} h_c(k) r_{sx}(k) = \frac{4}{3} - \left(\frac{1}{5}\right) \left(\frac{4}{3}\right) \sum_{k=0}^{\infty} (0.4)^k (0.5)^k = 1$$

Optimal prediction filter to estimate $s(n+2)$ from $\{x(n)\}_{-\infty}^n$: The whitening part of the causal filter does not depend on $s(n+2)$ and is given by $H_x(z)$. The coloring part depends on $s(n+2)$ and is given by $z^2 R_{sw}(z)$. Thus

$$H_o^{[2]}(z) = \frac{[R_{sw}(z)]_+}{\sigma_x^2 H_x(z)} (0.5)^2 = \frac{0.2 (0.5)^2}{(1-0.4z^{-1})} = \frac{0.05}{(1-0.4z^{-1})}, |z| > 0.4$$

and $h_o^{[2]}(n) = 0.05 (0.4)^n u(n)$. The MMSE is given by

$$\begin{aligned} P_o^{[2]} &= r_s(0) - \sum_{k=0}^{\infty} h_o^{[2]}(k) r_{sx}(k+2) = \frac{4}{3} - (0.05) \left(\frac{4}{3}\right) (0.5)^2 \sum_{k=0}^{\infty} (0.4)^k (0.5)^k \\ &= 1.3132 > P_o^{[0]} = P_c = 1 \end{aligned}$$

Optimal smoothing filter to estimate $s(n-2)$ from $\{x(n)\}_{-\infty}^n$: Once again the whitening part of the causal filter does not depend on $s(n+2)$ and is given by $H_x(z)$. The coloring part is obtained from

$$[r_{sw}^{[-2]}(l)]_+ \triangleq r_{sw}(l-2)u(l)$$

From $R_{sw}(z)$ above, we have $r_{sw}(l) = 1.25 (0.5)^l u(l) + 1.25 (2.5)^l u(-l-1)$. Hence

$$\begin{aligned} [r_{sw}^{[-2]}(l)]_+ &= r_{sw}(l-2)u(l) = 1.25 \{(0.5)^{l-2} u(l-2) + (2.5)^{l-2} u(-l+2-1)\} u(l) \\ &= 1.25 (0.5)^{l-2} u(l-2) + 1.25 \{0.4\delta(l) + \delta(l-1)\} \end{aligned}$$

or

$$\begin{aligned} [R_{sw}^{[-2]}(z)]_+ &= Z \{[r_{sw}^{[-2]}(l)]_+\} = \frac{5}{4} \frac{z^{-2}}{1 - 0.5z^{-1}} + \frac{1}{2} + \frac{5}{4} z^{-1} \\ &= \frac{1}{2} \frac{(1 + 2z^{-1} + 1.25z^{-2})}{(1 - 0.5z^{-1})} \end{aligned}$$

Finally

$$\begin{aligned} H_o^{[-2]}(z) &= \frac{[R_{sw}^{[-2]}(z)]_+}{\sigma_x^2 H_x(z)} = \frac{1}{6.25} \frac{(1 - 0.5z^{-1})}{(1 - 0.4z^{-1})} \frac{1}{2} \frac{(1 + 2z^{-1} + 1.25z^{-2})}{(1 - 0.5z^{-1})} \\ &= 0.08 \frac{(1 + 2z^{-1} + 1.25z^{-2})}{(1 - 0.4z^{-1})}, |z| > 0.4 \\ &= 0.08 \frac{(1 - 0.4z^{-1}) + 2.4z^{-1}(1 - 0.4z^{-1}) + 2.21z^{-2}}{(1 - 0.4z^{-1})}, |z| > 0.4 \\ &= 0.08 + 0.192z^{-1} + 0.1768z^{-2} \frac{1}{(1 - 0.4z^{-1})}, |z| > 0.4 \end{aligned}$$

or

$$h_o^{[-2]}(n) = 0.08\delta(n) + 0.192\delta(n-1) + 0.1768 (0.4)^{n-2} u(n-2)$$

To compute the MMSE, we first express $r_{sx}(l-2)u(l)$ in the form

$$r_{sx}(l-2)u(l) = r_s(l-2)u(l) = \frac{1}{3}\delta(l) + \frac{2}{3}\delta(l-1) + \frac{4}{3}(0.5)^{l-2}u(l-2)$$

Then

$$\begin{aligned} P_o^{[-2]} &= r_s(0) - \sum_{k=0}^{\infty} h_o^{[-2]}(k) r_{sx}(k-2) \\ &= \frac{4}{3} - (0.08) \left(\frac{1}{3}\right) - (0.192) \left(\frac{2}{3}\right) - (0.1768) \left(\frac{4}{3}\right) \sum_{k=0}^{\infty} (0.4)^k (0.5)^k \\ &= 0.884 < P_o^{[0]} = P_c = 1 \end{aligned}$$

6.29 Consider the random sequence $x(n)$ generated in Example 6.5.2.

$$x(n) = w(n) + \frac{1}{2}w(n-1)$$

where $w(n)$ is $\text{WN}(0, 1)$. Generate K sample functions $\{w_k(n)\}_{n=0}^N, k = 1, \dots, K$ of $w(n)$, in order to generate K sample functions $\{x_k(n)\}_{n=0}^N, k = 1, \dots, K$ of $x(n)$.

```
% Generate K samples functions of x(n)
K = 1000; N = 100;
x = zeros(N,K);
w = randn(N,K);
for k = 1:K; x(:,k) = filter([1,0.5],1,w(:,k)); end;
```

- (a) Use the second-order FLP a_k to obtain predictions $\{\hat{x}_k^f(n)\}_{n=2}^N$ of $x_k(n)$, for $k = 1, \dots, K$. Then determine the average error

$$\hat{P}^f = \frac{1}{N-1} \sum_{n=2}^N |x_k(n) - \hat{x}_k^f(n)|^2 \quad k = 1, \dots, K$$

and plot it as a function of k . Compare it with P_o^f .

```
% (a) Second-order forward linear predictor
M = 2; Rx = toeplitz([5/4,1/2,0]);
[a,Pfo]=olsigest(Rx,1);
a = [1;a];
ef = zeros(N,K);
for k = 1:K; ef(:,k) = filter(a,1,x(:,k)); end;
Pf = sum(ef(M+1:N,:).*ef(M+1:N,:),1)/(N-M);
subplot(3,1,1);
plot(1:K,Pf,'g',[0,K],[Pfo,Pfo],'r--');
axis([0,K,0,2]);
xlabel('sample function index k','fontsize',label_fontsize);
ylabel('MMSE','fontsize',label_fontsize);
title('MMSE of a Second-order FLP','fontsize',title_fontsize);
text(700,1.7,['Optimum P^f = ',num2str(Pfo)],'fontsize',text_fontsize);
```

The plot is shown in Figure 6.29.

- (b) Use the second-order BLP b_k to obtain predictions $\{\hat{x}_k^b(n)\}_{n=0}^{N-2}, k = 1, \dots, K$ of $x_k(n)$. Then determine the average error

$$\hat{P}^b = \frac{1}{N-1} \sum_{n=0}^{N-2} |x_k(n) - \hat{x}_k^b(n)|^2 \quad k = 1, \dots, K$$

and plot it as a function of k . Compare it with P_o^b .

```
% (b) Second-order backward linear predictor
[b,Pbo]=olsigest(Rx,3);
b = [b;1];
eb = zeros(N,K);
for k = 1:K; eb(:,k) = filter(b,1,x(:,k)); end;
```

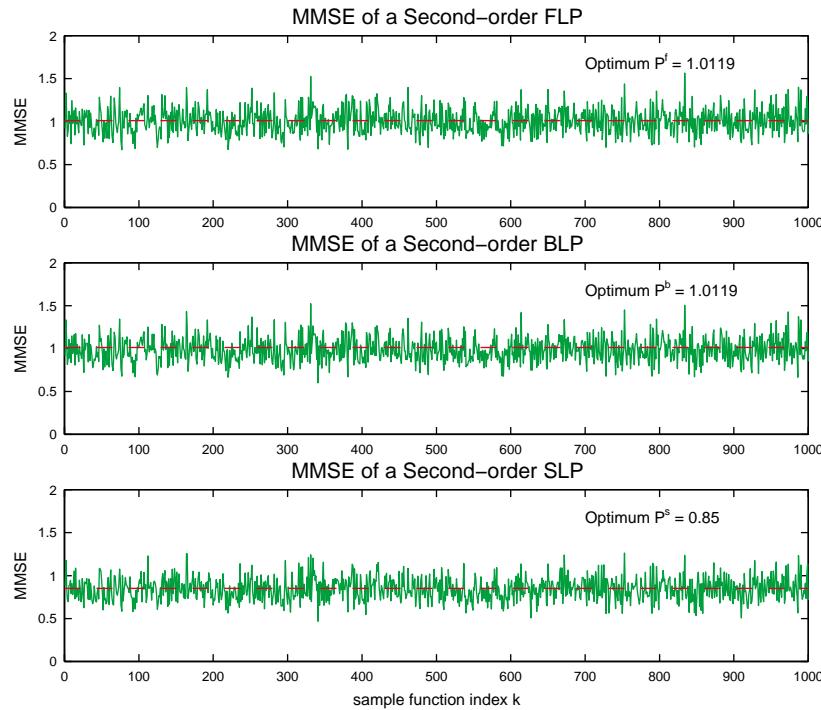


Figure 6.29: Plots of average errors vs sample function index k

```
Pb = sum(eb(1:N-M,:) .* eb(1:N-M,:),1)/(N-M);
subplot(3,1,2);
plot(1:K,Pb,'g',[0,K],[Pbo,Pbo],'r--');
axis([0,K,0,2]);
% xlabel('sample function index k','fontsize',label_fontsize);
% ylabel('MMSE','fontsize',label_fontsize);
title('MMSE of a Second-order BLP','fontsize',title_fontsize);
text(700,1.7,['Optimum P^b = ',num2str(Pbo)],'fontsize',text_fontsize);
```

The plot is shown in Figure 6.29.

- (c) Use the second-order symmetric linear smoother c_k to obtain smooth estimates $\{\hat{x}_k^c(n)\}_{n=0}^{N-2}$ of $x_k(n)$ for $k = 1, \dots, K$. Determine the average error

$$\hat{P}^s = \frac{1}{N-1} \sum_{n=1}^{N-1} |x_k(n) - \hat{x}_k^c(n)|^2 \quad k = 1, \dots, K$$

and plot it as a function of k . Compare it with P_o^s .

```
% (c) Second-order Symmetric linear predictor
[c,Pco]=olsigest(Rx,2);
c = [c(1);1;c(2)];
ec = zeros(N,K);
for k = 1:K; ec(:,k) = filter(c,1,x(:,k)); end;
Pc = sum(ec(2:N-M+1,:).*ec(2:N-M+1,:),1)/(N-M);
subplot(3,1,3);
plot(1:K,Pc,'g',[0,K],[Pco,Pco],'r--');
axis([0,K,0,2]);
```

```

xlabel('sample function index k','fontsize',label_fontsize);
ylabel('MMSE','fontsize',label_fontsize);
title('MMSE of a Second-order SLP','fontsize',title_fontsize);
text(700,1.7,['Optimum P^s = ',num2str(Pco)],'fontsize',text_fontsize);

```

The plot is shown in Figure 6.29.

- 6.30** Let $x(n) = y(n) + v(n)$ be a wide-sense stationary process. The linear, symmetric smoothing filter estimator of $y(n)$ is given by

$$\hat{y}(n) = \sum_{k=-L}^{L} h(k)x(n-k)$$

- (a) Determine the normal equations for the optimum MMSE filter.
- (b) Show that the smoothing filter \mathbf{c}_o^s has linear phase.
- (c) Use the Lagrange multiplier method to determine the MMSE M th-order estimator $\hat{y}(n) = \mathbf{c}^H \mathbf{x}(n)$, where $M = 2L + 1$, when the filter vector \mathbf{c} is constrained to be conjugate symmetric, that is, $\mathbf{c} = \mathbf{J}\mathbf{c}^*$. Compare the results with those obtained in part (a).

- 6.31** Consider the causal prediction filter discussed in Example 6.6.1.

- (a) Determine $h_c^{[D]}(n)$.

Using equation 6.6.42

$$R_{yw}(z) = \frac{R_{yx}(z)}{H_x(z^{-1})} = \frac{\frac{3}{5}}{1 - \frac{4}{5}z^{-1}} + \frac{0.3z}{1 - \frac{1}{2}z}$$

The causal part is

$$[R_{yw}(z)]_+ = \left[\frac{R_{yx}(z)}{H_x(z^{-1})} \right]_+ = \frac{\frac{3}{5}}{1 - \frac{4}{5}z^{-1}}$$

Using 6.6.21 and the relation $H_c^{[D]}(z) = z^D H_c(z)$, the optimal predictive filter is

$$H_c^{[D]}(z) = \frac{1}{\sigma_x^2 H_x(z)} \left[\frac{z^D R_{yx}(z)}{H_x(z^{-1})} \right]_+ = z^D \frac{\frac{3}{5}}{1 - \frac{4}{5}z^{-1}}$$

Therefore, taking the inverse z-transform of the above equation result in

$$h_c^{[D]}(n) = \frac{3}{8} \left(\frac{4}{5}\right)^D \left(\frac{1}{2}\right)^n u(n)$$

- (b) Using the above $h_c^{[D]}(n)$, show that

$$P_c^{[D]} = 1 - \frac{5}{8} \left(\frac{4}{5}\right)^{2D}$$

Solving directly using the following formula

$$\begin{aligned}
P_c^{[D]} &= r_y(0) - \sum_{k=0}^{\infty} h_c^{[D]}(k) r_{yx}(k) \\
&= 1 - \frac{3}{8} \sum_{k=0}^{\infty} \left(\frac{1}{2}\right)^k \left(\frac{4}{5}\right)^D \left(\frac{4}{5}\right)^k \left(\frac{4}{5}\right)^D \\
&= 1 - \frac{3}{8} \left(\frac{4}{5}\right)^2 D \sum_{k=0}^{\infty} \left(\frac{1}{2} \cdot \frac{4}{5}\right)^k
\end{aligned}$$

Evaluating the above equation results in $P_c^{[D]} = 1 - \frac{5}{8} \left(\frac{4}{5}\right)^{2D}$

6.32 Consider the causal smoothing filter discussed in Example 6.6.1.

- (a) Using $[r'_{yw}(l)]_+ = r_{yw}(l + D)u(l), D < 0$, show that $[r'_{yw}(l)]_+$ can be put in the form

$$[r'_{yw}(l)]_+ = \frac{3}{5}(\frac{4}{5})^{l+D}u(l + D) + \frac{3}{5}(2^{l+D})[u(l) - u(l + D)] \quad D < 0$$

Solving directly

$$\begin{aligned} [r'_{yw}(l)]_+ &= r_{yw}(l + D)u(l) \quad D < 0 \\ &= \left\{ \frac{3}{5}(\frac{4}{5})^{l+D}u(l + D) + (\frac{3}{5})2^{l+D}u(-l - D - 1) \right\} u(l) \\ &= (\frac{3}{5})(\frac{4}{5})^{l+D}u(l + D) + (\frac{3}{5})2^{l+D}u(-l - D - 1)u(l) \\ &= (\frac{3}{5})(\frac{4}{5})^{l+D}u(l + D) + (\frac{3}{5})(2^{l+D})[u(l) - u(l + D)] \quad D < 0 \end{aligned}$$

- (b) Hence, show that $[R'_{yw}(l)]_+$ is given by

$$[R'_{yw}(l)]_+ = \frac{3}{5} \frac{z^D}{1 - \frac{4}{5}z^{-1}} + \frac{3}{5}(2^D) \sum_{l=0}^{-D-1} 2^l z^{-l}$$

Taking the z-transform, the above result is found directly

- (c) Finally, using (6.6.21), prove (6.6.54)

$$\begin{aligned} H_c^{[D]}(z), \quad D < 0 &= \frac{5}{8} \left(\frac{1 - \frac{4}{5}z^{-1}}{1 - \frac{1}{2}z^{-1}} \right) \frac{3}{5} \left[\left(\frac{z^D}{1 - \frac{4}{5}z^{-1}} \right) + 2^D \sum_{l=0}^{-D-1} 2^l z^{-l} \right] \\ &= \frac{3}{8} \left[\frac{z^D}{1 - \frac{1}{2}z^{-1}} + \frac{(1 - \frac{4}{5}z^{-1})2^D \sum_{l=0}^{-D-1} 2^l z^{-l}}{1 - \frac{1}{2}z^{-1}} \right] \\ &= \frac{3}{8} \left[\frac{z^D}{1 - \frac{1}{2}z^{-1}} + \frac{2^D \sum_{l=0}^{-D-1} 2^l z^{-l}}{1 - \frac{1}{2}z^{-1}} - \left(\frac{4}{5} \right) \frac{2^D \sum_{l=0}^{-D-1} 2^l z^{-l-1}}{1 - \frac{1}{2}z^{-1}} \right] \end{aligned}$$

6.33 Proof of (6.6.57)

- (a) Consider

$$\begin{aligned} [r'_{yw}(l)]_+ &= r_{yw}(l + D)u(l), \quad D < 0 \\ &= \frac{3}{5} \left\{ \left(\frac{4}{5} \right)^{l+D} u(l + D) + 2^{l+D}u(-l - D - 1) \right\} u(l), \quad D < 0 \\ &= \frac{3}{5} \left(\frac{4}{5} \right)^{l+D} u(l + D) + \frac{3}{5}2^{l+D} \{u(l) - u(l + D)\}, \quad D < 0 \end{aligned}$$

Hence

$$\begin{aligned} [R'_{yw}(z)]_+ &= \frac{3}{5} \frac{z^D}{1 - \frac{4}{5}z^{-1}} + \frac{3}{5}2^D \sum_{l=0}^{-D-1} 2^l z^{-l} = \frac{3}{5} \frac{z^D}{1 - \frac{4}{5}z^{-1}} + \frac{3}{5}2^D \sum_{l=0}^{-D-1} \left(\frac{2}{z} \right)^l \\ &= \frac{3}{5} \frac{z^D}{1 - \frac{4}{5}z^{-1}} + \frac{3}{5}2^D \frac{1 - (z/D)^D}{1 - 2z^{-1}} = \frac{3}{5} \frac{z^D}{1 - \frac{4}{5}z^{-1}} + \frac{3}{5} \frac{2^D - z^D}{1 - 2z^{-1}} \end{aligned}$$

or

$$\left[R'_{yw}(z) \right]_+ = \frac{3}{5} \left(\frac{z^D}{1 - \frac{4}{5}z^{-1}} + \frac{2^D - z^D}{1 - 2z^{-1}} \right), \quad D < 0$$

(b) Using (6.6.21) and substituting the above result

$$\begin{aligned} H_c^{[D]}(z) &= \frac{5}{8} \left(\frac{1 - \frac{4}{5}z^{-1}}{1 - \frac{1}{2}z^{-1}} \right) \frac{3}{5} \left(\frac{z^D}{1 - \frac{4}{5}z^{-1}} + \frac{2^D - z^D}{1 - 2z^{-1}} \right) \\ &= \frac{3}{8} \left(\frac{1 - \frac{4}{5}z^{-1}}{1 - \frac{1}{2}z^{-1}} \right) \left(\frac{z^D (1 - 2z^{-1}) + (2^D - z^D) (1 - \frac{4}{5}z^{-1})}{(1 - \frac{4}{5}z^{-1})(1 - 2z^{-1})} \right) \\ &= \frac{3}{8} \left[\frac{2^D (1 - \frac{4}{5}z^{-1}) + \frac{3}{5}z^{D-1}}{(1 - \frac{1}{2}z^{-1})(1 - 2z^{-1})} \right], \quad D < 0 \end{aligned}$$

Hence as $-D \rightarrow \infty$ we have $2^D \rightarrow 0$ and $z^{D-1} \rightarrow z^D$. Hence

$$\lim_{-D \rightarrow \infty} H_c^{[D]}(z) = \frac{9}{40} \left[\frac{z^D}{(1 - \frac{1}{2}z^{-1})(1 - 2z^{-1})} \right] = z^D H_{nc}(z), \quad D < 0$$

(c) Finally, consider

$$P_c^{[D]} = r_y(0) - \sum_{k=0}^{\infty} h_c^{[D]}(k) r_{yx}(k+D) = r_y(0) - \sum_{k=-\infty}^{\infty} h_c^{[D]}(k) r_{yx}(k+D) u(k)$$

As $-D \rightarrow \infty$ we have

$$h_c^{[D]}(k) \rightarrow h_c(k) \text{ and } r_{yx}(k+D) u(k) \rightarrow r_{yx}(k)$$

Hence

$$\lim_{-D \rightarrow \infty} P_c^{[D]} = r_y(0) - \sum_{k=-\infty}^{\infty} h_c(k) r_{yx}(k) = P_{nc}$$

6.34 Consider the block diagram of a simple communication system shown in Figure 6.38 in which $H_1(z) = 1/(1 + 0.95z^{-1})$.

(a) Determine a second-order optimum FIR filter ($M = 2$) that estimates the signal $s(n)$ from the received signal $x(n) = z(n) + v(n)$. What is the corresponding MMSE P_o ?

From the block diagram, $s(n) = -0.95s(n-1) + w(n)$ and $z(n) = 0.85z(n-1) + s(n)$. The optimum filter is $C_o = R_z^{-1}r_{sx}$. The first two lags of the autocorrelation are

$$r_z(0) = 3.936 \quad \sigma_w^2 = 1.181 \quad r_z(1) = -2.045 \quad \sigma_w^2 = -0.614$$

and the first two lags of the cross-correlation are

$$\begin{aligned} r_{sx}(l) &= E\{s(n)x^*(n+l)\} = E\{s(n)z^*(n+l)\} \\ &= E\{[z(n) - 0.85z(n-1)]z^*(n+l)\} = r_z(l) - 0.85r_z(l-1) \\ r_{sx}(0) &= r_z(0) - 0.85, \quad r_z(1) = 1.702 \\ r_{sx}(1) &= r_z(1) - 0.85, \quad r_z(0) = -1.614 \end{aligned}$$

The optimum filter coefficients are shown in the top plot of Figure 6.34.

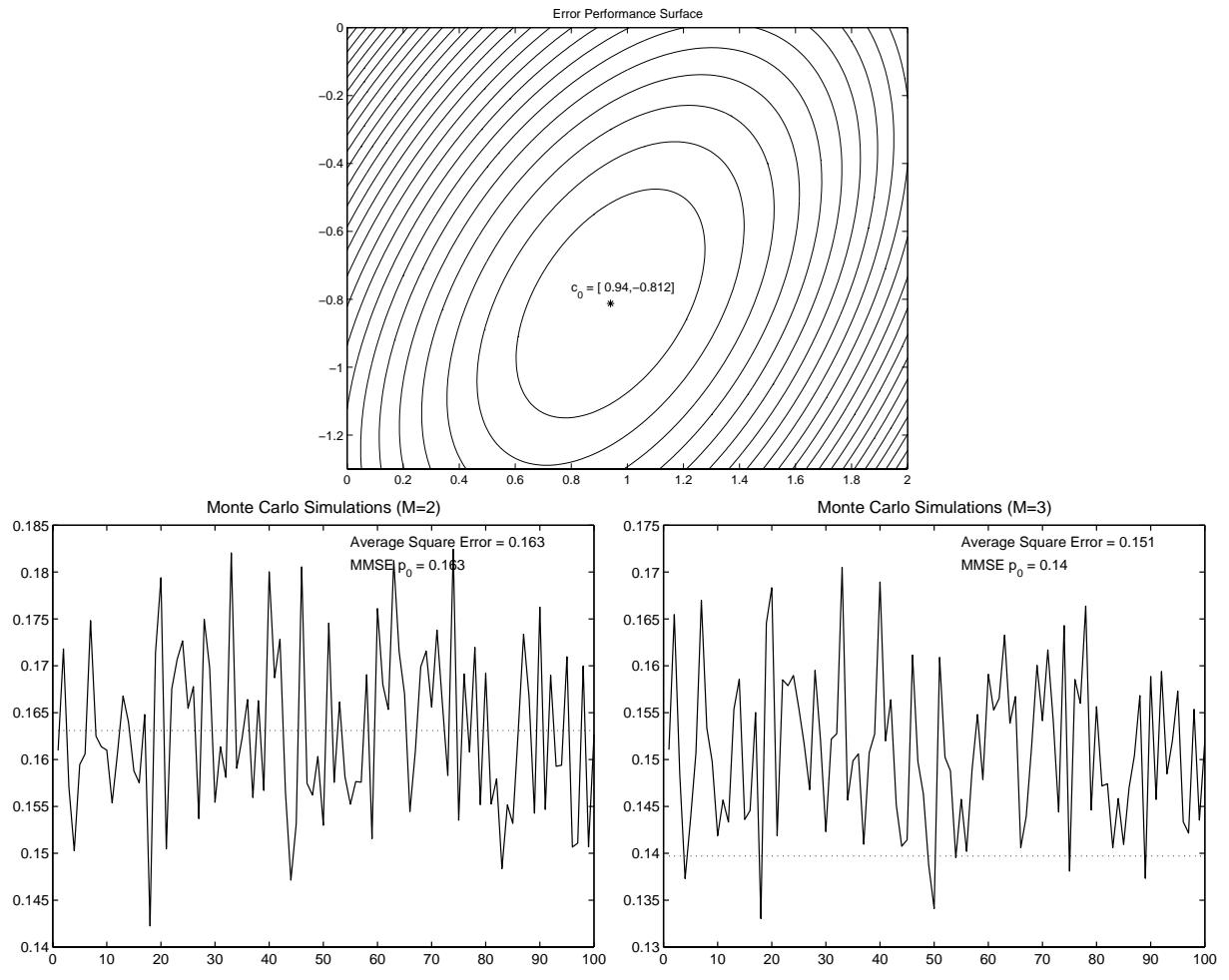


Figure 6.34: Error Performance Surface and Monte Carlo Simulations

- (b) Plot the error performance surface

The top plot of Figure 6.34 has the error performance surface plot. The surface is defined by

$$\varepsilon^2 = \sigma_y^2 - 2c^T d + c^T R_x c$$

where $\sigma_y^2 = 3.0769$, $d = [1.7023 \ - 1.6172]^T$, and $r_x = [1.2808 \ - 0.6135]^T$.

- (c) Use a Monte Carlo simulation to verify the theoretically obtained MMSE in part(a).

The lower left plot of Figure 6.34 shows a Monte Carlo simulation, with average square error and the theoretical MMSE.

- (d) Repeat part(a) for $M = 3$.

Figure 6.34 shows a Monte Carlo simulation for a third order FIR filter, where $r_x = [1.2808 \ - 0.6135 \ 1.0945]^T$ and $d = [1.7023 \ - 1.6172 \ 1.6160]^T$

6.35 Matlab program to reproduce the results shown in Figure 6.9.1.

- (a) The Matlab script is shown below and the plots are shown in Figure 6.35a.

```

clc; close all;
set(0,'defaultaxesfontsize',default_fontsize);

% Given parameters
var0 = 0.25;
a = 0.6;

% (a) Plots for rho = 0.1, -0.8, and 0.8, M = 8
M = 8; n = 0:M-1;
s = a.^n; s0 = s';
Ps = 1;

Hf_1 = figure('units','SCRUN','position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',[0,0,5,4]);
set(Hf_1,'NumberTitle','off','Name','Pr0635a');

subplot(2,2,1), stem(n,s,'filled','g');
ylabel('s(n)', 'fontsize',label_fontsize);
title('Signal s(n)', 'fontsize',title_fontsize);
axis([-1,M,0,1]);

ro = 0.1; Pv = var0/(1-ro*ro);
rv = ro.^n; Rv = Pv*toeplitz(rv);
c = inv(Rv)*s0;
subplot(2,2,2), stem(n,c,'filled','g');
ylabel('c(n)', 'fontsize',label_fontsize);
SNRmax = Ps*(s0'*inv(Rv)*s0)
title(['\rho = 0.1, SNR= ',num2str(SNRmax), ' dB'], 'fontsize',title_fontsize);
axis([-1,M,0,4]);

ro = -0.8; Pv = var0/(1-ro*ro);
rv = ro.^n; Rv = Pv*toeplitz(rv);
c = inv(Rv)*s0;

```

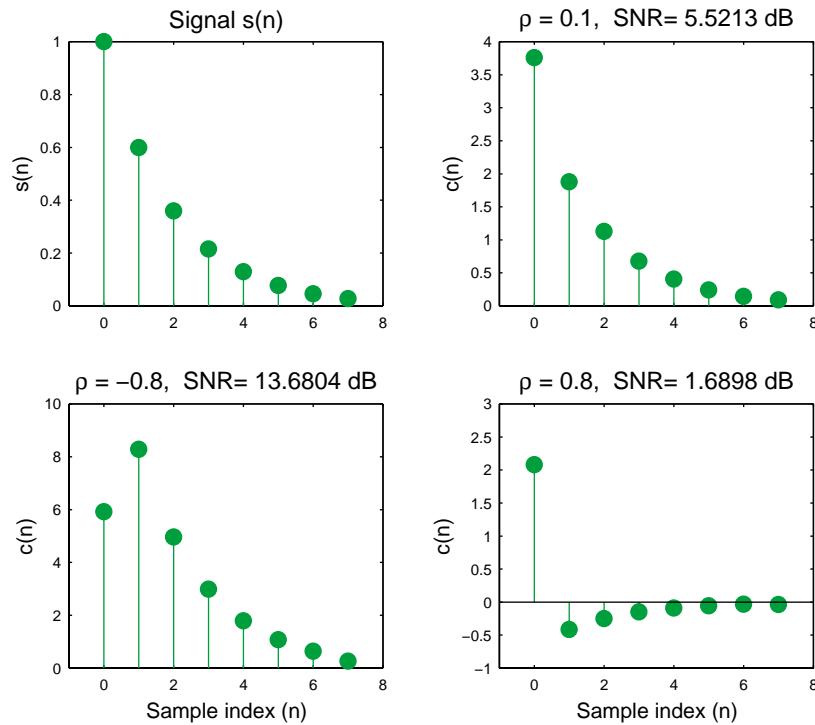


Figure 6.35a: Signal and impulse responses of optimal matched filters for $M = 8$

```

subplot(2,2,3), stem(n,c,'filled','g');
xlabel('Sample index (n)', 'fontsize', label_fontsize);
ylabel('c(n)', 'fontsize', label_fontsize);
SNRmax = Ps*(s0*inv(Rv)*s0)
title(['\rho = -0.8, SNR= ', num2str(SNRmax), ' dB'], 'fontsize', title_fontsize);
axis([-1,M,0,10]);

ro = 0.8; Pv = var0/(1-ro*ro);
rv = ro.^n; Rv = Pv*toeplitz(rv);
c = inv(Rv)*s0;
subplot(2,2,4), stem(n,c,'filled','g'); hold on;
plot([-1,M],[0,0], 'w');
xlabel('Sample index (n)', 'fontsize', label_fontsize);
ylabel('c(n)', 'fontsize', label_fontsize);
SNRmax = Ps*(s0*inv(Rv)*s0)
title(['\rho = 0.8, SNR= ', num2str(SNRmax), ' dB'], 'fontsize', title_fontsize);
axis([-1,M,-1,3]);

```

- (b) The Matlab script is shown below and the plots are shown in Figure 6.35b.

```

% (b) Plots for rho = 0.1, -0.8, and 0.8, M = 16
M = 16; n = 0:M-1;
s = a.^n; s0 = s';
Ps = 1;

Hf_2 = figure('units',SCRUN,'position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',[0,0,5,4]);

```

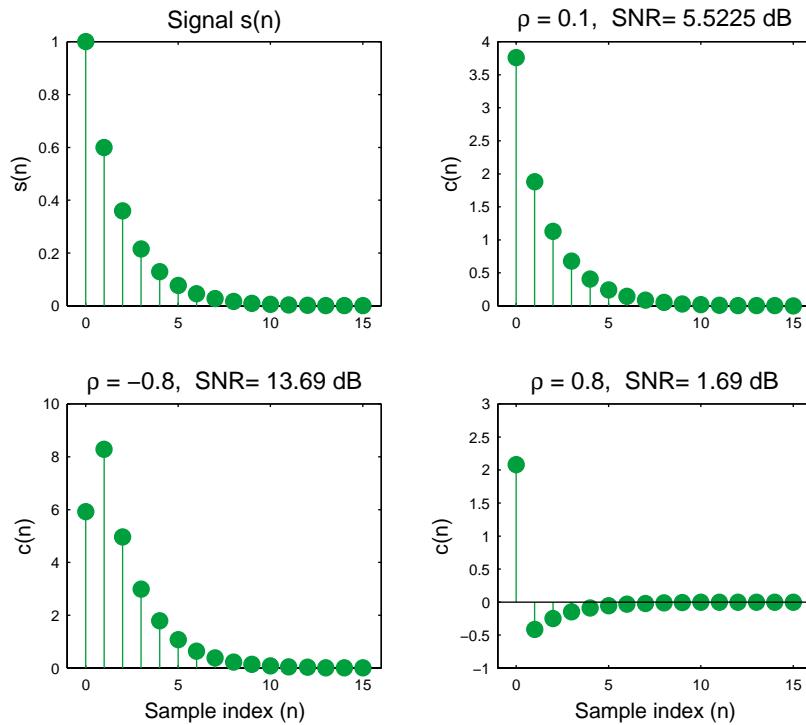


Figure 6.35b: Signal and impulse responses of optimal matched filters for $M = 16$

```

set(Hf_2,'NumberTitle','off','Name','Pr0635b');

subplot(2,2,1), stem(n,s,'filled','g');
ylabel('s(n)', 'fontsize',label_fontsize);
title('Signal s(n)', 'fontsize',title_fontsize);
axis([-1,M,0,1]);

ro = 0.1; Pv = var0/(1-ro*ro);
rv = ro.^n; Rv = Pv*toeplitz(rv);
c = inv(Rv)*s0;
subplot(2,2,2), stem(n,c,'filled','g');
ylabel('c(n)', 'fontsize',label_fontsize);
SNRmax = Ps*(s0'*inv(Rv)*s0)
title(['\rho = 0.1, SNR= ',num2str(SNRmax), ' dB'], 'fontsize',title_fontsize);
axis([-1,M,0,4]);

ro = -0.8; Pv = var0/(1-ro*ro);
rv = ro.^n; Rv = Pv*toeplitz(rv);
c = inv(Rv)*s0;
subplot(2,2,3), stem(n,c,'filled','g');
xlabel('Sample index (n)', 'fontsize',label_fontsize);
ylabel('c(n)', 'fontsize',label_fontsize);
SNRmax = Ps*(s0'*inv(Rv)*s0)
title(['\rho = -0.8, SNR= ',num2str(SNRmax), ' dB'], 'fontsize',title_fontsize);
axis([-1,M,0,10]);

```

```

ro = 0.8; Pv = var0/(1-ro*ro);
rv = ro.^n; Rv = Pv*toeplitz(rv);
c = inv(Rv)*s0;
subplot(2,2,4), stem(n,c,'filled','g'); hold on;
plot([-1,M],[0,0],'w');
xlabel('Sample index (n)', 'fontsize', label_fontsize);
ylabel('c(n)', 'fontsize', label_fontsize);
SNRmax = Ps*(s0*inv(Rv)*s0)
title(['\rho = 0.8, SNR= ', num2str(SNRmax), ' dB'], 'fontsize', title_fontsize);
axis([-1,M,-1,3]);

```

- 6.36** Matlab program to reproduce the plot shown in Figure 6.9.2. The Matlab script is shown below and the plots are shown in Figure 6.36.

```

clc; close all;
set(0,'defaultaxesfontsize',default_fontsize);

Hf_1 = figure('units',SCRUN,'position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',[0,0,3,3]);
set(Hf_1,'NumberTitle','off','Name','Pr0636a');

ro = -0.81; % correlation coefficient
R = toeplitz([1 ro]); % Autocorrelation matrix
th = [0:1000]*2*pi/1000; % Angle samples
x = [cos(th);sin(th)]; % c'*c = 1
[Q,L] = eig(R); % Eigenanalysis
lam1 = L(1,1); % Lambda 1
lam2 = L(2,2); % Lambda 2
y = Q*sqrt(L)*x; % c'*R*c = 1
y1 = Q*sqrt(L)*(1/sqrt(lam1))*x;
y2 = Q*sqrt(L)*(1/sqrt(lam2))*x;
minx = floor(min([x(1,:),y(1,:),y1(1,:),y2(1,:)]));
maxx = ceil(max([x(1,:),y(1,:),y1(1,:),y2(1,:)]));
miny = floor(min([x(2,:),y(2,:),y1(2,:),y2(2,:)]));
maxy = ceil(max([x(2,:),y(2,:),y1(2,:),y2(2,:)]));

plot(x(1,:),x(2,:), 'g', y(1,:),y(2,:), 'm'); hold on;
plot(y1(1,:),y1(2,:), 'c:', y2(1,:),y2(2,:), 'r--');
plot([minx,maxx],[0,0], 'w', [0,0],[miny,maxy], 'w');
axis equal
axis([minx,maxx,miny,maxy]);
title('\rho = -0.81', 'fontsize', title_fontsize);
xlabel('c_1', 'fontsize', label_fontsize);
ylabel('c_2', 'fontsize', label_fontsize);
grid; hold off;
[leg,hobj] = legend('c^Hc=1', 'c^HRC=1',...
    'c^HRC=\lambda_1', 'c^HRC=\lambda_2', 1);
pause

```

```

exportfig(gcf,'p0636a.eps','fontmode','scaled',...
    'linemode','scaled','color','cmyk');

Hf_2 = figure('units',SCRUN,'position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',[0,0,3,3]);
set(Hf_1,'NumberTitle','off','Name','Pr0636b');

ro = 0.81; % correlation coefficient
R = toeplitz([1 ro]); % Autocorrelation matrix
th = [0:1000]*2*pi/1000; % Angle samples
x = [cos(th);sin(th)]; % c'*c = 1
[Q,L] = eig(R); % Eigenanalysis
lam1 = L(1,1); % Lambda 1
lam2 = L(2,2); % Lambda 2
y = Q*sqrt(L)*x; % c'*R*c = 1
y1 = Q*sqrt(L)*(1/sqrt(lam1))*x;
y2 = Q*sqrt(L)*(1/sqrt(lam2))*x;
minx = floor(min([x(1,:),y(1,:),y1(1,:),y2(1,:)]));
maxx = ceil(max([x(1,:),y(1,:),y1(1,:),y2(1,:)]));
miny = floor(min([x(2,:),y(2,:),y1(2,:),y2(2,:)]));
maxy = ceil(max([x(2,:),y(2,:),y1(2,:),y2(2,:)]));

plot(x(1,:),x(2,:), 'g',y(1,:),y(2,:), 'm');hold on;
plot(y1(1,:),y1(2,:), 'c:',y2(1,:),y2(2,:), 'r--');
plot([minx,maxx],[0,0], 'w',[0,0],[miny,maxy], 'w');
title('rho = 0.81','fontsize',title_fontsize);
xlabel('c_1','fontsize',label_fontsize);
ylabel('c_2','fontsize',label_fontsize);
axis equal
axis([minx,maxx,miny,maxy]);
grid; hold off;
[legh,objh] = legend('c^Hc=1','c^HRC=1',...
    'c^HRC=\lambda_1','c^HRC=\lambda_2',1);
pause

exportfig(gcf,'p0636b.eps','fontmode','scaled',...
    'linemode','scaled','color','cmyk');

```

6.37 Interference rejection filter analysis.

The signal of interest is random with PSD $\mathbf{R}_s = P_a \mathbf{I}$. The background is colored with correlation $r_v(l) = \rho^{|l|}$, $1 \leq l \leq M$.

- (a) SNRs for the matched and forward linear prediction error (LPE) filters:

M = 2: For the eigenfilter design, the autocorrelation matrix is given by

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

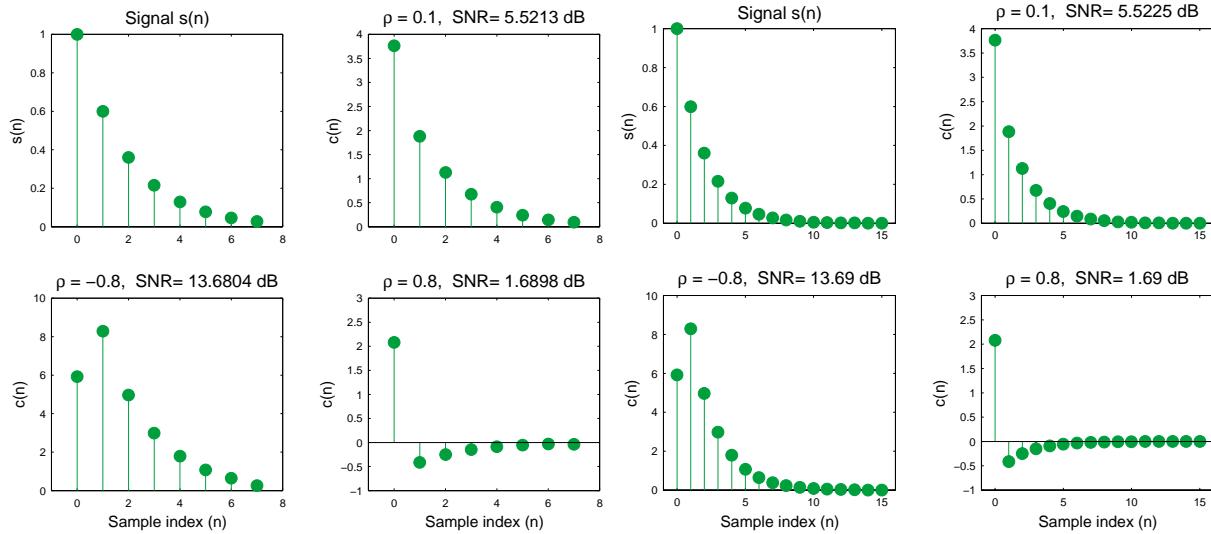


Figure 6.36: Geometric interpretation of the optimization process for $\rho = -0.8$ and $\rho = 0.8$

whose eigenvalues are: $-\rho + 1, \rho + 1$. The minimum eigenvalue is $\lambda_{\min} = (1 - \rho)$ and the maximum SNR is

$$\text{SNR}_{\max} = \frac{1}{\lambda_{\min}} = \frac{1}{1 - \rho}$$

For the LPE filter the maximum SNR is given by

$$\text{SNR}_{\max} = \frac{\mathbf{a}_o^T \mathbf{a}_o}{\mathbf{a}_o^T \mathbf{R}_v \mathbf{a}_o}$$

where \mathbf{a}_o is the forward linear predictor given by $\mathbf{a}_o = [1 \ -\rho]^T$. Thus

$$\text{SNR}_{\max} = \frac{[1 \ -\rho] \begin{bmatrix} 1 \\ -\rho \end{bmatrix}}{[1 \ -\rho] \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -\rho \end{bmatrix}} = \frac{1 + \rho^2}{1 - \rho^2}$$

M = 3: For the eigenfilter design, the autocorrelation matrix is given by

$$\mathbf{R}_v = \begin{bmatrix} 1 & \rho & \rho^4 \\ \rho & 1 & \rho \\ \rho^4 & \rho & 1 \end{bmatrix}$$

whose eigenvalues are: $1 - \rho^4, 1 + \frac{1}{2}\rho^4 + \frac{1}{2}\sqrt{(\rho^8 + 8\rho^2)}, 1 + \frac{1}{2}\rho^4 - \frac{1}{2}\sqrt{(\rho^8 + 8\rho^2)}$. The minimum eigenvalue is

$$\begin{aligned} \lambda_{\min} &= 1 + \frac{1}{2}\rho^4 - \frac{1}{2}\sqrt{(\rho^8 + 8\rho^2)} = \frac{1}{2}(2 + \rho^4 - \rho^4\sqrt{1 + 8\rho^{-6}}) \\ &= \frac{1}{2}[2 + \rho^4(1 - \sqrt{1 + 8\rho^{-6}})] \end{aligned}$$

Hence the maximum SNR is

$$\text{SNR}_{\max} = \frac{1}{\lambda_{\min}} = \frac{2}{2 + \rho^4(1 - \sqrt{1 + 8\rho^{-6}})}$$

For the LPE we have to first solve

$$\begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = -\begin{bmatrix} \rho \\ \rho^4 \end{bmatrix} \Rightarrow \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} -(1+\rho^2)\rho \\ \rho^2 \end{bmatrix}$$

Thus $\mathbf{a}_o = [1 \ -(1+\rho^2)\rho \ \rho^2]^T$. Hence

$$\begin{aligned} \text{SNR}_{\max} &= \frac{[1 \ -(1+\rho^2)\rho \ \rho^2] \begin{bmatrix} 1 \\ -(1+\rho^2)\rho \\ \rho^2 \end{bmatrix}}{[1 \ -(1+\rho^2)\rho \ \rho^2] \begin{bmatrix} 1 & \rho & \rho^4 \\ \rho & 1 & \rho \\ \rho^4 & \rho & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -(1+\rho^2)\rho \\ \rho^2 \end{bmatrix}} \\ &= \frac{1+\rho^2+3\rho^4+\rho^6}{1-\rho^2-\rho^4+\rho^6} = \frac{1+\rho^2+3\rho^4+\rho^6}{(\rho^2-1)(\rho^4-1)} \end{aligned}$$

M = 4: Using the above approach, we obtain

$$\text{SNR}_{\max}^{(\text{matched})} = \frac{2}{2-\rho+\rho^9-\sqrt{(\rho-\rho^9)^2+4(\rho-\rho^4)^2}}$$

and

$$\text{SNR}_{\max}^{(\text{LPE})} = \frac{1+3\rho^4+3\rho^6+2\rho^8+\rho^{10}}{(1-\rho^2)^2(1-\rho^6)}$$

(b) Plots of the SNRs for $M = 2, 3$, and 4 and $\rho = 0.6, 0.8, 0.9, 0.95, 0.99$, and 0.995 :

```
% (b) Plots of SNRs
ro = [0.6,0.8,0.9,0.95,0.99,0.995];

Hf_1 = figure('units','SCRUN','position',SCRPOS,...
    'paperunits','PAPUN','paperposition',[0,0,5,4]);
set(Hf_1,'NumberTitle','off','Name','Pr0637b');

% 1. M = 2
SNR_M = 1. / (1-ro); SNR_Mdb = 10*log10(SNR_M);
SNR_L = (1+ro.*ro)./(1-ro.*ro); SNR_Ldb = 10*log10(SNR_L);
subplot(3,1,1);
plot(ro,SNR_Mdb,'g',ro,SNR_Ldb,'r:');
%xlabel('\rho','fontsize',label_fontsize);
ylabel('Decibels','fontsize',label_fontsize);
title('M = 2','fontsize',title_fontsize);
legend('Matched filter','LPE filter',0)

% 2. M = 3
SNR_M = 2. / (2+ro.^4.*(1-sqrt(1+8*ro.^(-6)))); % Error in original code
SNR_Mdb = 10*log10(SNR_M);
SNR_L = (1+ro.^2+3*ro.^4+ro.^6)./(ro.^2-1).*(ro.^4-1);
SNR_Ldb = 10*log10(SNR_L);
```

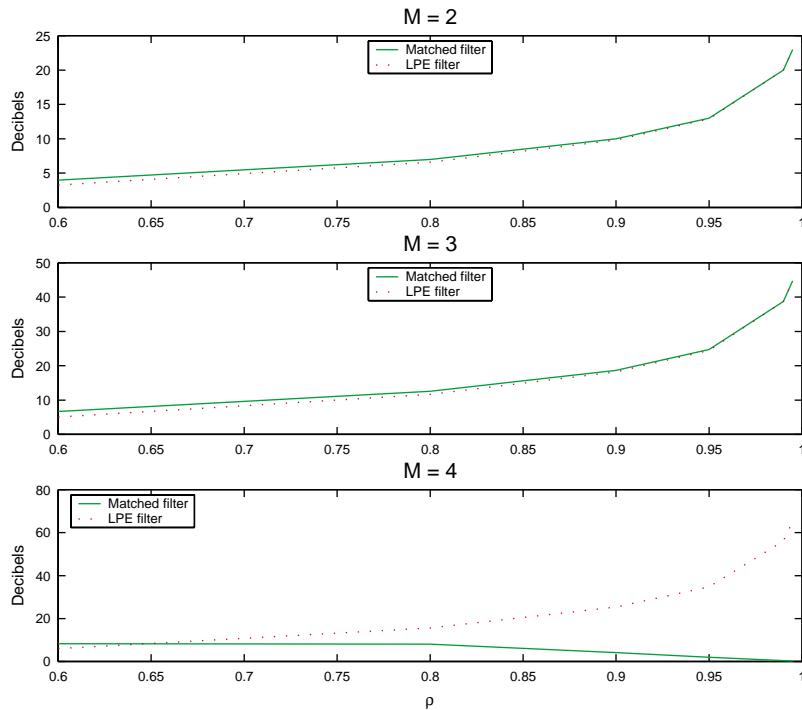


Figure 6.37b: Plots of the SNRs for matched and LPE filters

```

subplot(3,1,2);
plot(ro,SNR_Mdb,'g',ro,SNR_Ldb,'r:');
%xlabel('\rho','fontsize',label_fontsize);
ylabel('Decibels','fontsize',label_fontsize);
title('M = 3','fontsize',title_fontsize);
legend('Matched filter','LPE filter',0)

% 3. M = 4
SNR_M = 2./(2-ro+ro.^9-sqrt((ro-ro.^9).^2+4*(ro-ro.^4).^2));
SNR_Mdb = 10*log10(SNR_M);
SNR_L = (1+3*ro.^4+3*ro.^6+2*ro.^8+ro.^10)./((1-ro.^2).^2.*(1-ro.^6));
SNR_Ldb = 10*log10(SNR_L);
subplot(3,1,3);
plot(ro,SNR_Mdb,'g',ro,SNR_Ldb,'r:');
xlabel('\rho','fontsize',label_fontsize);
ylabel('Decibels','fontsize',label_fontsize);
title('M = 4','fontsize',title_fontsize);
legend('Matched filter','LPE filter',0)

```

The plots are shown in Figure 6.37b.

- (c) Magnitude responses of matched, LPE, and binomial filters for $M = 3$ and $\rho = 0.9$:

```

% (c) Magnitude response plots
M = 3; ro = 0.9
gr=(1+sqrt(5))/2;

```

```
Hf_2 = figure('units',SCRUN,'position',SCRPOS,...
```

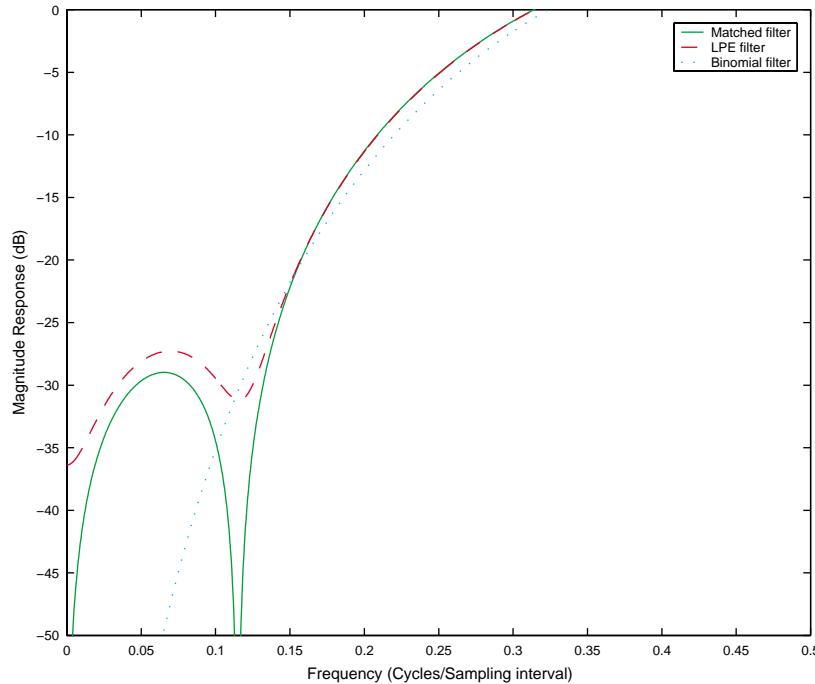


Figure 6.37c: Magnitude responses of matched, LPE, and binomial filters

```

'paperunits',PAPUN,'paperposition',[0,0,5,4]);
set(Hf_2,'NumberTitle','off','Name','Pr0637c');

r = [ro^0 ro^1 ro^4 ro^9];
R = toeplitz(r(1:4));
[Q,D] = eig(R);
lmin = D(1,1)
q = Q(:,1);
[Hq,om] = freqz(q,1,1000);
[a,Pf] = olsigest(R,1);
a1 = [1;a]; a1=a1/norm(a1);
b1 = [1 -1];
b = conv(b1,conv(b1,conv(b1,b1))); b=b/norm(b);
[Hb,om] = freqz(b,1,1000);
[Ha,om] = freqz(a1,1,1000);
plot(om/(2*pi),20*log10(abs(Hq)),'g-',
     om/(2*pi),20*log10(abs(Ha)),'r--',
     om/(2*pi),20*log10(abs(Hb)),'c:');
legend('Matched filter','LPE filter', 'Binomial filter');
ylabel('Magnitude Response (dB)', 'fontsize',label_fontsize)
xlabel('Frequency (Cycles/Sampling interval)', 'fontsize',label_fontsize);
axis([0 0.5 -50 0]);

```

The plots are shown in Figure 6.37c.

6.38 Determine the matched filter for the deterministic pulse $s(n) = \cos \omega_0 n$ for $0 \leq n \leq M-1$ and zero elsewhere.

- (a) White noise with variance $\sigma_v^2 = 1$

For the matched filter case in white noise, the spectrum of the filter matches the spectrum of the signal.

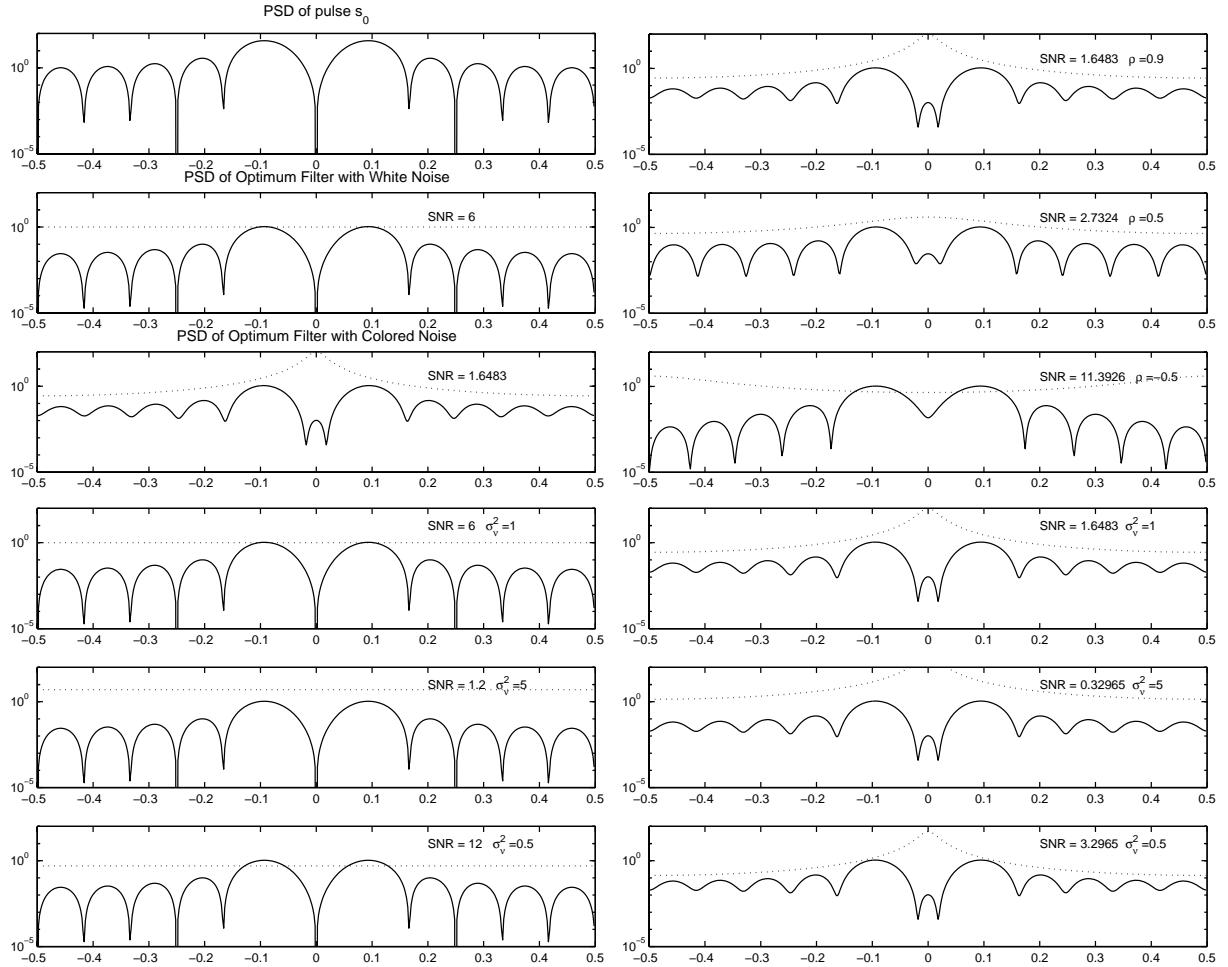


Figure 6.38:

The equation for the optimum filter is

$$c_o = \kappa s_o$$

where κ is chosen such that $c_o^H s_o = 1$.

- (b) Colored noise with autocorrelation $r_v(l) = \sigma_v^2 \rho^{|l|} / (1 - \rho^2)$, and $\rho = 0.9$.

For the matched filter case in colored noise, the equation for the optimum filter is

$$c_o = \kappa R_v^{-1} s_o$$

where $\kappa = (s_o^H R_v^{-1} s_o)^{-1}$. The upper left plot of Figure 6.38 shows the spectrum of the signal, the spectrum of the matched filter in white noise, and the spectrum of the matched filter in colored noise. The dotted line in these plots represents the spectrum of the noise. As can be easily seen from this plot, the matched filter will emphasize the spectrum with the least amount of noise energy, and de-emphasize the spectrum with higher noise energy.

- (c) Study the effect of the SNR in part(a) by varying the value of σ_v^2

The lower left and right plots of figure 6.38 show the matched filter for different values of σ_v^2 in white noise and colored noise respectively.

- (d) Study the effect of the noise correlation in part(b) by varying the value of ρ

The upper right plot shows the matched filter for the colored noise case with different values of ρ

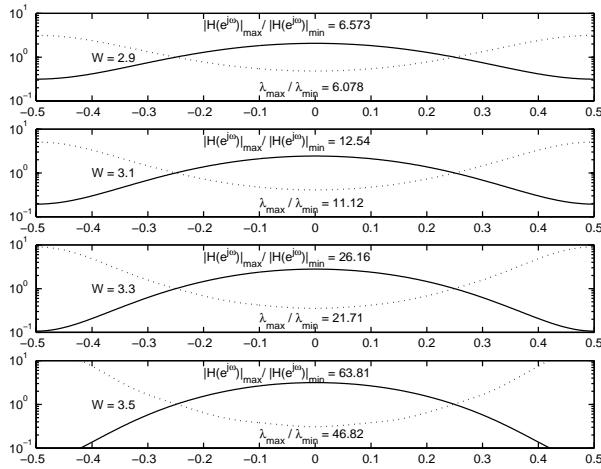


Figure 6.39: Magnitude Response

6.39 Using the equalization experiment in Example 6.8.1 with $M = 11$ and $D = 7$.

- (a) Compute and plot the magnitude response $|H(e^{j\omega})|$ of the channel and $|C_o(e^{j\omega})|$ of the optimum equalizer for $W = 2.9, 3.1, 3.3$, and 3.5 and comment upon the results.

Using Example 6.8.1, the magnitude of $|H(e^{j\omega})|$ is shown for each W in Figure 6.39 where $H(e^{j\omega})$ is

$$H(e^{j\omega}) = e^{-2j\omega}[1 + (1 + \cos(2\pi/W)) \cos \omega]$$

The optimum filter coefficients are found directly from $C_o = R_x^{-1}d$ where $R_x = E\{x(n)x^*(n)\}$ and $d = E\{a(n)x^*(n)\}$. Using equation 6.8.41,

$$D = 7 \quad d(l) = h(7-l) \Rightarrow d(4) = h(3) \quad d(5) = h(2) \quad d(6) = h(1) \quad d(l) = 0 \text{ elsewhere}$$

Figure 6.39 shows the magnitude response of the optimum filter for each W . Note that the magnitude response of the filter is the inverse of the channel. In effect, removing the influence of the channel from the signal.

- (b) Compute the spectral dynamic range $|H(e^{j\omega})|_{max}/|H(e^{j\omega})|_{min}$ of the channel and the eigenvalue spread $\lambda_{max}/\lambda_{min}$ of the $M \times M$ input correlation matrix.

Figure 6.39 shows the values for the spectral dynamic range and the eigenvalue spread for each value of W . Note that the value of these two parameters has a direct relation to each other.

6.40 In this problem we clarify some of the properties of the MSE equalizer discussed in Example 6.8.1.

- (a) Compute and plot the MMSE P_o as a function of M , and recommend how to choose a “reasonable” value.
- (b) Compute and plot P_o as a function of the delay D for $0 \leq D \leq 11$. What is the best value of D ?
- (c) Study the effect of input SNR upon P_o for $M = 11$ and $D = 7$ by fixing $\sigma_y^2 = 1$ and varying σ_v^2 .

6.41 In this problem we formulate the design of optimum linear signal estimators (LSE) using a constrained optimization framework. To this end we consider the estimator $e(n) = c_0^*x(n) + \dots + c_M^*x(n-M) \triangleq c^H x(n)$ and we wish to minimize the output power $E\{|e(n)|^2\} = c^H R c$. To prevent the trivial solution $c = 0$ we need to impose some constraint on the filter coefficients and use Lagrange multipliers to determine the minimum. Let u_i be an $M \times 1$ vector with one at the i th position and zeros elsewhere.

- (a) Show that minimizing $c^H R c$ under the linear constraint $u_i^T c = 1$ provides the following estimators: FLP if $i = 0$, BLP if $i = M$, and linear smoother if $i \neq 0, M$.

- (b) Determine the appropriate set of constraints for the L -steps ahead linear predictor, defined by $c_0 = 1$ and $\{c_k = 0\}_1^{L-1}$, and solve the corresponding constrained optimization problem. Verify your answer by obtaining the normal equations using the orthogonality principle.
- (c) Determine the optimum linear estimator by minimizing $c^H R c$ under the quadratic constraints $c^H c = 1$ and $c^H W c = 1$ (W is a positive definite matrix) which impose a constraint on the length of the filter vector.

Algorithms and Structures for Optimum Linear Filters

7.1 By first computing the matrix product

$$\begin{bmatrix} \mathbf{R}_m & \mathbf{r}_m^b \\ \mathbf{r}_m^{bH} & \rho_m^b \end{bmatrix} \begin{bmatrix} \mathbf{I}_m & -\mathbf{R}_m^{-1}\mathbf{r}_m \\ \mathbf{0}_m & 1 \end{bmatrix}$$

and then the determinants of both sides, prove Equation (7.1.25). Another proof, obtained using the LDL^H decomposition, is given by Equation (7.2.4).

Computing the matrix product directly

$$\begin{bmatrix} \mathbf{R}_m & \mathbf{r}_m^b \\ \mathbf{r}_m^{bH} & \rho_m^b \end{bmatrix} \begin{bmatrix} \mathbf{I}_m & -\mathbf{R}_m^{-1}\mathbf{r}_m^b \\ \mathbf{0}_m & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_m & \mathbf{0}_m \\ \mathbf{r}_m^{bH} & \rho_m^b - \mathbf{r}_m^{bH}\mathbf{R}_m^{-1}\mathbf{r}_m^b \end{bmatrix}$$

The determinants of both sides are found to be

$$\det \mathbf{R}_{m+1} = (\rho_m^b - \mathbf{r}_m^{bH}\mathbf{R}_m^{-1}\mathbf{r}_m^b)\det \mathbf{R}_m$$

Using (7.1.23)

$$\alpha_m^b = \rho_m^b - \mathbf{r}_m^{bH}\mathbf{R}_m^{-1}\mathbf{r}_m^b$$

then (7.1.25) is proven directly

$$\frac{\det \mathbf{R}_{m+1}}{\det \mathbf{R}_m} = \alpha_m^b$$

7.2 Prove the matrix inversion lemma for lower right corner partitioned matrices, which is described by Equations (7.1.26) and (7.1.28).

Start with a slightly different partition of $\mathbf{R}_{m+1}\mathbf{Q}_{m+1}$ in (7.1.13)

$$\mathbf{R}_{m+1}\mathbf{Q}_{m+1} = \begin{bmatrix} \rho_m^f & \mathbf{r}_m^{fH} \\ \mathbf{r}_m^f & \mathbf{R}_m^f \end{bmatrix} \begin{bmatrix} q_m & \mathbf{q}_m^H \\ \mathbf{q}_m & \mathbf{Q}_m \end{bmatrix} = \begin{bmatrix} 1 & \mathbf{0}_m^H \\ \mathbf{0}_m & \mathbf{I}_m \end{bmatrix}$$

where the matrix multiplication equations (7.1.14) to (7.1.17) change to

$$\rho_m^f q_m + \mathbf{r}_m^{fH} \mathbf{q}_m = 1 \quad (1)$$

$$\rho_m^f \mathbf{q}_m^H + \mathbf{r}_m^{fH} \mathbf{Q}_m = \mathbf{0}_m^H \quad (2)$$

$$\mathbf{r}_m^f q_m + \mathbf{R}_m^f \mathbf{q}_m = \mathbf{0}_m$$

$$\mathbf{r}_m^f \mathbf{q}_m^H + \mathbf{R}_m^f \mathbf{Q}_m = \mathbf{I}_m \quad (2)$$

Assuming matrix \mathbf{R}_m is invertible

$$\mathbf{q}_m = -(\mathbf{R}_m^f)^{-1} \mathbf{r}_m^f q_m \quad (3)$$

Substitute into (1) above to obtain q_m

$$q_m = \frac{1}{\rho_m^f - \mathbf{r}_m^{fH} (\mathbf{R}_m^f)^{-1} \mathbf{r}_m^f} \quad (4)$$

Substitute (3) into (2)

$$\begin{aligned} \mathbf{q}_m &= \frac{- (\mathbf{R}_m^f)^{-1} \mathbf{r}_m^f}{\rho_m^f - \mathbf{r}_m^{fH} (\mathbf{R}_m^f)^{-1} \mathbf{r}_m^f} \\ &= \frac{\mathbf{a}_m}{\alpha_m^f} \end{aligned} \quad (5)$$

where \mathbf{a}_m and α_m^f are defined per (7.1.27) and (7.1.28). Now, (4) into (5) yields

$$\begin{aligned} \mathbf{Q}_m &= (\mathbf{R}_m^f)^{-1} (\mathbf{I}_m - \mathbf{r}_m^f \mathbf{q}_m^H) \\ &= (\mathbf{R}_m^f)^{-1} \left(\mathbf{I}_m + \frac{\mathbf{r}_m^f \left((\mathbf{R}_m^f)^{-1} \mathbf{r}_m^f \right)^H}{\rho_m^f - \mathbf{r}_m^{fH} (\mathbf{R}_m^f)^{-1} \mathbf{r}_m^f} \right) \end{aligned}$$

Therefore, using (3), (4), and (6)

$$\begin{aligned} \mathbf{Q}_{m+1} = \mathbf{R}_{m+1}^{-1} &= \begin{bmatrix} q_m & \mathbf{q}_m^H \\ \mathbf{q}_m & \mathbf{Q}_m \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{\alpha_m^f} & \frac{\mathbf{a}_m^H}{\alpha_m^f} \\ \frac{\mathbf{a}_m}{\alpha_m^f} & \mathbf{R}_m^{-f} + \frac{(\mathbf{R}_m^{-f} \mathbf{r}_m^f)(\mathbf{R}_m^{-f} \mathbf{r}_m^f)^H}{\rho_m^f - \mathbf{r}_m^{fH} (\mathbf{R}_m^f)^{-1} \mathbf{r}_m^f} \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{\alpha_m^f} & \frac{\mathbf{a}_m^H}{\alpha_m^f} \\ \frac{\mathbf{a}_m}{\alpha_m^f} & \mathbf{R}_m^{-f} + \frac{(\mathbf{a}_m)(\mathbf{a}_m^H)}{\alpha_m^f} \end{bmatrix} \\ &= \begin{bmatrix} 1 & \mathbf{0}_m^H \\ \mathbf{0}_m & \mathbf{R}_m^{-f} \end{bmatrix} + \frac{1}{\alpha_m^f} \begin{bmatrix} 1 \\ \mathbf{a}_m \end{bmatrix} [1 \quad \mathbf{a}_m^H] \end{aligned}$$

7.3 This problem generalizes the matrix inversion lemmas to non-symmetric matrices.

- (a) Show that if R^{-1} exists, the inverse of an upper left corner partitioned matrix is given by

$$\begin{bmatrix} R & r \\ \tilde{r}^T & \sigma \end{bmatrix}^{-1} = \frac{1}{\alpha} \begin{bmatrix} \alpha R^{-1} + wv^T & w \\ v^T & 1 \end{bmatrix}$$

where

$$\begin{aligned} Rw &= -r \\ R^T v &= -\tilde{r} \\ \alpha &= \sigma - \tilde{r}^T R^{-1} r = \sigma + v^T r = \sigma + \tilde{r}^T w \end{aligned}$$

Start with the partition of $\mathbf{R}\mathbf{Q} = \mathbf{I}$ in (7.1.13)

$$\begin{bmatrix} \mathbf{R} & \mathbf{r} \\ \tilde{\mathbf{r}}^T & \sigma \end{bmatrix} \begin{bmatrix} Q & \mathbf{q} \\ \tilde{\mathbf{q}}^T & q \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}$$

and computing the product results in the following four equations

$$\mathbf{R}\mathbf{Q} + \mathbf{r}\tilde{\mathbf{q}}^T = \mathbf{I} \quad (1)$$

$$\mathbf{R}\mathbf{q} + \mathbf{r}q = \mathbf{0} \quad (2)$$

$$\tilde{\mathbf{r}}^T\mathbf{Q} + \sigma\tilde{\mathbf{q}}^T = \mathbf{0}^T \quad (3)$$

$$\tilde{\mathbf{r}}^T\mathbf{q} + \sigma q = 1 \quad (4)$$

Assuming matrix \mathbf{R}_m is invertible

$$\mathbf{q} = -\mathbf{R}^{-1}\mathbf{r}q \quad (5)$$

Substitute (5) into (4) to obtain q

$$q = \frac{1}{\sigma - \tilde{\mathbf{r}}^H\mathbf{R}^{-1}\mathbf{r}} = \frac{1}{\alpha} \quad (6)$$

where $\alpha = \sigma - \tilde{\mathbf{r}}^H\mathbf{R}^{-1}\mathbf{r}$.

Substitute (6) into (5) to obtain \mathbf{q}

$$\mathbf{q} = \frac{-\mathbf{R}^{-1}\mathbf{r}}{\sigma - \tilde{\mathbf{r}}^H\mathbf{R}^{-1}\mathbf{r}} = \frac{-\mathbf{R}^{-1}\mathbf{r}}{\alpha} \quad (7)$$

Solve (1) for \mathbf{Q} and substitute into (3) to obtain $\tilde{\mathbf{q}}^T$

$$\tilde{\mathbf{q}}^T = \frac{-\tilde{\mathbf{r}}^T\mathbf{R}^{-1}}{\sigma - \tilde{\mathbf{r}}^H\mathbf{R}^{-1}\mathbf{r}} = \frac{-\tilde{\mathbf{r}}^T\mathbf{R}^{-1}}{\alpha} \quad (8)$$

Lastly, substitute (9) into (8) to obtain \mathbf{Q}

$$\mathbf{Q} = \frac{\alpha\mathbf{R}^{-1} + (\mathbf{R}^{-1}\mathbf{r})(\mathbf{R}^{-1}\tilde{\mathbf{r}})^T}{\alpha} \quad (9)$$

Therefore, \mathbf{R}_{m+1}^{-1} is

$$\begin{aligned} \mathbf{R}_{m+1}^{-1} &= \begin{bmatrix} R & r \\ \tilde{r}^T & \sigma \end{bmatrix}^{-1} = \begin{bmatrix} Q & \mathbf{q} \\ \tilde{\mathbf{q}}^T & q \end{bmatrix} \\ &= \frac{1}{\alpha} \begin{bmatrix} \alpha R^{-1} + wv^T & w \\ v^T & 1 \end{bmatrix} \end{aligned}$$

where

$$\mathbf{R}w = -\mathbf{r}$$

$$\mathbf{R}^T v = -\tilde{\mathbf{r}}$$

$$\alpha = \sigma - \tilde{\mathbf{r}}^T\mathbf{R}^{-1}\mathbf{r} = \sigma + v^T\mathbf{r} = \sigma + \tilde{\mathbf{r}}^T w$$

(b) Show that if R^{-1} exists, the inverse of an upper left corner partitioned matrix is given by

$$\begin{bmatrix} \sigma & \tilde{\mathbf{r}}^T \\ \mathbf{r} & \mathbf{R} \end{bmatrix}^{-1} = \frac{1}{\alpha} \begin{bmatrix} 1 & v^T \\ w & \alpha R^{-1} + wv^T \end{bmatrix}$$

where

$$\begin{aligned} \mathbf{R}w &= -\mathbf{r} \\ \mathbf{R}^T v &= -\tilde{\mathbf{r}} \\ \alpha &= \sigma - \tilde{\mathbf{r}}^T \mathbf{R}^{-1} \mathbf{r} = \sigma + v^T \mathbf{r} = \sigma + \tilde{\mathbf{r}}^T w \end{aligned}$$

Start with the partition of $\mathbf{RQ} = \mathbf{I}$ in (7.1.13)

$$\begin{bmatrix} \sigma & \tilde{\mathbf{r}}^T \\ \mathbf{r} & \mathbf{R} \end{bmatrix} \begin{bmatrix} q & \tilde{\mathbf{q}}^T \\ \mathbf{q} & \mathbf{Q} \end{bmatrix} = \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0}^T & \mathbf{I} \end{bmatrix}$$

and computing the product results in the following four equations

$$\sigma q + \tilde{\mathbf{r}}^T \mathbf{q} = 1 \quad (1)$$

$$\sigma \tilde{\mathbf{q}}^T + \tilde{\mathbf{r}}^T \mathbf{Q} = \mathbf{0}^T \quad (2)$$

$$\mathbf{r}q + \mathbf{R}\mathbf{q} = \mathbf{0} \quad (3)$$

$$\mathbf{RQ} + \mathbf{r}\tilde{\mathbf{q}}^T = \mathbf{I} \quad (4)$$

Assuming matrix \mathbf{R}_m is invertible

$$\mathbf{q} = -\mathbf{R}^{-1} \mathbf{r}q \quad (5)$$

Substitute (5) into (1) to obtain q

$$q = \frac{1}{\sigma - \tilde{\mathbf{r}}^H \mathbf{R}^{-1} \mathbf{r}} = \frac{1}{\alpha} \quad (6)$$

where $\alpha = \sigma - \tilde{\mathbf{r}}^H \mathbf{R}^{-1} \mathbf{r}$.

Substitute (6) into (5) to obtain \mathbf{q}

$$\mathbf{q} = \frac{-\mathbf{R}^{-1} \mathbf{r}}{\sigma - \tilde{\mathbf{r}}^H \mathbf{R}^{-1} \mathbf{r}} = \frac{-\mathbf{R}^{-1} \mathbf{r}}{\alpha} \quad (7)$$

Solve (4) for \mathbf{Q} and substitute into (2) to obtain $\tilde{\mathbf{q}}^T$

$$\tilde{\mathbf{q}}^T = \frac{-\tilde{\mathbf{r}}^T \mathbf{R}^{-1}}{\sigma - \tilde{\mathbf{r}}^H \mathbf{R}^{-1} \mathbf{r}} = \frac{-\tilde{\mathbf{r}}^T \mathbf{R}^{-1}}{\alpha} \quad (8)$$

Lastly, substitute (8) into (1) to obtain \mathbf{Q}

$$\mathbf{Q} = \frac{\alpha \mathbf{R}^{-1} + (\mathbf{R}^{-1} \mathbf{r})(\mathbf{R}^{-1} \tilde{\mathbf{r}})^T}{\alpha} \quad (9)$$

Therefore, \mathbf{R}_{m+1}^{-1} is

$$\begin{aligned}\mathbf{R}_{m+1}^{-1} &= \begin{bmatrix} \sigma & \tilde{\mathbf{r}}^T \\ \mathbf{r} & \mathbf{R} \end{bmatrix}^{-1} = \begin{bmatrix} q & \tilde{\mathbf{q}}^T \\ \mathbf{q} & \mathbf{Q} \end{bmatrix} \\ &= \frac{1}{\alpha} \begin{bmatrix} 1 & v^T \\ w & \alpha R^{-1} + wv^T \end{bmatrix}\end{aligned}$$

where

$$\begin{aligned}\mathbf{R}w &= -\mathbf{r} \\ \mathbf{R}^T v &= -\tilde{\mathbf{r}} \\ \alpha &= \sigma - \tilde{\mathbf{r}}^T \mathbf{R}^{-1} \mathbf{r} = \sigma + v^T \mathbf{r} = \sigma + \tilde{\mathbf{r}}^T w\end{aligned}$$

(c)

7.4 Develop an order-recursive algorithm to solve the linear system in Example 7.1.2, using the lower right corner partitioning lemma (7.1.26).

Using (7.1.26), and starting with $m = 1$, then $\mathbf{R}_1^f = 1$, $\mathbf{r}_1^f = 1/2$, and $\rho_1^f = 1$. This leads to

$$\mathbf{a}_1 = -(\mathbf{R}_1^f)^{-1} \mathbf{r}_1^f = -1/2$$

and

$$\alpha_1^f = \rho_1^f - \mathbf{r}_1^{fH} (\mathbf{R}_1^f)^{-1} \mathbf{r}_1^f = 3/4$$

Using these results into (7.1.26) to find \mathbf{R}_2^{-f}

$$\begin{aligned}\mathbf{R}_2^{-f} &= \begin{bmatrix} 1 & \mathbf{0}_1^H \\ \mathbf{0}_1 & \mathbf{R}_1^{-f} \end{bmatrix} + \frac{1}{\alpha_1^f} \begin{bmatrix} 1 \\ \mathbf{a}_1 \end{bmatrix} [1 \quad \mathbf{a}_1^H] \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \frac{4}{3} \begin{bmatrix} 1 \\ -\frac{1}{2} \end{bmatrix} [1 \quad -\frac{1}{2}] \\ &= \begin{bmatrix} \frac{4}{3} & -\frac{2}{3} \\ -\frac{2}{3} & \frac{4}{3} \end{bmatrix}\end{aligned}$$

Therefore

$$\begin{aligned}\mathbf{a}_2 &= -(\mathbf{R}_2^f)^{-1} \mathbf{r}_2^f \\ &= - \begin{bmatrix} \frac{4}{3} & -\frac{2}{3} \\ -\frac{2}{3} & \frac{4}{3} \end{bmatrix} \begin{bmatrix} \frac{1}{2} \\ \frac{1}{3} \end{bmatrix} \\ &= \begin{bmatrix} -\frac{4}{9} \\ -\frac{1}{9} \end{bmatrix}\end{aligned}$$

and

$$\begin{aligned}\alpha_2^f &= \rho_2^f - \mathbf{r}_2^{fH} (\mathbf{R}_2^f)^{-1} \mathbf{r}_2^f \\ &= \rho_2^f - \mathbf{r}_2^{fH} \mathbf{a}_2 \\ &= 1 + [\frac{1}{2} \quad \frac{1}{3}] \begin{bmatrix} -\frac{4}{9} \\ -\frac{1}{9} \end{bmatrix} \\ &= \frac{20}{27}\end{aligned}$$

Using (7.1.26) again

$$\begin{aligned}\mathbf{R}_3^{-f} &= \begin{bmatrix} 1 & \mathbf{0}_2^H \\ \mathbf{0}_2 & \mathbf{R}_2^{-f} \end{bmatrix} + \frac{1}{\alpha_2^f} \begin{bmatrix} 1 \\ \mathbf{a}_2 \end{bmatrix} [1 \quad \mathbf{a}_2^H] \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{4}{3} & -\frac{2}{3} \\ 0 & -\frac{2}{3} & \frac{4}{3} \end{bmatrix} + \frac{27}{20} \begin{bmatrix} 1 \\ -\frac{4}{9} \\ -\frac{1}{9} \end{bmatrix} [1 \quad -\frac{4}{9} \quad -\frac{1}{9}] \\ &= \begin{bmatrix} 1.35 & -0.6 & -0.15 \\ -0.6 & 1.6 & -0.6 \\ -0.15 & -0.6 & 1.35 \end{bmatrix}\end{aligned}$$

The desired solution is found using \mathbf{R}_3^{-f} and d_3

$$c_3 = \mathbf{R}_3^{-f} d_3 = \begin{bmatrix} -\frac{9}{20} \\ \frac{4}{20} \\ \frac{81}{20} \end{bmatrix}$$

7.5 In this problem we consider two different approaches for inversion of symmetric and positive definite matrices by constructing an arbitrary fourth-order positive definite correlation matrix \mathbf{R} and comparing their computational complexities.

- (a) Given that the inverse of a lower (upper) triangular matrix is itself lower (upper) triangular, develop an algorithm for triangular matrix inversion.

The Cholesky decomposition of the autocorrelation matrix is $R = LDL^H$ where L is lower triangular and D is a diagonal matrix. Therefore the inverse of R is

$$R^{-1} = (LDL^H)^{-1} = L^{-H} D^{-1} L^{-1}$$

The computational cost of L^{-1} is $O(M^3)$.

- (b) Compute the inverse of \mathbf{R} , using the algorithm in part(a) and Equation (7.1.58).

We can build up the lower triangular inverse in a recursive manner from (7.1.37)

$$L_{m+1} = \begin{bmatrix} L_m & 0 \\ l_m^H & 1 \end{bmatrix}$$

where the inverse of L_{m+1} is

$$L_{m+1}^{-1} = \begin{bmatrix} L_m & 0 \\ l_m^H & 1 \end{bmatrix}^{-1} = \begin{bmatrix} L_m^{-1} & 0 \\ v_m^T & 1 \end{bmatrix}$$

where $v_m^T = L_m^{-H} l_m$. This allows the order recursive build up of L_{m+1}^{-1} . The computational cost is $O(M^3)$.

(c) Build up the inverse of \mathbf{R} , using the recursion (7.1.24).

Using (7.1.22), (7.1.23) and (7.1.24), the recursion for the computation of the inverse of \mathbf{R} is

$$\begin{aligned}\mathbf{R}_{m+1}^{-1} &= \begin{bmatrix} \mathbf{R}_m & \mathbf{r}_m^b \\ \mathbf{r}_m^{bH} & \rho_m^b \end{bmatrix}^{-1} \\ &= \begin{bmatrix} \mathbf{R}_m^{-1} & \mathbf{0}_m \\ \mathbf{0}_m^H & 0 \end{bmatrix} = \frac{1}{\alpha_m^b} \begin{bmatrix} b_m \\ 1 \end{bmatrix} [b_m^H \quad 1]\end{aligned}$$

where $b_m = -\mathbf{R}_m^{-1} \mathbf{r}_m^b$ and $\alpha_m^b = \rho_m^b + \mathbf{r}_m^{bH} b_m$

(d) Estimate the number of operations for each method as a function of order M , and check their validity for $M = 4$, using MATLAB.

7.6 Using the appropriate orthogonality principles and definitions, prove Equation (7.3.32)

The partial correlation $E\{e_m^b(n-1)e_m^{f*}(n)\}$ can be expanded using (7.3.39) as

$$E\{e_m^b(n-1)(x^*(n) + x_m^*(n-1)a_m^H(n))\} = E\{e_m^b(n-1)x^*(n)\}E\{e_m^b(n-1)x_m^*(n-1)a_m^H(n)\}$$

Using the fact that x_m is orthogonal to e_m , the above equation can be simplified to

$$E\{e_m^b(n-1)x^*(n)\} = E\{(b_m^H(n-1)x_m(n-1) + x(n-M-1))x^*(n)\}$$

again using (7.3.39). The final step uses (7.3.2) and (7.3.26) to finish the proof

$$\begin{aligned}E\{(b_m^H(n-1)x_m(n-1) + x(n-M-1))x^*(n)\} &= E\left\{[b_m^H(n-1) \quad 1] \begin{bmatrix} x_m(n-1) \\ x(n-M-1) \end{bmatrix} x^*(n)\right\} \\ &= [b_m^H(n-1) \quad 1] E\left\{\begin{bmatrix} x_m(n-1) \\ x(n-M-1) \end{bmatrix} x^*(n)\right\} \\ &= [b_m^H(n-1) \quad 1] E\{x_{m+1}(n-1)x^*(n)\} \\ &= [b_m^H(n-1) \quad 1] \begin{bmatrix} r_m^f(n) \\ r_{m+1}^f(n) \end{bmatrix}\end{aligned}$$

Therefore, $E\{e_m^b(n-1)e_m^{f*}(n)\} = r_{m+1}^f(n) + b_m^H(n-1)r_m^f(n)$

7.7 Prove Equations (7.3.36) to (7.3.38), using Equation (7.1.45)

$$\begin{aligned}P_{m+1}^f(n) &= P_m^f(n) + r_{m+1}^{fH}(n)a_{m+1}(n) \\ &= P_m^f(n) - r_{m+1}^{fH}(n)R_{m+1}^{-1}(n-1)r_{m+1}^f(n) \\ &= P_m^f(n) - r_{m+1}^{fH}(n)L_{m+1}^{-1}(n-1)D_{m+1}^{-1}(n-1)L_{m+1}^{-1}(n-1)r_{m+1}^f(n) \\ &= P_m^f(n) - r_{m+1}^{fH}(n)L_{m+1}^{-1}(n-1)D_{m+1}^{-1}(n-1)D_{m+1}(n-1)D_{m+1}^{-1}(n-1)L_{m+1}^{-1}(n-1)r_{m+1}^f(n) \\ &= P_m^f(n) - k_{m+1}^b(n)D_{m+1}(n-1)k_{m+1}^b(n) \\ &= P_m^f(n) - D_{m+1}(n-1)|k_{m+1}^b(n)|^2 \\ &= ???\end{aligned}$$

7.8 Working as in Example 6.3.1, develop an algorithm for the upper-lower decomposition of a symmetric positive definite matrix. Then use it to factorize the matrix in Example 6.3.1, and verify your results, using the function [U,D]=udut(R).

Writing the decomposition directly for $M=4$, we have

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \\ r_{41} & r_{42} & r_{43} & r_{44} \end{bmatrix} = \begin{bmatrix} 1 & u_{10} & u_{20} & u_{30} \\ 0 & 1 & u_{21} & u_{31} \\ 0 & 0 & 1 & u_{32} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \xi_1 & 0 & 0 & 0 \\ 0 & \xi_2 & 0 & 0 \\ 0 & 0 & \xi_3 & 0 \\ 0 & 0 & 0 & \xi_4 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ u_{10}^* & 1 & 0 & 0 \\ u_{20}^* & u_{21}^* & 1 & 0 \\ u_{30}^* & u_{31}^* & u_{32}^* & 1 \end{bmatrix}$$

where $r_{ij} = r_{ji}$ and $\xi_i > 0$. If we perform the matrix multiplication on the right-hand side of (6.3.7) and equate the matrix elements on the left and right sides, we obtain,

$$\begin{aligned} r_{11} = \xi_1 &\Rightarrow \xi_1 = r_{11} \\ r_{21} = \xi_1 u_{10} &\Rightarrow u_{10} = \frac{r_{21}}{\xi_1} \\ r_{22} = \xi_1 |u_{10}|^2 + \xi_2 &\Rightarrow \xi_2 = r_{22} - \xi_1 |u_{10}|^2 \\ r_{31} = \xi_1 u_{20} &\Rightarrow u_{20} = \frac{r_{31}}{\xi_1} \\ r_{32} = \xi_1 u_{20} u_{10}^* + \xi_2 u_{21} &\Rightarrow u_{21} = \frac{r_{32} - \xi_1 u_{20} u_{10}^*}{\xi_2} \\ r_{33} = \xi_1 |u_{20}|^2 + \xi_2 |u_{21}|^2 + \xi_3 &\Rightarrow \xi_3 = r_{33} - \xi_1 |u_{20}|^2 - \xi_2 |u_{21}|^2 \\ r_{41} = \xi_1 u_{30} &\Rightarrow u_{30} = \frac{r_{41}}{\xi_1} \\ r_{42} = \xi_1 u_{30} u_{10}^* + \xi_2 u_{31} &\Rightarrow u_{31} = \frac{r_{42} - \xi_1 u_{30} u_{10}^*}{\xi_2} \\ r_{43} = \xi_1 u_{30} u_{20}^* + \xi_2 u_{31} u_{21}^* + \xi_3 u_{32} &\Rightarrow u_{32} = \frac{r_{43} - \xi_1 u_{30} u_{20}^* - \xi_2 u_{31} u_{21}^*}{\xi_3} \\ r_{44} = \xi_1 |u_{30}|^2 + \xi_2 |u_{31}|^2 + \xi_3 |u_{32}|^2 + \xi_4 &\Rightarrow \xi_4 = r_{44} - \xi_1 |u_{30}|^2 - \xi_2 |u_{31}|^2 - \xi_3 |u_{32}|^2 \end{aligned}$$

This provides a row-by-row computation of the elements of the $U^H D U$ decomposition. Note that the computation of the next row does not change the already computed rows. This recursion can be continued for higher order matrices.

7.9

- 7.9** In this problem we explore the meaning of the various quantities in the decomposition $\mathbf{R} = \mathbf{U}\bar{\mathbf{D}}\mathbf{U}^H$ of the correlation matrix.
- Show that the rows of $\mathbf{A} = \mathbf{U}^{-1}$ are the MMSE estimator of x_m from $x_{m+1}, x_{m+2}, \dots, x_M$.
 - Show that the decomposition $\mathbf{R} = \mathbf{U}\bar{\mathbf{D}}\mathbf{U}^H$ can be obtained by the Gram-Schmidt orthogonalization process, starting with the random variable x_M and ending with x_1 , that is, proceeding backward.

- 7.10** In this problem we clarify the various quantities and the form of the partitionings involved in the UDU^H decomposition, using an $m = 4$ correlation matrix.

- Prove that the components of the forward prediction error vector (7.3.65) are uncorrelated.

The forward prediction error vector is uncorrelated if its autocorrelation is a diagonal matrix. Using (7.3.66)

$$e(n) = A(n)x(n)$$

with an autocorrelation of

$$R_e = AE\{xx^H\}A^H = AR_xA^H = A(U\bar{D}U^H)A^H$$

where the UDU^H is the Gram-Schmidt decomposition. From table 7.1, $A = U^{-1}$ and R_e is

$$R_e = A(U\bar{D}U^H)A^H = U^{-1}(U\bar{D}U^H)U^{-H} = \bar{D}$$

which a diagonal matrix.

- (b) Writing explicitly the matrix R , identify and express the quantities in Equations (7.3.62) through (7.3.67)

Using (7.3.62) the order-recursion of R for $m = 4$ is

$$R_1(n-3) \rightarrow R_2(n-2) \rightarrow R_3(n-1) \rightarrow R_4(n)$$

where $R_1(n-3) = P_x(n-3)$ and

$$\begin{aligned} R_2(n-2) &= \begin{bmatrix} P_x(n-2) & r_1^{fH}(n-2) \\ r_1^f(n-2) & R_1(n-3) \end{bmatrix} \\ R_3(n-1) &= \begin{bmatrix} P_x(n-1) & r_2^{fH}(n-1) \\ r_2^f(n-1) & R_2(n-2) \end{bmatrix} \\ R_4(n) &= \begin{bmatrix} P_x(n) & r_3^{fH}(n) \\ r_3^f(n) & R_3(n-1) \end{bmatrix} \end{aligned}$$

Using (7.3.63), the order-recursion for the FLP coefficients for $m = 4$ is

$$a_1(n-3) \rightarrow a_2(n-2) \rightarrow a_3(n-1) \rightarrow a_4(n)$$

where $a_1(n-3) = a_1^{(1)}(n-3)$ and

$$\begin{aligned} a_2(n-2) &= \begin{bmatrix} a_1^{(2)}(n-2) \\ a_2^{(2)}(n-2) \end{bmatrix} \\ a_3(n-1) &= \begin{bmatrix} a_1^{(3)}(n-1) \\ a_2^{(3)}(n-1) \\ a_3^{(3)}(n-1) \end{bmatrix} \\ a_4(n) &= \begin{bmatrix} a_1^{(4)}(n) \\ a_2^{(4)}(n) \\ a_3^{(4)}(n) \\ a_4^{(4)}(n) \end{bmatrix} \end{aligned}$$

Using (7.3.64) and (7.3.65) the order-recursion for the FLP errors for $m = 4$ is

$$e_1^f(n-3) \rightarrow e_2^f(n-2) \rightarrow e_3^f(n-1) \rightarrow e_4^f(n)$$

and the FLP error vector is

$$\begin{aligned} e_1^f(n-3) &= e_1^f(n-3) \\ e_2^f(n-2) &= \begin{bmatrix} e_2^f(n-2) \\ e_1^f(n-3) \end{bmatrix} \\ e_3^f(n-1) &= \begin{bmatrix} e_3^f(n-1) \\ e_2^f(n-2) \\ e_3^f(n-3) \end{bmatrix} \\ e_4(n) &= \begin{bmatrix} e_4^f(n) \\ e_3^f(n-1) \\ e_2^f(n-2) \\ e_1^f(n-3) \end{bmatrix} \end{aligned}$$

(c)

- 7.11** Given an all-zero lattice filter with coefficients k_0 and k_1 , determine the MSE $P(k_0, k_1)$ as a function of the required second-order moments, assumed jointly stationary, and plot the error performance surface. Use the statistics in Example 6.2.1

For an all-zero lattice filter, the estimate $\hat{y}(n)$ is

$$\hat{y}(n) = x(n) + (k_0^* + k_0 k_1^*)x(n-1) + k_1^*x(n-2)$$

where $x(n)$ is the input to the lattice. The MSE $P(k_0, k_1)$ is

$$P(k_0, k_1) = P_y - 2d_1c_1 - 2d_2c_2 + r_{11}c_1^2 + 2r_{12}c_1c_2 + r_{22}c_2^2$$

where $c_1 = k_0^* + k_0 k_1^*$ and $c_2 = k_1^*$. Using the statistics from Example 6.2.1 ($P_y = 0.5$, $r_{11} = r_{22} = 4.5$, $r_{12} = r_{21} = -0.1545$, $d_1 = -0.5$, and $d_2 = -0.1545$), the MSE $P(k_0, k_1)$ is

$$P(k_0, k_1) = 0.5 + k_0 + k_0 k_1 + (0.3090)k_1 + 4.5(k_0 + k_0 k_1)^2 - (0.3090)(k_0 + k_0 k_1)(k_1) + 4.5(k_1)^2$$

- 7.12** Given the autocorrelation $r(0) = 1$, $r(1) = r(2) = \frac{1}{2}$, and $r(3) = \frac{1}{4}$, determine all possible representations for the third-order prediction error filter (see Figure 7.7)

Given the autocorrelation, we need to find the forward predictor coefficients \mathbf{a}_m and the lattice parameters k_m . Using Table 7.2, step (2a)

$$P_0 = r(0) = 1, \quad \beta_0 = r^*(1) = 1/2$$

and step (2b)

$$k_0 = -\frac{r(1)}{r(0)}, \quad a_1^{(1)} = k_0$$

Using step (3), for $m = 1$

$$\begin{aligned}
 (a) \quad P_1 &= P_0 + \beta_0 k_0 = 1 + (\frac{1}{2})(-\frac{1}{2}) = 1/4 \\
 (b) \quad \mathbf{r}_1 &= r(1) = 1/2 \\
 (c) \quad \beta_1 &= \mathbf{a}_1^T J \mathbf{r}_1^* + r^*(2) = (1/2)(-1/2) + (1/2) = 1/4 \\
 (d) \quad k_1 &= -\frac{\beta_1}{P_1} = \frac{-1/4}{3/4} = -1/3 \\
 (e) \quad \mathbf{a}_2 &= \begin{bmatrix} \mathbf{a}_1 \\ 0 \end{bmatrix} + \begin{bmatrix} J \mathbf{a}_1^* \\ 1 \end{bmatrix} k_1 \\
 &= \begin{bmatrix} -1/2 \\ 0 \end{bmatrix} + \begin{bmatrix} -1/2 \\ 1 \end{bmatrix} (-1/3) = \begin{bmatrix} -1/3 \\ -1/3 \end{bmatrix}
 \end{aligned}$$

Repeating step (3) for $m = 2$

$$\begin{aligned}
 (a) \quad P_2 &= P_1 + \beta_1 k_1 = 3/4 + (\frac{1}{4})(-\frac{1}{3}) = 2/3 \\
 (b) \quad \mathbf{r}_2 &= [r(1)r(2)]^T = [1/2 \ 1/2]^T \\
 (c) \quad \beta_2 &= \mathbf{a}_2^T J \mathbf{r}_2^* + r^*(3) = [-1/3 \ -1/3] \begin{bmatrix} -1/3 \\ -1/3 \end{bmatrix} + (1/4) = -1/2 \\
 (d) \quad k_2 &= -\frac{\beta_2}{P_2} = \frac{1/2}{2/3} = 3/4 \\
 (e) \quad \mathbf{a}_3 &= \begin{bmatrix} \mathbf{a}_2 \\ 0 \end{bmatrix} + \begin{bmatrix} J \mathbf{a}_2^* \\ 1 \end{bmatrix} k_2 \\
 &= \begin{bmatrix} -1/3 \\ -1/3 \end{bmatrix} + \begin{bmatrix} -1/3 \\ -1/3 \end{bmatrix} (3/4) = \begin{bmatrix} -7/12 \\ -7/12 \end{bmatrix}
 \end{aligned}$$

Lastly, using step (4)

$$P_3 = P_2 + \beta_2 k_2 = 2/3 - (1/2)(3/4) = 7/24$$

This can be checked using (7.4.30)

$$P_3 = r(0) \prod_{m=1}^3 (1 - |k_{m-1}|^2)$$

Therefore, the forward predictor coefficients \mathbf{a}_m and the lattice parameters k_m are

$$\mathbf{a}_m = \begin{bmatrix} -7/12 \\ -7/12 \\ 3/4 \end{bmatrix}$$

and

$$k_0 = -1/2 \quad k_1 = -1/3 \quad k_2 = 3/4$$

7.13 Repeat Problem 7.12 for $k_0 = k_1 = k_2 = 1/3$ and $P_3 = (2/3)^3$.

Given the lattice coefficients, we need to find the autocorrelation \mathbf{r}_m and forward predictor coefficients \mathbf{a}_m . The lag zero autocorrelation coefficient can be found using (7.4.30),

$$r(0) = \frac{P_3}{\prod_{m=1}^3 (1 - |k_{m-1}|^2)} = 27/64$$

Using (7.5.28) we can find P_m

$$\begin{aligned} P_m &= P_{m-1}(1 - |k_{m-1}|^2) \\ P_1 &= P_0(1 - |k_0|^2) \\ &= (27/64)(1 - |1/3|)^2 = 3/8 \\ P_2 &= P_1(1 - |k_1|^2) \\ &= (3/8)(1 - |1/3|^2) = 1/3 \end{aligned}$$

The autocorrelation coefficients can be found using (7.5.36)

$$\begin{aligned} r(m+1) &= -k_m^* P_m - \mathbf{a}_m^H J \mathbf{r}_m \\ r(1) &= -k_0^* P_0 \\ &= -(1/3)(27/64) = -9/64 \\ r(2) &= -k_1^* P_1 - \mathbf{a}_1^H \mathbf{r}_1 \\ &= (-1/3)(3/8) - (1/3)(-9/64) = -5/64 \\ r(3) &= -k_2^* P_2 - \mathbf{a}_2^H J \mathbf{r}_2 \\ &= -k_2^* P_2 - [a_1^{(2)} r(2) + a_2^{(2)} r(1)] \\ &= (-1/3)(1/3) + (4/9)(5/64) + (1/3)(9/64) = 17/576 \end{aligned}$$

where $a_1^{(1)} = k_0 = 1/3$, $a_2^{(2)} = k_1 = 1/3$, and $a_1^{(2)} = k_0 + k_0^* k_1 = 4/9$.

The forward predictor coefficients \mathbf{a}_m can be found using (7.5.27)

$$\begin{aligned} \mathbf{a}_m &= \begin{bmatrix} \mathbf{a}_{m-1} \\ 0 \end{bmatrix} + \begin{bmatrix} J \mathbf{a}_{m-1}^* \\ 1 \end{bmatrix} k_{m-1} \\ \mathbf{a}_1 &= k_1 = 1/3 \\ \mathbf{a}_2 &= \begin{bmatrix} 1/3 \\ 0 \end{bmatrix} + \begin{bmatrix} 1/3 \\ 1 \end{bmatrix} (1/3) = \begin{bmatrix} 4/9 \\ 1/3 \end{bmatrix} \\ \mathbf{a}_3 &= \begin{bmatrix} 4/9 \\ 1/3 \end{bmatrix} + \begin{bmatrix} 1/3 \\ 4/9 \end{bmatrix} (1/3) = \begin{bmatrix} 5/9 \\ 13/27 \end{bmatrix} \end{aligned}$$

7.14 Use Levinson's algorithm to solve the normal equations $Rc = d$ where $R = \text{Toeplitz}\{3, 2, 1\}$ and $d = [6 \ 6 \ 2]^T$.

Using the Levinson algorithm (Table 7.3), starting with step 2

- (a) $P_0 = r(0) = 3, \beta_0 = r^*(1) = 2$
- (b) $k_0 = -\beta_0/P_0 = -2/3, a_1^{(1)} = k_0 = -2/3$
- (c) $\beta_0^c = d_1 = 6$
- (d) $k_0^c = -\beta_0^c/P_0 = -6/3 = -2, c_1^{(1)} = k_0^c = -2$

Solving step 3, with $m = 1$ yields the following

- (a) $r_1 = [r(1)]$
- (b) $\beta_1 = a_1^T J r_1^* + r(2) = (-2/3)(2) + 1 = -1/3$
- (c) $P_1 = P_0 + \beta_0 k_0^* = 3 + 2(-2/3) = 5/9$
- (d) $k_1 = -\beta_1/P_1 = \frac{1/3}{5/9} = 3/5$
- (e) $a_2 = \begin{bmatrix} a_1 \\ 0 \end{bmatrix} + \begin{bmatrix} a_1 \\ 1 \end{bmatrix} k_1 = \begin{bmatrix} -2/3 \\ 0 \end{bmatrix} + \begin{bmatrix} -2/3 \\ 1 \end{bmatrix} (3/5) = \begin{bmatrix} -16/15 \\ 3/5 \end{bmatrix}$
- (f) $\beta_1^c = -c_1^{(1)} r(1) + d_2 = (2)(2) + 6 = 10$
- (g) $k_1^c = \beta_1^c/P_1 = \frac{10}{5/9} = 18$
- (h) $c_2 = \begin{bmatrix} c_1 \\ 0 \end{bmatrix} + \begin{bmatrix} Ja_1 \\ 1 \end{bmatrix} k_1^c = \begin{bmatrix} -2 \\ 0 \end{bmatrix} + \begin{bmatrix} -2/3 \\ 1 \end{bmatrix} (18) = \begin{bmatrix} -14 \\ 18 \end{bmatrix}$

The last recursion for $m = 2$ repeats step 3 only for the optimum filter

- (f) $\beta_2^c = -c_2^H J r(2) + d_3 = -[-14 \quad 18] \begin{bmatrix} 2 \\ 1 \end{bmatrix} + 2 = 12$
- (g) $k_2^c = \beta_2^c/P_2 = \frac{\beta_2^c}{P_1 + \beta_1 k_1} = 33.75$
- (h) $c_3 = \begin{bmatrix} c_2 \\ 0 \end{bmatrix} + \begin{bmatrix} Ja_2 \\ 1 \end{bmatrix} k_2^c = \begin{bmatrix} -14 \\ 18 \\ 0 \end{bmatrix} + \begin{bmatrix} 3/5 \\ -16/15 \\ 1 \end{bmatrix} (33.75) = \begin{bmatrix} 6.25 \\ -18.00 \\ 33.75 \end{bmatrix}$

Therefore, the optimum filter is $c_3 = [6.25 \quad -18.0 \quad 33.75]^T$

7.15 Consider a random sequence with autocorrelation $\{r(L)\}_0^3 = \{1, 0.8, 0.6, 0.4\}$

- (a) Determine the FLP \mathbf{a}_m and the corresponding error P_m^f for $m = 1, 2, 3$.

Given the autocorrelation coefficients, the FLP and the corresponding errors can be found using Levinson-Durbin algorithm (Table 7.2). Using Table 7.2, step (2a)

$$P_0 = r(0) = 1, \quad \beta_0 = r^*(1) = 0.8$$

and step (2b)

$$k_0 = -\frac{r(1)}{r(0)} = -0.8, \quad a_1^{(1)} = k_0 = -0.8$$

Using step (3), for $m = 1$

$$\begin{aligned}
 (a) \quad P_1 &= P_0 + \beta_0 k_0 = 1 - (0.8)^2 = 0.36 \\
 (b) \quad \mathbf{r}_1 &= r(1) = 0.8 \\
 (c) \quad \beta_1 &= \mathbf{a}_1^T J \mathbf{r}_1^* + r^*(2) = (-0.8)(0.8) + (0.6) = -0.04 \\
 (d) \quad k_1 &= -\frac{\beta_1}{P_1} = \frac{0.04}{0.36} = 0.111 \\
 (e) \quad \mathbf{a}_2 &= \begin{bmatrix} \mathbf{a}_1 \\ 0 \end{bmatrix} + \begin{bmatrix} J \mathbf{a}_1^* \\ 1 \end{bmatrix} k_1 \\
 &= \begin{bmatrix} -0.8 \\ 0 \end{bmatrix} + \begin{bmatrix} -0.8 \\ 1 \end{bmatrix} (-0.111) = \begin{bmatrix} -0.889 \\ 0.111 \end{bmatrix}
 \end{aligned}$$

Repeating step (3) for $m = 2$

$$\begin{aligned}
 (a) \quad P_2 &= P_1 + \beta_1 k_1 = 0.36 - (0.04)(0.111) = 0.356 \\
 (b) \quad \mathbf{r}_2 &= [r(1)r(2)]^T = [0.8 \ 0.6]^T \\
 (c) \quad \beta_2 &= \mathbf{a}_2^T J \mathbf{r}_2^* + r^*(3) = [-0.889 \ 0.111] \begin{bmatrix} 0.6 \\ 0.8 \end{bmatrix} + (0.4) = -0.044 \\
 (d) \quad k_2 &= -\frac{\beta_2}{P_2} = \frac{0.044}{0.356} = 0.1248 \\
 (e) \quad \mathbf{a}_3 &= \begin{bmatrix} \mathbf{a}_2 \\ 0 \end{bmatrix} + \begin{bmatrix} J \mathbf{a}_2^* \\ 1 \end{bmatrix} k_2 \\
 &= \begin{bmatrix} -0.889 \\ 0.111 \\ 0 \end{bmatrix} + \begin{bmatrix} 0.111 \\ -0.889 \\ 1 \end{bmatrix} (0.1248) = \begin{bmatrix} -0.8765 \\ 0.0123 \\ 0.1111 \end{bmatrix}
 \end{aligned}$$

Lastly, using step (4)

$$P_3 = P_2 + \beta_2 k_2 = 0.356 - (0.044)(0.1248) = 0.35$$

This can be checked using (7.4.30)

$$P_3 = r(0) \prod_{m=1}^3 (1 - |k_{m-1}|^2)$$

- (b) Determine and draw the flow diagram of the third-order lattice prediction error filter.

Figure 7.15 shows the third-order lattice prediction error filter where the lattice parameters are $k_0 = -0.8$, $k_1 = 0.1111$, and $k_2 = 0.1248$ which have been computed above in part (a).

- 7.16** Using the Levinson-Durbin algorithm, determine the third-order linear predictor \mathbf{a}_3 and the MMSE P_3 for the signal with autocorrelation $r(0) = 1, r(1) = r(2) = 1/2$, and $r(3) = 1/4$.

Given the autocorrelation coefficients, the FLP and the corresponding error can be found using Levinson-Durbin algorithm (Table 7.2). Using Table 7.2, step (2a)

$$P_0 = r(0) = 1, \quad \beta_0 = r^*(1) = 1/2$$

and step (2b)

$$k_0 = -\frac{r(1)}{r(0)} = -1/2, \quad a_1^{(1)} = k_0 = -1/2$$

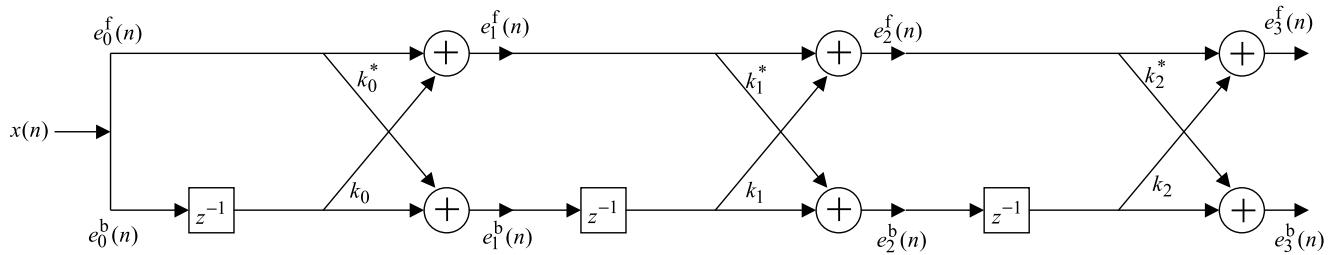


Figure 7.15: Third-order lattice prediction error filter

Using step (3), for $m = 1$

$$\begin{aligned}
 (a) \quad P_1 &= P_0 + \beta_0 k_0 = 1 - (1/2)(1/2) = 3/4 \\
 (b) \quad \mathbf{r}_1 &= \mathbf{r}(1) = 1/2 \\
 (c) \quad \beta_1 &= \mathbf{a}_1^T \mathbf{J} \mathbf{r}_1^* + r^*(2) = (-1/2)(1/2) + (1/2) = 1/4 \\
 (d) \quad k_1 &= -\frac{\beta_1}{P_1} = \frac{-1/4}{3/4} = -1/3 \\
 (e) \quad \mathbf{a}_2 &= \begin{bmatrix} \mathbf{a}_1 \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{J} \mathbf{a}_1^* \\ 1 \end{bmatrix} k_1 \\
 &= \begin{bmatrix} -1/2 \\ 0 \end{bmatrix} + \begin{bmatrix} -1/2 \\ 1 \end{bmatrix} (-1/3) = \begin{bmatrix} -1/3 \\ -1/3 \end{bmatrix}
 \end{aligned}$$

Repeating step (3) for $m = 2$

$$\begin{aligned}
 (a) \quad P_2 &= P_1 + \beta_1 k_1 = 3/4 - (1/4)(1/3) = 2/3 \\
 (b) \quad \mathbf{r}_2 &= [\mathbf{r}(1) \ \mathbf{r}(2)]^T = [1/2 \ 1/2]^T \\
 (c) \quad \beta_2 &= \mathbf{a}_2^T \mathbf{J} \mathbf{r}_2^* + r^*(3) = [-1/3 \ 1/3] \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix} + (1/4) = -1/12 \\
 (d) \quad k_2 &= -\frac{\beta_2}{P_2} = \frac{1/12}{2/3} = 1/8 \\
 (e) \quad \mathbf{a}_3 &= \begin{bmatrix} \mathbf{a}_2 \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{J} \mathbf{a}_2^* \\ 1 \end{bmatrix} k_2 \\
 &= \begin{bmatrix} -1/3 \\ -1/3 \\ 0 \end{bmatrix} + \begin{bmatrix} -1/3 \\ -1/3 \\ 1 \end{bmatrix} (1/8) = \begin{bmatrix} -3/8 \\ -3/8 \\ 1/8 \end{bmatrix}
 \end{aligned}$$

Lastly, using step (4)

$$P_3 = P_2 + \beta_2 k_2 = 2/3 - (1/12)(1/8) = 21/32$$

Therefore, the third-order linear predictor \mathbf{a}_3 is

$$\mathbf{a}_3 = \begin{bmatrix} -3/8 \\ -3/8 \\ 1/8 \end{bmatrix}$$

and the MMSE is $P_3 = 21/32$

7.17 Given the autocorrelation sequence $r(0) = 1, r(1) = r(2) = 1/2$, and $r(3) = 1/4$, compute the lattice and direct-form coefficients of the prediction error filter, using the algorithm of Schür.

Using Table 7.4 to help solve for the lattice coefficients k_m , starting with step 2

$$(a) \quad \xi_0^f(l) = \xi_0^b(l) = r(l)$$

$$(b) \quad k_0 = -\frac{\xi_0^f(1)}{\xi_0^b(0)} = \frac{-1/2}{1} = -1/2$$

$$(c) \quad P_1 = r(0)(1 - |k_0|^2) = (1)(1 - |1/4|^2) = 3/4$$

Continuing the algorithm with step 3, starting with $m = 1$

$$(a) \quad \xi_1^f(l) = \xi_{m-1}^f(l) + k_{m-1}^* \xi_{m-1}^b(l-1)$$

$$\xi_1^f(2) = \xi_0^f(2) + k_0^* \xi_0^b(1) = 1/2 + (-1/2)(1/2) = 1/4$$

$$\xi_1^b(1) = \xi_0^f(0) + k_0^* \xi_0^b(1) = 1 + (-1/2)(1/2) = 3/4$$

$$(b) \quad k_m = -\frac{\xi_m^f(m+1)}{\xi_m^b(m)}$$

$$k_1 = -\frac{\xi_1^f(2)}{\xi_1^b(1)} = \frac{-1/4}{3/4} = -1/3$$

$$(c) \quad P_{m+1} = P_m(1 - |k_m|^2)$$

$$P_2 = P_1(1 - |k_1|^2) = 2/3$$

Repeating step 3, with $m = 2$

$$(a) \quad \xi_1^f(3) = \xi_0^f(3) + k_0^* \xi_0^b(2) = 1/4 + (-1/2)(1/2) = 0$$

$$\xi_1^b(2) = \xi_0^f(1) + k_0^* \xi_0^b(2) = 1/2 + (-1/2)(1/2) = 1/4$$

$$\xi_2^f(3) = \xi_1^f(3) + k_1^* \xi_1^b(2) = 0 + (-1/3)(1/4) = -1/12$$

$$\xi_2^b(2) = \xi_1^f(1) + k_1^* \xi_1^b(2) = 3/4 + (-1/3)(1/4) = 2/3$$

$$(b) \quad k_2 = -\frac{\xi_2^f(3)}{\xi_2^b(2)} = \frac{1/12}{2/3} = 1/8$$

$$(c) \quad P_2 = P_1(1 - |k_1|^2) = (2/3)(1 - |1/8|^2) = 21/32$$

Lastly, the output of the Schür algorithm is $k_0 = -1/2, k_1 = -1/3, k_2 = 1/8$, and $P_3 = 21/32$.

The direct-form coefficients can be found using (7.5.27)

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

$$\mathbf{a}_1 = k_0 = -1/2$$

$$\mathbf{a}_2 = \begin{bmatrix} -1/2 \\ 0 \end{bmatrix} + \begin{bmatrix} -1/2 \\ 1 \end{bmatrix} (-1/3) = \begin{bmatrix} -1/3 \\ -1/3 \end{bmatrix}$$

$$\mathbf{a}_3 = \begin{bmatrix} -1/3 \\ -1/3 \\ 0 \end{bmatrix} + \begin{bmatrix} -1/3 \\ -1/3 \\ 1 \end{bmatrix} (1/8) = \begin{bmatrix} -3/8 \\ -3/8 \\ 1/8 \end{bmatrix}$$

7.18 Determine ρ_1 and ρ_2 so that the matrix $\mathbf{R} = \text{Toeplitz}\{1, \rho_1, \rho_2\}$ is positive definite.

If $\det \mathbf{R} < 1$ then \mathbf{R} is positive definite. The determinant can be written as

$$\begin{aligned} \det \mathbf{R} &< 1 \\ \left| \begin{bmatrix} 1 & \rho_1 & \rho_2 \\ \rho_1 & 1 & \rho_1 \\ \rho_2 & \rho_1 & 1 \end{bmatrix} \right| &< 1 \\ 1 + 2\rho_1^2\rho_2 - 2\rho_1^2 - \rho_2^2 &< 1 \\ (\rho_2 - 1)(2\rho_1^2 - \rho_2 - 1) &< 1 \end{aligned}$$

This last line yields two conditions

$$\rho_2 < 1$$

and

$$2\rho_1^2 - 1 < \rho_2$$

Combining the two yields $2\rho_1^2 - 1 < \rho_2 < 1$ which is exactly the condition shown in (4.2.105).

7.19 An AR(2) model fit for a sinusoidal random-phase signal in additive noise with autocorrelation

$$r(l) = P_0 \cos \omega_0 l + \sigma_v^2 \delta(l)$$

(a) Model parameters $a_1^{(2)}$, $a_2^{(2)}$, and σ_w^2 in terms of P_0 , ω_0 , and σ_v^2 : The model parameters are given by the normal equation

$$\begin{bmatrix} r(0) & r(1) \\ r(1) & r(0) \end{bmatrix} \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \end{bmatrix} = - \begin{bmatrix} r(1) \\ r(2) \end{bmatrix}$$

or

$$\begin{bmatrix} P_0 + \sigma_v^2 & P_0 \cos \omega_0 \\ P_0 \cos \omega_0 & P_0 + \sigma_v^2 \end{bmatrix} \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \end{bmatrix} = - \begin{bmatrix} P_0 \cos \omega_0 \\ P_0 \cos 2\omega_0 \end{bmatrix}$$

Hence

$$\begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \end{bmatrix} = \begin{bmatrix} -P_0 (\cos \omega_0) \frac{P_0 + \sigma_v^2 - P_0 \cos 2\omega_0}{P_0^2 + 2P_0\sigma_v^2 + \sigma_v^4 - P_0^2 \cos^2 \omega_0} \\ -P_0 \frac{-P_0 \cos^2 \omega_0 + P_0 \cos 2\omega_0 + (\cos 2\omega_0)\sigma_v^2}{P_0^2 + 2P_0\sigma_v^2 + \sigma_v^4 - P_0^2 \cos^2 \omega_0} \end{bmatrix}$$

Also

$$\begin{aligned} \sigma_w^2 &= P_0 + \begin{bmatrix} a_1^{(2)} & a_2^{(2)} \end{bmatrix} \begin{bmatrix} r(1) \\ r(2) \end{bmatrix} \\ &= P_0 + \begin{bmatrix} -P_0 (\cos \omega_0) \frac{P_0 + \sigma_v^2 - P_0 \cos 2\omega_0}{P_0^2 + 2P_0\sigma_v^2 + \sigma_v^4 - P_0^2 \cos^2 \omega_0} & -P_0 \frac{-P_0 \cos^2 \omega_0 + P_0 \cos 2\omega_0 + (\cos 2\omega_0)\sigma_v^2}{P_0^2 + 2P_0\sigma_v^2 + \sigma_v^4 - P_0^2 \cos^2 \omega_0} \end{bmatrix} \begin{bmatrix} P_0 \cos \omega_0 \\ P_0 \cos 2\omega_0 \end{bmatrix} \\ &= P_0 \frac{-2P_0^2 \cos^2 \omega_0 - P_0 (\cos^2 \omega_0) \sigma_v^2 + 2P_0^2 \cos^2 \omega_0 \cos 2\omega_0 - P_0^2 \cos^2 2\omega_0 - P_0 (\cos^2 2\omega_0) \sigma_v^2 + P_0^2 + 2P_0\sigma_v^2 + \sigma_v^4}{P_0^2 + 2P_0\sigma_v^2 + \sigma_v^4 - P_0^2 \cos^2 \omega_0} \end{aligned}$$

(b) Lattice parameters k_1 and k_2 in terms of P_0 , ω_0 , and σ_v^2 : We have

$$k_1 = a_2^{(2)} = -P_0 \frac{-P_0 \cos^2 \omega_0 + P_0 \cos 2\omega_0 + (\cos 2\omega_0) \sigma_v^2}{P_0^2 + 2P_0 \sigma_v^2 + \sigma_v^4 - P_0^2 \cos^2 \omega_0}$$

and

$$\begin{aligned} k_0 &= \frac{a_1^{(2)} - a_1^{(2)*} k_1}{1 - k_1^2} = \frac{a_1^{(2)}}{1 + k_1} = \frac{a_1^{(2)}}{1 + a_2^{(2)}} \\ &= -P_0 \frac{\cos \omega_0}{P_0 + \sigma_v^2} \end{aligned}$$

(c) Limiting values as $\sigma_v^2 \rightarrow 0$:

$$a_1^{(1)} = -P_0 (\cos \omega_0) \frac{P_0 - P_0 \cos 2\omega_0}{P_0^2 - P_0^2 \cos^2 \omega_0} = \frac{(\cos \omega_0) (1 - \cos 2\omega_0)}{(\cos \omega_0 - 1) (\cos \omega_0 + 1)}$$

$$a_2^{(2)} = -P_0 \frac{-P_0 \cos^2 \omega_0 + P_0 \cos 2\omega_0}{P_0^2 - P_0^2 \cos^2 \omega_0} = \frac{(\cos 2\omega_0 - \cos^2 \omega_0)}{(\cos \omega_0 - 1) (\cos \omega_0 + 1)} = 1 = k_1$$

$$k_0 = -P_0 \frac{\cos \omega_0}{P_0} = -\cos \omega_0$$

and

$$\begin{aligned} \sigma_w^2 &= P_0 \frac{-2P_0^2 \cos^2 \omega_0 + 2P_0^2 \cos^2 \omega_0 \cos 2\omega_0 - P_0^2 \cos^2 2\omega_0 + P_0^2}{P_0^2 - P_0^2 \cos^2 \omega_0} \\ &= P_0 \frac{(\cos 2\omega_0 - 1) (-\cos 2\omega_0 + 2 \cos^2 \omega_0 - 1)}{-(\cos \omega_0 - 1) (\cos \omega_0 + 1)} = 0 \end{aligned}$$

7.20 Given the parameters $r(0) = 1$, $k_0 = k_1 = 1/2$, and $k_2 = 1/4$, determine all other equivalent representations of the prediction error filter (see Figure 7.7)

The forward predictor can be found directly using (7.5.27)

$$\begin{aligned} \mathbf{a}_m &= \begin{bmatrix} \mathbf{a}_{m-1} \\ 0 \end{bmatrix} + \begin{bmatrix} J \mathbf{a}_{m-1}^* \\ 1 \end{bmatrix} k_{m-1} \\ a_1 &= k_0 = 1/2 \\ \mathbf{a}_2 &= \begin{bmatrix} 1/2 \\ 0 \end{bmatrix} + \begin{bmatrix} 1/2 \\ 1 \end{bmatrix} (1/2) \\ &= \begin{bmatrix} 3/4 \\ 1/2 \end{bmatrix} \\ \mathbf{a}_3 &= \begin{bmatrix} 3/4 \\ 1/2 \\ 0 \end{bmatrix} + \begin{bmatrix} 1/2 \\ 3/4 \\ 1 \end{bmatrix} (1/4) \\ &= \begin{bmatrix} 7/8 \\ 11/16 \\ 1/4 \end{bmatrix} \end{aligned}$$

The autocorrelation can be found directly using (7.5.36) and using the fact that $r(0) = P_0$ then for $m = 0$

$$r(1) = -k_0^* P_0 = -(1/2)(1) = -1/2$$

For $m = 2$

$$r(2) = -k_1^* P_1 - a_1^{(1)*} r(1) = -(1/2)(3/4) - (1/2)(-1/2) = -1/8$$

where

$$P_1 = P_0(1 - |k_0|^2) = (1)(1 - |1/2|^2) = 3/4$$

Finally, for $m = 2$

$$\begin{aligned} r(3) &= -k_2^* P_2 - [a_1^{(2)*} r(2) + k_1^* r(1)] \\ &= -(1/4)(3/4)^2 - [(3/4)(-1/8) + (1/2)(-1/2)] \\ &= 5/16 \end{aligned}$$

where

$$P_2 = P_1(1 - |k_1|^2) = (3/4)(1 - |1/2|^2) = (3/4)^2$$

Therefore the autocorrelation is $r(0) = 1$, $r(1) = -1/2$, $r(2) = -1/8$, and $r(3) = 5/16$

7.21 Let $\{r(l)\}_0^P$ be samples of the autocorrelation sequence of a stationary random signal $x(n)$

(a) Is it possible to extend $r(l)$ for $|l| > P$ so the the PSD

$$R(e^{j\omega}) = \sum_{l=-\infty}^{\infty} r(l)e^{-j\omega l}$$

is valid, that is, $R(e^{j\omega}) \geq 0$?

We first extend $R(l)$ for negative lags using $r(-l) = r^*(l)$. We next recall that

$$\det R_{m+1} = \det R_m P_{m+1} = \det R_m P_m (1 - |k_m|^2)$$

Since R_m is positive definite and $P_m > 0$ (because $r(l)$ is a valid autocorrelation of a stationary process), $\det R_{m+1}$ is positive definite if and only if $(1 - |k_m|^2) > 0$, but

$$k_m = \frac{b_m}{P_m} = \frac{r^*(m+1) + a_m^T J r_m^*}{P_m}$$

Therefore $\det R_{m+1} > 0$ if and only if

$$|r^*(m+1) + a_m^T J r_m^*| < P_m^2$$

which defines an open disk in the complex plane. Any value of $r(m+1)$ inside the disk provides a valid extension of $r(l)$ which inturn leads to a valid PSD $R(e^{j\omega})$

(b) Using the algorithm of Levinson-Durbin, develop a procedure to check if a given autocorrelation extension is valid

For a given autocorrelation extension $r(m+1)$, the Levinson-Durbin algorithm (Table 7.2) can be used to determine its validity. Specifically computing β_m in step 3(c)

$$\beta_m = a_m^T J r_m^* + r^*(m+1)$$

and using this result to compute k_m in step 3(d)

$$k_m = -\beta_m / P_m$$

If $|k_m| < 1$, then the autocorrelation is valid.

- (c) Use the algorithm in part (b) to find the necessary and sufficient conditions so that $r(0) = 1$, $r(1) = \rho_1$, and $r(2) = \rho_2$ are a valid autocorrelation sequence. Is the resulting extension unique?

Using the algorithm from step (b) above

$$\beta_1 = a_1^T J r_1^* + r^*(2) = \rho_2 - \rho_1^2$$

where $a_1^{(1)} = -r(1)/r(0)$ and with $P_1 = P_0 + \beta_0 k_0 = r(0) + r(1) \frac{-r(1)}{r(0)}$

$$k_1 = -\beta_1/P_1 = \frac{a_1^T J r_1^* + r^*(2)}{r^2(0) - |r(1)|^2} = \frac{\rho_2^2 - \rho_1^2}{1 - \rho_2^2}$$

Setting $|k_1| < 1$ results in $2\rho_1^2 - 1 < \rho_2^2$ which is the necessary and sufficient condition and this agrees with (4.2.105). Since the disk defined by (1) is finite, the autocorrelation extension is not unique.

7.22 Justify the following statements.

- (a) The whitening filter for stationary process $x(n)$ is time-varying

Suppose that we use $x(0), x(1), \dots, x(m-1)$ to predict $x(m)$. The FPEF provides an approximation to the whitening filter of the process $x(n)$ determined by spectral factorization (see section 6.6.4). Note that, due to stationarity, we can use $x(0), x(1), \dots, x(m)$ or $x(n), x(n-1), \dots, x(n-m)$ to determine the m th-order FLP. The two filters are identical for an AR(m) process. Since, the coefficients of a_m and a_{m+1} are different and m represents time in $x(0), x(1), \dots, x(m)$, the finite order whitening filter is time varying.

- (b) The filter in (a) can be implemented by using a lattice structure and switching its stages on one by one with the arrival of each new sample.

In a lattice implementation, order recursion corresponds to adding lattice stages. Therefore, to determine the new FPEF when $x(m)$ arrives, we add a new stage to the lattice.

- (c) If $x(n)$ is AR(P), the whitening filter becomes time-invariant $P + 1$ sampling intervals after the first sample is applied. *Note*: We assume that the input is applied to the filter at $n = 0$. If the input is applied at $n = -\infty$, the whitening filter of a stationary process is always time-invariant.

For an AR(P) process, $k_m = 0$ for $m > P$. Therefore, the lattice does not change for $m = n > P$. Hence, it is time-invariant.

7.23 Given the parameters $r(0) = 1$, $k_0 = 1/2$, $k_1 = 1/3$, and $k_2 = 1/4$, compute the determinant of the matrix $\mathbf{R}_4 = \text{Toeplitz}\{r(0), r(1), r(2), r(3)\}$.

The value of the determinant can be computed without determining the autocorrelation using (7.4.19)

$$\det \mathbf{R}_4 = \prod_{m=0}^3 P_m = r(0) P_1 P_2 P_3$$

where $r(0) = 1$, and

$$\begin{aligned} P_1 &= P_0(1 - |k_0|^2) = (1)(1 - 1/4) = 3/4 \\ P_2 &= P_1(1 - |k_1|^2) = (3/4)(1 - 1/9) = 2/3 \\ P_3 &= P_2(1 - |k_2|^2) = (2/3)(1 - 1/16) = 5/8 \end{aligned}$$

Therefore the $\det \mathbf{R}_4 = (3/4)(2/3)(5/8) = 5/16$

7.24 (a) Determine the lattice second-order prediction error filter (PEF) for a sequence $x(n)$ with autocorrelation $r(l) = (\frac{1}{2})^{|l|}$. (b) Repeat part (a) for the sequence $y(n) = x(n) + v(n)$, where $v(n) \sim \text{WN}(0, 0.2)$ is uncorrelated to $x(n)$. (c) Explain the change in the lattice parameters using frequency domain reasoning (think of the PEF as a whitening filter).

7.25 Consider a prediction error filter specified by $P_3 = (\frac{15}{16})^2$, $k_0 = 1/4$, $k_1 = 1/2$, and $k_2 = 1/4$.

(a) Determine the direct-form filter coefficients.

The forward predictor can be found directly using (7.5.27)

$$\begin{aligned}\mathbf{a}_m &= \begin{bmatrix} \mathbf{a}_{m-1} \\ 0 \end{bmatrix} + \begin{bmatrix} J\mathbf{a}_{m-1}^* \\ 1 \end{bmatrix} k_{m-1} \\ a_1 &= k_0 = 1/4 \\ \mathbf{a}_2 &= \begin{bmatrix} 1/4 \\ 0 \end{bmatrix} + \begin{bmatrix} 1/4 \\ 1 \end{bmatrix} (1/2) \\ &= \begin{bmatrix} 3/8 \\ 1/2 \end{bmatrix} \\ \mathbf{a}_3 &= \begin{bmatrix} 3/8 \\ 1/2 \\ 0 \end{bmatrix} + \begin{bmatrix} 1/2 \\ 3/8 \\ 1 \end{bmatrix} (1/4) \\ &= \begin{bmatrix} 1/2 \\ 19/32 \\ 1/4 \end{bmatrix}\end{aligned}$$

(b) Determine the autocorrelation values $r(1)$, $r(2)$, and $r(3)$.

The autocorrelation can be found directly using (7.5.36) and using the fact that

$$\begin{aligned}r(0) = P_0 &= \frac{P_3}{\prod_{m=0}^2 (1 - |k_m|^2)} \\ &= \frac{(\frac{15}{16})^2}{(15/16)(3/4)(15/16)} \\ &= 4/3\end{aligned}$$

then for $m = 0$

$$r(1) = -k_0^* P_0 = -(1/4)(4/3) = -1/3$$

For $m = 2$

$$r(2) = -k_1^* P_1 - a_1^{(1)*} r(1) = -(1/2)(5/4) + (1/4)(1/3) = -13/24$$

where

$$P_1 = P_0(1 - |k_0|^2) = (4/3)(1 - |1/4|^2) = 5/4$$

Finally, for $m = 2$

$$\begin{aligned}r(3) &= -k_2^* P_2 - [a_1^{(2)*} r(2) + k_1^* r(1)] \\ &= -(1/4)(15/16) - [(3/8)(-13/24) + (1/2)(-1/3)] \\ &= 13/96\end{aligned}$$

where

$$P_2 = P_1(1 - |k_1|^2) = (5/4)(1 - |1/2|^2) = 15/16$$

Therefore the autocorrelation is $r(0) = 4/3$, $r(1) = -1/3$, $r(2) = -13/24$, and $r(3) = 13/96$

- (c) Determine the value $r(4)$ so that the MMSE P_4 for the corresponding fourth-order filter is the minimum possible.

The autocorrelation extension formula (7.5.36)

$$r(m+1) = -k_m^* P_m - \mathbf{a}_m^H J \mathbf{r}_m$$

can be used to compute the next autocorrelation value. Since the value of k_3 is not set, an infinite number of possible values can be used provided that $|k_3| < 1$, therefore insuring that \mathbf{R} is positive definite.

- 7.26** Consider a prediction error filter $A_M(z) = 1 + a_1^{(M)}z^{-1} + \dots + a_M^{(M)}z^{-M}$ with lattice parameters k_1, k_2, \dots, k_M .

- (a) Show that if we set $\hat{k}_m = (-1)^m k_m$, then $\hat{a}_m^{(M)} = (-1)^m a_m^{(M)}$.

For $M = 2$, the prediction error filter is

$$\begin{aligned} A_2 &= 1 + a_1 z^{-1} + a_2 z^{-2} \\ &= 1 + (k_0^* + k_0 k_1^*) z^{-1} + k_1^* z^{-2} \end{aligned}$$

With the substitution of \hat{k}_m into the above equation

$$\begin{aligned} \hat{A}_2 &= 1 + \hat{a}_1 z^{-1} + \hat{a}_2 z^{-2} \\ &= 1 + (\hat{k}_0^* + \hat{k}_0 \hat{k}_1^*) z^{-1} + \hat{k}_1^* z^{-2} \\ &= 1 + (-k_0^* - k_0 k_1^*) z^{-1} - k_1^* z^{-2} \end{aligned}$$

then $\hat{a}_1^{(2)} = -a_1^{(2)}$ and $\hat{a}_2^{(2)} = a_2^{(2)}$. ?????

- (b) What are the new filter coefficients if we set $\hat{k}_m = \rho^m k_m$, where ρ is a complex number with $|\rho| = 1$? What happens if $|\rho| < 1$?

If $\hat{k}_0 = \rho k_0$ and $\hat{k}_1 = \rho^2 k_1$ where $\rho = r e^{j\omega}$ then

$$\begin{aligned} \hat{a}_1^{(2)} &= \hat{k}_0^* + \hat{k}_0 \hat{k}_1^* \\ &= \rho^* k_0^* + \rho k_0 \rho^2 k_1^* \\ &= r e^{j\omega} k_0^* + r e^{j\omega} k_0 r^2 e^{j2\omega} k_1^* \\ &= r e^{j\omega} (k_0^* + r^2 k_0 k_1^*) \end{aligned}$$

When $r = 1$ then $\hat{a}_1^{(2)} = e^{-j\omega} a_1^{(2)}$. For $\hat{a}_2^{(2)}$

$$\hat{a}_2^{(2)} = \hat{k}_1^* = \rho^* k_1^* = r^2 e^{-j2\omega} k_1^*$$

so when $r = 1$, $\hat{a}_2^{(2)} = e^{-j2\omega} a_2^{(2)}$

- 7.27** Suppose that we are given the values $\{r(l)\}_{-m+1}^{m-1}$ of an autocorrelation sequence such that the Toeplitz matrix \mathbf{R}_m is positive definite.

- (a) Show that the values of $r(m)$ such that \mathbf{R}_{m+1} is positive definite determine a disk in the complex plane. Find the center α_m and the radius ζ_m of the disk

Figure similar to 6.3 in Haykin

If \mathbf{R}_m is positive definite, then the necessary and sufficient condition for \mathbf{R}_{m+1} to be positive definite is

$$r(m+1) = -k_m^* P_m - \mathbf{a}_m^H J \mathbf{r}_m$$

for any value of k_m such that $|k_m| < 1$. This establishes a circular region in the Real plane of possible values for $r(m+1)$, with the circle centered at $\{\text{Re}[\mathbf{a}_m^H J \mathbf{r}_m], \text{Im}[\mathbf{a}_m^H J \mathbf{r}_m]\}$ and a maximum radius of P_m .

(b) By induction show that there are infinitely many extension of $\{r(l)\}_{-m+1}^{m-1}$ that make $\{r(l)\}_{-\infty}^{\infty}$ a valid autocorrelation sequence.

Any choice of $|k_m| < 1$ provides a valid $r(m+1)$. Hence, there are infinitely many extensions.

7.28 Consider the MA(1) sequence $x(n) = w(n) + d_1 w(n-1)$, $w(n) \sim \text{WN}(0, \sigma_w^2)$. Then

$$r(0) = \sigma_w^2 (1 + |d_1|^2), \quad r(1) = r^*(-1) = d_1 \sigma_w^2, \quad \text{and } r(l) = 0, \quad |l| > 0 \quad (1)$$

and

$$\mathbf{R}_m = \begin{bmatrix} r(0) & r(1) & 0 & \cdots & 0 & 0 \\ r^*(1) & r(0) & r(1) & \cdots & 0 & 0 \\ 0 & r^*(1) & r(0) & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \ddots & r(0) & r(1) \\ 0 & 0 & 0 & \cdots & r^*(1) & r(0) \end{bmatrix} \quad (2)$$

(a) Show that

$$\det \mathbf{R}_m = r(0) \det \mathbf{R}_{m-1} - |r(1)|^2 \det \mathbf{R}_{m-2} \quad m \geq 2$$

From (2) for $m \geq 2$

$$\begin{aligned} \det \mathbf{R}_m &= r(0) \det \left(\begin{bmatrix} r(0) & r(1) & \cdots & 0 & 0 \\ r^*(1) & r(0) & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \ddots & r(0) & r(1) \\ 0 & 0 & \cdots & r^*(1) & r(0) \end{bmatrix} \right) - \\ &\quad r(1)r^*(1) \det \left(\begin{bmatrix} r(0) & \ddots & 0 & 0 \\ \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & r(0) & r(1) \\ 0 & \cdots & r^*(1) & r(0) \end{bmatrix} \right) \end{aligned}$$

where the first determinant is of $(m-1) \times (m-1)$ matrix \mathbf{R}_{m-1} and the second determinant is of $(m-2) \times (m-2)$ matrix \mathbf{R}_{m-2} . Hence

$$\det \mathbf{R}_m = r(0) \det \mathbf{R}_{m-1} - |r(1)|^2 \det \mathbf{R}_{m-2}, \quad m \geq 2 \quad (3)$$

(b) Show that $k_m = -r^m(1)/\det \mathbf{R}_m$ and that

$$\frac{1}{k_m} = -\frac{r(0)}{r(1)} \frac{1}{k_{m-1}} - \frac{r^*(1)}{r(1)} \frac{1}{k_{m-2}}$$

The reflection coefficients k_i 's are given by (4.2.23)

$$\mathbf{R}_m \begin{bmatrix} a_1^{(m)} \\ a_2^{(m)} \\ \vdots \\ k_m \end{bmatrix} = - \begin{bmatrix} r(1) \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Hence

$$k_m = \frac{\det \begin{bmatrix} r(0) & r(1) & 0 & \cdots & 0 & -r(1) \\ r^*(1) & r(0) & r(1) & \cdots & 0 & 0 \\ 0 & r^*(1) & r(0) & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \ddots & r(0) & 0 \\ 0 & 0 & 0 & \cdots & r^*(1) & 0 \end{bmatrix}}{\det \mathbf{R}_m} = \frac{-r^m(1)}{\det \mathbf{R}_m} \quad (4)$$

Dividing both sides of (3) by $-r^m(1)$, we obtain

$$\begin{aligned} \frac{\det \mathbf{R}_m}{-r^m(1)} &= \frac{r(0) \det \mathbf{R}_{m-1}}{-r^m(1)} - |r(1)|^2 \frac{\det \mathbf{R}_{m-2}}{-r^m(1)} \\ \frac{1}{k_m} &= -\frac{r(0)}{-r(1)} \left(\frac{\det \mathbf{R}_{m-1}}{-r^{m-1}(1)} \right) - \frac{r(1)r^*(1)}{r(1)r(1)} \left(\frac{\det \mathbf{R}_{m-2}}{-r^{m-2}(1)} \right) \\ \frac{1}{k_m} &= -\frac{r(0)}{-r(1)} \frac{1}{k_{m-1}} - \frac{r^*(1)}{r(1)} \frac{1}{k_{m-2}} \end{aligned} \quad (5)$$

(c) Determine the initial conditions and solve the recursion in (b) to show that

$$k_m = \frac{(1 - |d_1|^2)(-d_1)^m}{1 - |d_1|^{2m+2}}$$

which tends to zero as $m \rightarrow \infty$.

Note that (5) which is a difference equation in $p(m) \triangleq \frac{1}{k_{m+3}}$, $m \geq 0$ with initial conditions (see Problem 4.25)

$$p(-2) = \frac{1}{k_1} = \frac{1 + |d_1|^2}{(-d_1)} \text{ and } p(-1) = \frac{1}{k_2} = -\frac{1 + |d_1|^6}{(1 - |d_1|^2)(-d_1)^2} \quad (6)$$

Using unilateral z -transform along with (6) with fair amount of algebra, we obtain

$$k_m = \frac{(1 - |d_1|^2)(-d_1)^m}{1 - |d_1|^{2m+2}}$$

which tends to zero as $m \rightarrow \infty$ since $|d_1| < 1$.

7.29 Proof of (7.6.6).

From (7.6.3), substituting $m0l$ for l , we have

$$\xi_m^{f*}(m-l) = r^*(m-l) + \mathbf{a}_m^H \tilde{\mathbf{r}}_m(m-l-1) = r(l-m) + \mathbf{a}_m^H \tilde{\mathbf{r}}_m^*(m-l-1) \quad (1)$$

Using $\mathbf{b}_m = \mathbf{J} \mathbf{a}_m^*$ or $\mathbf{a}_m^{H*} = \mathbf{b}_m^H \mathbf{J}$ in (1), we have

$$\xi_m^{f*}(m-l) = r(l-m) + \mathbf{b}_m^H \mathbf{J} \tilde{\mathbf{r}}_m^*(m-l-1) \quad (2)$$

From (7.6.5)

$$\begin{aligned} \mathbf{J} \tilde{\mathbf{r}}_m^*(m-l-1) &= \mathbf{J} [r^*(m-l-1) \ r^*(m-l-2) \ \cdots \ r^*(-l)] \\ &= \mathbf{J} [r(l-m+1) \ r(l-m+2) \ \cdots \ r(l)] \\ &= [r(l) \ r(l-1) \ \cdots \ r(l-m+1)] = \tilde{\mathbf{r}}_m(l) \end{aligned} \quad (3)$$

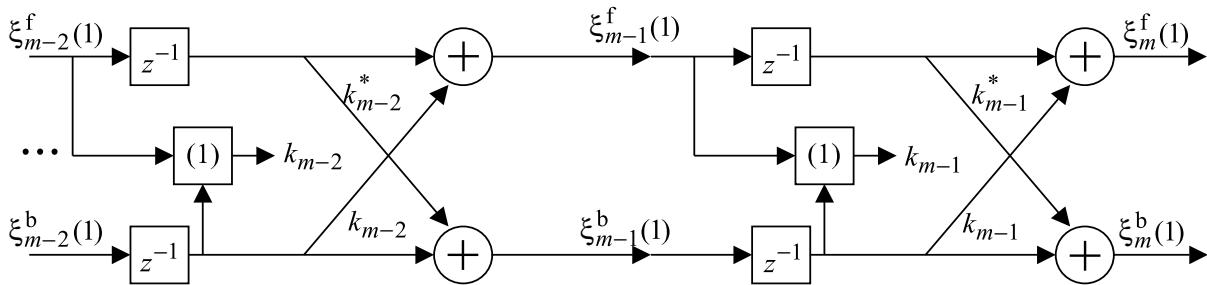


Figure 7.30: Flow diagram of the lattice filter with the lattice parameters

Substituting (3) in (2), we obtain

$$\xi_m^{f*}(m-l) = r(l-m) + \mathbf{b}_m^H \mathbf{J} \tilde{\mathbf{r}}_m(l) = \xi_m^b(l)$$

which proves (7.6.6).

- 7.30** In this problem we show that the lattice parameters can be obtained by "feeding" the autocorrelation sequence through the lattice filter as a signal and switching on the stages one by one after the required lattice coefficient is computed. The value of k_m is computed at time $n = m$ from the inputs to stage m .

- (a) Using (7.6.10), draw the flow diagram of a third-order lattice filter that implements this algorithm.

Figure 7.30 shows the flow diagram of the lattice filter with the lattice parameter computed per

$$k_m = -\frac{\xi_m^f(l)}{\xi_m^b(m)} \quad (1)$$

- (b) Using the autocorrelation sequence in Example 7.6.1, "feed" the sequence $\{r(n)\}_0^3 = \{3, 2, 1, \frac{1}{2}\}$ through the filter one sample at a time, and compute the lattice parameters. *Hint* : Use Example 7.6.1 for guidance.

Using (7.6.10)

$$\begin{aligned} \xi_0^f(l) &= \xi_0^b(l) = r(l) \\ \xi_m^f(l) &= \xi_{m-1}^f(l) + k_{m-1}^* \xi_{m-1}^b(l-1) \\ \xi_m^b(l) &= \xi_{m-1}^b(l-1) + k_{m-1} \xi_{m-1}^f(l) \end{aligned}$$

and the flow diagram above, the lattice parameters can be computed using the autocorrelation as the input. For each instance in time, the lattice has a number of stages equal to the time index starting with time equal zero. With $l = 0$, the first stage is $\xi_0^f(0) = \xi_0^b(0) = r(0) = 3$. The second stage is computed with $l = 1$ and $r(1) = 2$ as the input, with the following output

$$\begin{aligned} \xi_0^f(1) &= \xi_0^b(1) = r(1) = 2 \\ k_0 &= \frac{-\xi_0^f(1)}{\xi_0^b(0)} = -2/3 \\ \xi_1^b(1) &= \xi_0^b(0) + k_0 \xi_0^f(1) = 3 + (-2/3)(2) = 5/3 \end{aligned}$$

The second lattice parameter is computed with $l = 2$ and $r(2) = 1$

$$\begin{aligned}\xi_0^f(2) &= \xi_0^b(2) = r(2) = 1 \\ \xi_1^f(2) &= \xi_0^f(2) + k_0^* \xi_0^b(1) = 1 + (-2/3)(2) = -1/3 \\ \xi_1^b(2) &= \xi_0^b(1) + k_0 \xi_0^f(2) = 2 + (-2/3)(1) = 4/3 \\ k_1 &= -\frac{\xi_1^f(2)}{\xi_1^b(1)} = -\frac{-1/3}{4/3} = 1/5 \\ \xi_2^b(2) &= \xi_1^b(1) + k_1 \xi_1^f(2) = 5/3 + (1/5)(-1/3) = 8/5\end{aligned}$$

and the last lattice parameter is computed with $l = 3$ and $r(3) = 1/2$

$$\begin{aligned}\xi_0^f(3) &= \xi_0^b(3) = r(3) = 1/2 \\ \xi_1^f(3) &= \xi_0^f(3) + k_0^* \xi_0^b(2) = 1/2 + (-2/3)(1) = -1/6 \\ \xi_2^f(3) &= \xi_1^f(3) + k_1^* \xi_1^b(2) = (-1/6) + (1/5)(4/3) = 1/10 \\ k_2 &= -\frac{\xi_2^f(3)}{\xi_2^b(2)} = -\frac{1/10}{8/5} = -1/16\end{aligned}$$

Therefore the lattice parameters are $k_0 = -2/3$, $k_1 = -1/4$, and $k_2 = -1/16$

- 7.31** Draw the superlattice structure for $M = 8$, and show how it can be partitioned to distribute the computations to three processors for parallel execution.

Figure 7.31 shows the superlattice structure, and a partition of computations for three parallel processors.

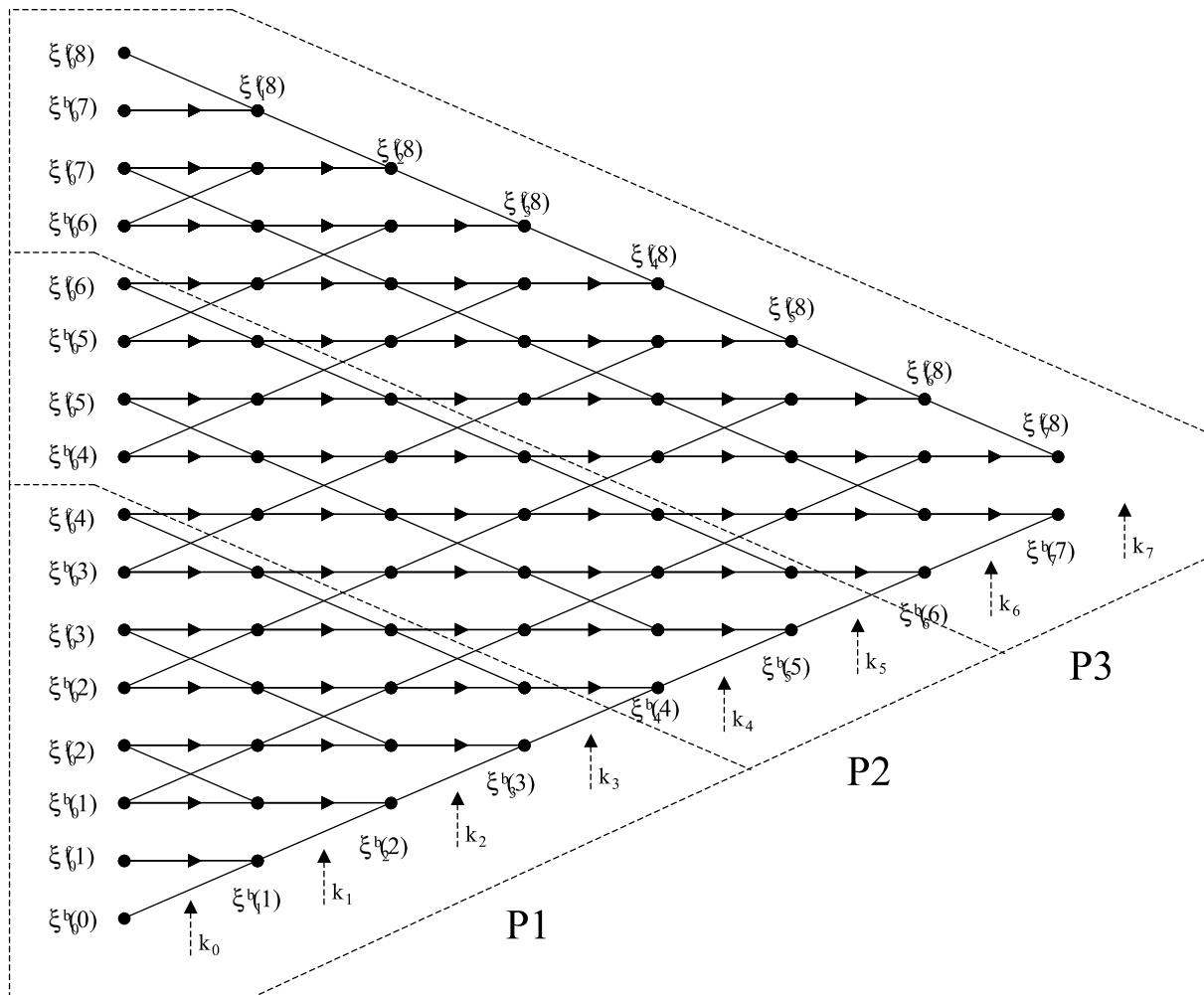
- 7.32** Derive the superladder structure shown in Figure 7.10.

The superladder structure shown in Figure 7.10 are computed using (7.6.15), (7.6.16), and (7.6.17). Each stage of the superladder is shown below. Given the autocorrelation and the cross-correlation parameters, the first stage, working left to right, is

$$\begin{aligned}\xi_0^c(0) &= d_1 \\ \xi_0^c(1) &= d_2 \\ \xi_0^c(2) &= d_3 \\ \xi_0^c(3) &= d_4 \\ \xi_0^b(0) &= r(0) \\ \xi_0^b(1) &= r(1) \\ \xi_0^b(2) &= r(2) \\ \xi_0^b(3) &= r(3)\end{aligned}$$

and the lattice-ladder coefficient k_0^c is

$$k_0^c = \frac{-\xi_0^c(0)}{\xi_0^b(0)}$$

**Figure 7.31:** Superlattice structure

The second stage is

$$\begin{aligned}\xi_1^c(1) &= \xi_0^c(1) + k_0^c \xi_0^b(1) \\ \xi_1^c(2) &= \xi_0^c(2) + k_0^c \xi_0^b(2) \\ \xi_1^c(3) &= \xi_0^c(3) + k_0^c \xi_0^b(3)\end{aligned}$$

and the lattice-ladder coefficient k_1^c is

$$k_1^c = \frac{-\xi_1^c(1)}{\xi_1^b(1)}$$

The third stage is

$$\begin{aligned}\xi_2^c(2) &= \xi_1^c(2) + k_1^c \xi_1^b(2) \\ \xi_2^c(3) &= \xi_1^c(3) + k_1^c \xi_1^b(3)\end{aligned}$$

and the lattice-ladder coefficient k_2^c is

$$k_2^c = \frac{-\xi_2^c(2)}{\xi_2^b(2)}$$

Finally, the last stage is

$$\xi_3^c(3) = \xi_2^c(3) + k_2^c \xi_2^b(3)$$

and

$$k_3^c = \frac{-\xi_3^c(3)}{\xi_3^b(3)}$$

- 7.33** Extend the algorithm of Schür to compute the LDL^H decomposition of a Hermitian Toeplitz matrix, and write a MATLAB function for its implementation.

The LDL^H decomposition of a Hermitian Toeplitz matrix can be accomplished using (7.7.12)

$$L = B^{-1} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ \bar{\xi}_0^b(1) & 1 & \cdots & 0 & 0 \\ \bar{\xi}_0^b(2) & \bar{\xi}_1^b(1) & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \bar{\xi}_0^b(M) & \bar{\xi}_1^b(M) & \cdots & \bar{\xi}_{M-1}^b(M) & 1 \end{bmatrix}$$

where $\bar{\xi}_m^b(l) = \frac{\xi_m^b(l)}{\xi_m^b(m)} = \frac{\xi_m^b(l)}{P_m}$ and $D = \text{diag}\{P_0, P_1, \dots, P_M\}$.

- 7.34** Given the matrix $\mathbf{R}_3 = \text{Toeplitz}\{1, 1/2, 1/2\}$, use the appropriate order-recursive algorithms to compute the following:

- (a) The LDL^H and UDU^H decompositions of \mathbf{R}

The decompositions of \mathbf{R} into LDL^H can be done using (7.1.38)

$$\mathbf{R}_{m+1} = L_{m+1} D_{m+1} L_{m+1}^H$$

where L_{m+1} and D_{m+1} are generated from L_m and D_m using (7.1.37)

$$L_{m+1} = \begin{bmatrix} L_m & 0 \\ l_m^H & 1 \end{bmatrix} \quad D_{m+1} = \begin{bmatrix} D_m & 0 \\ 0^H & \xi_{m+1} \end{bmatrix}$$

and $\xi_{m+1} = \det \mathbf{R}_{m+1} / \det \mathbf{R}_m$. The vector l_m can be found by solving (7.1.39)

$$(L_m D_m) l_m = r_m^b$$

Starting with $m = 1$, then $L_1 = 1$, $D_1 = 1$, and $r_1^b = 1/2$ so

$$(L_1 D_1) l_1 = r_1^b \Rightarrow l_1 = 1/2$$

and $\xi_2 = \det \mathbf{R}_2 / \det \mathbf{R}_1 = 3/4$. For $m = 2$

$$\begin{aligned} (L_2 D_2) l_2 &= r_2^b \Rightarrow \begin{bmatrix} 1 & 0 \\ 1/2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 3/4 \end{bmatrix} l_2 = \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix} \\ \begin{bmatrix} 1 & 0 \\ 1/2 & 3/4 \end{bmatrix} l_2 &= \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix} \Rightarrow l_2 = \begin{bmatrix} 1/2 \\ 1/3 \end{bmatrix} \end{aligned}$$

and $\xi_3 = \det \mathbf{R}_3 / \det \mathbf{R}_2 = \frac{1/2}{3/4} = 2/3$. So for $m = 3$ the LDL^H decomposition is

$$L_{m+1} = \begin{bmatrix} L_m & 0 \\ l_m^H & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ 1/2 & 1/3 & 1 \end{bmatrix} \quad D_{m+1} = \begin{bmatrix} D_m & 0 \\ 0^H & \xi_{m+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3/4 & 0 \\ 0 & 0 & 2/3 \end{bmatrix}$$

The UDU^H decomposition is easily found using (7.7.15)

$$\mathbf{R} = J \mathbf{R}^* J = (J L^* J) (J D J) (J L^H J) = U \bar{D} U^H$$

(b) The LDL^H and UDU^H decompositions of \mathbf{R}^{-1}

The LDL^H decomposition of \mathbf{R}^{-1} can be computed in an order-recursive manner using

$$L_{m+1}^{-1} = \begin{bmatrix} \mathbf{L}_m & \mathbf{0} \\ l_m^H & 1 \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{L}_m^{-1} & \mathbf{0} \\ v_m^H & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{L}_m^{-1} & \mathbf{0} \\ b_m^H & 1 \end{bmatrix}$$

where b_m is computed as shown in part (c) below. Also $D_m = \text{diag}\{P_0^b, P_1^b, \dots, P_{m-1}^b\}$, where P_m^b is

$$P_m^b = \rho_m^b - \mathbf{r}_m^{bH} \mathbf{R}_m^{-1} \mathbf{r}_m^b = \rho_m^b + \mathbf{r}_m^{bH} b_m = \alpha_m^b$$

where α_m^b is computed in part (c) below. Then L_m^{-1} is

$$\begin{aligned} b_1 &= -1/2 \Rightarrow L_2^{-1} = \begin{bmatrix} 1 & 0 \\ -1/2 & 1 \end{bmatrix} \\ D_2 &= \begin{bmatrix} P_0^b & 0 \\ 0 & P_1^b \end{bmatrix} = \begin{bmatrix} \alpha_1^b & 0 \\ 0 & \alpha_2^b \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 3/4 \end{bmatrix} \\ b_1 &= \begin{bmatrix} -1/3 \\ -1/3 \end{bmatrix} \Rightarrow L_3^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ -1/2 & 1 & 0 \\ -1/3 & -1/3 & 1 \end{bmatrix} \\ D_3 &= \begin{bmatrix} P_0^b & 0 & 0 \\ 0 & P_1^b & 0 \\ 0 & 0 & P_2^b \end{bmatrix} = \begin{bmatrix} \alpha_1^b & 0 & 0 \\ 0 & \alpha_2^b & 0 \\ 0 & 0 & \alpha_3^b \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3/4 & 0 \\ 0 & 0 & 2/3 \end{bmatrix} \end{aligned}$$

(c) The inverse matrix \mathbf{R}^{-1} .

The order-recursive algorithm for the decomposition of the inverse of \mathbf{R} is (7.1.24)

$$\mathbf{R}_{m+1}^{-1} = \begin{bmatrix} \mathbf{R}_m & \mathbf{r}_m^b \\ \mathbf{r}_m^{bH} & \rho_m^b \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{R}_m^{-1} & \mathbf{0}_m \\ \mathbf{0}_m^H & 0 \end{bmatrix} + \frac{1}{\alpha_m^b} \begin{bmatrix} b_m \\ 1 \end{bmatrix} [b_m^H \ 1]$$

where b_m and α_m^b are found using (7.1.22) and (7.1.23) respectively. Starting with $m = 1$, $\mathbf{R}_1 = 1$ therefore $\mathbf{R}_1^{-1} = 1$, $\mathbf{r}_1^b = 1/2$, and $\rho_1^b = 1$. Then \mathbf{R}_2^{-1} can be computed as

$$\begin{aligned} b_1 &= -\mathbf{R}_1^{-1}\mathbf{r}_1^b = -(1)(1/2) = -1/2 \\ \alpha_1^b &= \rho_1^b + \mathbf{r}_2^{bH}b_2 = 1 + (1/2)(-1/2) = 3/4 \\ \mathbf{R}_2^{-1} &= \begin{bmatrix} \mathbf{R}_1^{-1} & \mathbf{0}_1 \\ \mathbf{0}_1^H & 0 \end{bmatrix} + \frac{1}{\alpha_1^b} \begin{bmatrix} b_1 \\ 1 \end{bmatrix} [b_1^H \ 1] \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + \frac{4}{3} \begin{bmatrix} -1/2 \\ 1 \end{bmatrix} [-1/2 \ 1] \\ &= \begin{bmatrix} 4/3 & -1/2 \\ -1/2 & 4/3 \end{bmatrix} \end{aligned}$$

The above algorithm is repeated for $m = 2$, with $\mathbf{r}_2^b = [1/2 \ 1/2]^T$ and $\rho_2^b = 1$

$$\begin{aligned} b_2 &= -\mathbf{R}_2^{-1}\mathbf{r}_2^b = - \begin{bmatrix} 4/3 & -1/2 \\ -1/2 & 4/3 \end{bmatrix} \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix} = \begin{bmatrix} -1/3 \\ -1/3 \end{bmatrix} \\ \alpha_2^b &= \rho_2^b + \mathbf{r}_3^{bH}b_3 = 1 + [1/2 \ 1/2] \begin{bmatrix} -1/3 \\ -1/3 \end{bmatrix} = 2/3 \\ \mathbf{R}_3^{-1} &= \begin{bmatrix} \mathbf{R}_2^{-1} & \mathbf{0}_2 \\ \mathbf{0}_2^H & 0 \end{bmatrix} + \frac{1}{\alpha_2^b} \begin{bmatrix} b_2 \\ 1 \end{bmatrix} [b_2^H \ 1] \\ &= \begin{bmatrix} 7/3 & -2/3 & 0 \\ -2/3 & 4/3 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \frac{3}{2} \begin{bmatrix} -1/3 \\ -1/3 \\ 1 \end{bmatrix} [-1/3 \ -1/3 \ 1] \\ &= \begin{bmatrix} 3/2 & -1/2 & -1/2 \\ -1/2 & 3/2 & -1/2 \\ -1/2 & -1/2 & 3/2 \end{bmatrix} \end{aligned}$$

7.35 Consider the AR(1) process $x(n) = \rho x(n-1) + w(n)$, where $w(n) \sim \text{WN}(0, \sigma_w^2)$ and $-1 < \rho < 1$. The z -domain PSD of $x(n)$ is given by

$$R_x(z) = \frac{\sigma_w^2}{(1 - \rho z)(1 - \rho z^{-1})}; \quad \frac{1}{|\rho|} < |z| < |\rho|, \quad -1 < \rho < 1$$

and the correlation sequence is given by $r_x(l) = \frac{\sigma_w^2}{1 - \rho^2} \rho^{|l|}$.

(a) Determine the correlation matrix \mathbf{R}_{M+1} of the process.

For an arbitrary value of M , we have

$$\mathbf{R}_{M+1} = \begin{bmatrix} 1 & \rho & \rho^2 & \cdots & \rho^M \\ \rho & 1 & \rho & \cdots & \rho^{M-1} \\ \rho^2 & \rho & 1 & \cdots & \rho^{M-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho^M & \rho^{M-1} & \rho^{M-2} & \cdots & 1 \end{bmatrix}$$

The following Matlab script is given for $M = 3$, $\rho = 0.8$, and $\sigma_w^2 = 1$.

```

rho = 0.8; M = 3; var_w = 1;

% (a) Correlation Matrix
R = var_w/(1-rho*rho)*toeplitz(rho.^[0:M])
R =
    2.7778    2.2222    1.7778    1.4222
    2.2222    2.7778    2.2222    1.7778
    1.7778    2.2222    2.7778    2.2222
    1.4222    1.7778    2.2222    2.7778

```

(b) Determine the M th-order FLP, using the algorithm of Levinson-Durbin.

The following Matlab script is given for $M = 3$, $\rho = 0.8$, and $\sigma_w^2 = 1$.

```

% (b) Mth-order FLP using Levinson-Durbin
[a,k,Po] = durbin(R(:,1),M); a
a =
    1.0000
   -0.8000
   -0.0000
      0

```

(c) Determine the inverse matrix \mathbf{R}_{M+1}^{-1} , using the triangular decomposition discussed in Section 7.7.

The following Matlab script is given for $M = 3$, $\rho = 0.8$, and $\sigma_w^2 = 1$.

```

% (c) Inverse of the R matrix
Q = invtoepl(R(:,1),M+1)
Q =
    1.0000   -0.8000   -0.0000       0
   -0.8000    1.6400   -0.8000   -0.0000
   -0.0000   -0.8000    1.6400   -0.8000
      0   -0.0000   -0.8000    1.0000

% Check
R_inv = inv(R);
err = max(max(abs(Q-R_inv)))
err =
  4.4409e-016

```

7.36 If $r(l) = \cos \omega_0 l$, determine the second-order prediction error filter and check whether it is minimum-phase.

The second-order forward prediction error filter is given by

$$e(n) = x(n) + a_1 x(n-1) + a_2 x(n-2)$$

where prediction error filter coefficients are given by $\mathbf{R} \mathbf{a} = -\mathbf{r}$, that is,

$$\begin{bmatrix} r(0) & r(1) \\ r(1) & r(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = -\begin{bmatrix} r(1) \\ r(2) \end{bmatrix}$$

or

$$\begin{bmatrix} 1 & \cos \omega_0 \\ \cos \omega_0 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = -\begin{bmatrix} \cos \omega_0 \\ \cos 2\omega_0 \end{bmatrix}$$

or

$$\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \frac{\cos \omega_0}{\cos^2 \omega_0 - 1} - \frac{\cos \omega_0}{\cos^2 \omega_0 - 1} \cos 2\omega_0 \\ -\frac{\cos \omega_0}{\cos^2 \omega_0 - 1} + \frac{1}{\cos^2 \omega_0 - 1} \cos 2\omega_0 \end{bmatrix} = \begin{bmatrix} -2 \cos \omega_0 \\ 1 \end{bmatrix}$$

Thus the prediction error filter is

$$e(n) = x(n) - 2 \cos \omega_0 x(n-1) + x(n-2)$$

with zeros at $z = \pm \angle \omega_0$, that is, on the unit circle. Thus the system is (critically) minimum-phase system.

- 7.37** Show that the MMSE linear predictor of $x(n+D)$ in terms of $x(n), x(n-1), \dots, x(n-M+1)$ for $D \geq 1$ is given by

$$\mathbf{R}\mathbf{a}^{(D)} = -\mathbf{r}^{(D)}$$

where $\mathbf{r}^{(D)} = [r(D) \ r(D+1) \ \dots \ r(D+M-1)]^T$.

The predictor is given by

$$\hat{x}(n+D) = - \sum_{k=0}^{M-1} a_k x(n-k)$$

and the prediction error is given by

$$e_D(n) \triangleq x(n+D) - \hat{x}(n+D) = x(n+D) + \sum_{k=0}^{M-1} a_k x(n-k)$$

Using the orthogonality principal, $e_D(n) \perp x(n-l)$, $l = 0, 1, \dots, M-1$, we obtain

$$\mathbb{E} \left[\left\{ x(n+D) + \sum_{k=0}^{M-1} a_k x(n-k) \right\} x(n-l) \right] = 0, \quad l = 0, 1, \dots, M-1$$

or

$$r(D+l) + \sum_{k=0}^{M-1} a_k r(l-k) = 0 \Rightarrow \sum_{k=0}^{M-1} a_k r(l-k) = -r(D+l), \quad l = 0, 1, \dots, M-1$$

or

$$\begin{bmatrix} r(0) & r(-1) & \cdots & r(1-M) \\ r(1) & r(0) & \cdots & r(2-M) \\ \vdots & \vdots & \ddots & \vdots \\ r(M-1) & r(M-2) & \cdots & r(0) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{M-1} \end{bmatrix} = - \begin{bmatrix} r(D) \\ r(D+1) \\ \vdots \\ r(D+M-1) \end{bmatrix}$$

$$\mathbf{R}\mathbf{a}^{(D)} \triangleq -\mathbf{r}^{(D)}$$

Develop a recursion that computes $\mathbf{a}^{(D+1)}$ from $\mathbf{a}^{(D)}$ by exploring the shift invariance of the vector $\mathbf{r}^{(D)}$.

We want to determine $\mathbf{a}^{(D+1)}$ (which is otherwise obtained from $\mathbf{R}\mathbf{a}^{(D+1)} \triangleq -\mathbf{r}^{(D+1)}$) directly from $\mathbf{a}^{(D)}$. We will use subscripts to denote the size of a matrix or vector. Consider the partitioning

$$\mathbf{R}_M \mathbf{a}_M^{(D)} = - \begin{bmatrix} r(D) \\ r(D+1) \\ \vdots \\ r(D+M-1) \end{bmatrix} = \begin{bmatrix} r(D) \\ r(D+1) \\ \vdots \\ r(D+M-1) \end{bmatrix} \quad (1)$$

and

$$\mathbf{R}_M \mathbf{a}_M^{(D+1)} = - \begin{bmatrix} r(D+1) \\ \vdots \\ r(D+M-1) \\ r(D+M) \end{bmatrix} \quad (2)$$

Since \mathbf{R}_M is Hermitian Toeplitz, we have $\mathbf{R}_M = \mathbf{J} \mathbf{R}_M \mathbf{J}$, $m = 1, \dots, M$. Substituting in (1), we have

$$\mathbf{J} \mathbf{R}_M \mathbf{J} \mathbf{a}_M^{(D)} = - \begin{bmatrix} r(D) \\ r(D+1) \\ \vdots \\ r(D+M-1) \end{bmatrix}$$

Premultiplying by \mathbf{J}

$$\mathbf{R}_M (\mathbf{J} \mathbf{a}_M^{(D)}) = - \begin{bmatrix} r(D+M-1) \\ \vdots \\ r(D+1) \\ r(D) \end{bmatrix} \quad (3)$$

Now consider the system of order $(M-1)$ from (2)

$$\mathbf{R}_{M-1} \mathbf{a}_{M-1}^{(D+1)} = - \begin{bmatrix} r(D+1) \\ \vdots \\ r(D+M-1) \end{bmatrix}$$

or

$$\mathbf{R}_{M-1} (\mathbf{J} \mathbf{a}_{M-1}^{(D+1)}) = - \begin{bmatrix} r(D+M-1) \\ \vdots \\ r(D+1) \end{bmatrix} \quad (4)$$

From (3) and (4), we have using the Levinson Algorithm

$$\mathbf{J} \mathbf{a}_M^{(D)} = \begin{bmatrix} \mathbf{J} \mathbf{a}_{M-1}^{(D+1)} \\ 0 \end{bmatrix} + k_M^{(D)} \begin{bmatrix} \mathbf{b}_{M-1}^{(D)} \\ 0 \end{bmatrix}$$

or

$$\begin{bmatrix} \mathbf{J} \mathbf{a}_{M-1}^{(D+1)} \\ 0 \end{bmatrix} = \mathbf{J} \mathbf{a}_M^{(D)} - k_M^{(D)} \begin{bmatrix} \mathbf{b}_{M-1}^{(D)} \\ 0 \end{bmatrix} : \text{(Step-down recursion)} \quad (5)$$

Finally, to obtain $\mathbf{a}_M^{(D+1)}$ from $\mathbf{a}_{M-1}^{(D+1)}$, we again use the Levinson recursion

$$\mathbf{a}_M^{(D+1)} = \begin{bmatrix} \mathbf{a}_{M-1}^{(D+1)} \\ 0 \end{bmatrix} + k_M^{(D+1)} \begin{bmatrix} \mathbf{b}_{M-1}^{(D)} \\ 0 \end{bmatrix} : \text{(Step-up recursion)} \quad (6)$$

Thus the overall approach requires a step-down and step-up recursions.

7.38 The normal equations for the optimum symmetric signal smoother (see Section 6.5.1) can be written as

$$\mathbf{R}_{2m+1} \mathbf{c}_{2m+1} = \begin{bmatrix} \mathbf{0} \\ P_{2m+1} \\ \mathbf{0} \end{bmatrix}$$

where P_{2m+1} is the MMSE, $\mathbf{c}_{2m+1} = \mathbf{J}\mathbf{c}_{2m+1}^*$, and $c_m^{(2m+1)} = 1$. (a) Using a “central” partitioning of \mathbf{R}_{2m+3} and the persymmetry property of Toeplitz matrices, develop a recursion to determine \mathbf{c}_{2m+3} from \mathbf{c}_{2m+1} . (b) Develop a complete order-recursive algorithm for the computation of $\{\mathbf{c}_{2m+1}, P_{2m+1}\}_0^M$ (see Kok et al. 1993).

7.39 Using the triangular decomposition of a Toeplitz correlation matrix, show that

- (a) the forward prediction errors of various orders and at the same time instant, that is,

$$\mathbf{e}^f(n) = [e_0^f(n) \ e_1^f(n) \ \cdots \ e_m^f(n)]^T$$

are correlated.

Using the definition of $\{e_k^f(n)\}_{k=0}^m$, we obtain

$$\mathbf{e}^f(n) = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & a_1^{(1)*} & 0 & \cdots & 0 \\ 1 & a_1^{(2)*} & a_2^{(2)*} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & a_1^{(M)*} & a_2^{(M)*} & \cdots & a_M^{(M)*} \end{bmatrix} \begin{bmatrix} x(n) \\ x(n-1) \\ x(n-2) \\ \vdots \\ x(n-M) \end{bmatrix} \triangleq \bar{\mathbf{A}}_{M+1} \mathbf{x}_{M+1}(n)$$

Hence

$$E[\mathbf{e}^f(n)\mathbf{e}^{fH}(n)] = \bar{\mathbf{A}}_{M+1} E[\mathbf{x}_{M+1}(n)\mathbf{x}_{M+1}(n)] \bar{\mathbf{A}}_{M+1}^H = \bar{\mathbf{A}}_{M+1} \mathbf{R}_{M+1} \bar{\mathbf{A}}_{M+1}^H$$

Using triangular decomposition of $\mathbf{R}_{M+1} = \mathbf{A}_{M+1}^{-1} \bar{\mathbf{D}}_{M+1} \mathbf{A}_{M+1}^{-H}$ from (7.7.8), we have

$$E[\mathbf{e}^f(n)\mathbf{e}^{fH}(n)] = \bar{\mathbf{A}}_{M+1} \mathbf{A}_{M+1}^{-1} \bar{\mathbf{D}}_{M+1} \mathbf{A}_{M+1}^{-H} \bar{\mathbf{A}}_{M+1}^H$$

which, in general, is not a diagonal matrix. Hence components of $\mathbf{e}^f(n)$ are correlated.

- (b) the forward prediction errors

$$\bar{\mathbf{e}}^f(n) = [e_M^f(n) \ e_{M-1}^f(n-1) \ \cdots \ e_0^f(n-M)]^T$$

are uncorrelated.

Using (7.3.66)

$$\bar{\mathbf{e}}^f(n) = \mathbf{A}_{M+1} \mathbf{x}_{M+1}(n)$$

Hence

$$E[\bar{\mathbf{e}}^f(n)\bar{\mathbf{e}}^{fH}(n)] = \mathbf{A}_{M+1} E[\mathbf{x}_{M+1}(n)\mathbf{x}_{M+1}(n)] \mathbf{A}_{M+1}^H = \mathbf{A}_{M+1} \mathbf{R}_{M+1} \mathbf{A}_{M+1}^H$$

where

$$\mathbf{A}_{M+1} = \begin{bmatrix} 1 & a_1^{(M)*} & a_2^{(M)*} & \cdots & a_M^{(M)*} \\ 0 & 1 & a_1^{(M-1)*} & \cdots & a_{M-1}^{(M-1)*} \\ 0 & 0 & 1 & \cdots & a_{M-2}^{(M-2)*} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

Using triangular decomposition of $\mathbf{R}_{M+1} = \mathbf{A}_{M+1}^{-1} \bar{\mathbf{D}}_{M+1} \mathbf{A}_{M+1}^{-H}$ from (7.7.8), we have

$$\mathbb{E} [\bar{\mathbf{e}}^f(n) \bar{\mathbf{e}}^{fH}(n)] = \mathbf{A}_{M+1} \mathbf{A}_{M+1}^{-1} \bar{\mathbf{D}}_{M+1} \mathbf{A}_{M+1}^{-H} \mathbf{A}_{M+1}^H = \bar{\mathbf{D}}_{M+1}$$

which is a diagonal matrix. Hence components of $\bar{\mathbf{e}}^f(n)$ are uncorrelated.

7.40 Generalize the inversion algorithm described in Section 7.7.3 to handle Hermitian Toeplitz matrices.

$$\text{Given: } \mathbf{R}_M = \begin{bmatrix} \mathbf{R}_{M-1} & \\ \mathbf{r}^H & \rho \end{bmatrix}, \quad \text{Determine: } \mathbf{Q}_M = \begin{bmatrix} \mathbf{Q}_{M-1} & \mathbf{q} \\ \mathbf{q}^H & q \end{bmatrix}$$

where

$$\begin{aligned} q &= \frac{1}{\rho - \mathbf{r}^H \mathbf{R}_{M-1}^{-1} \mathbf{r}} \triangleq \frac{1}{\rho + \mathbf{r}^H \mathbf{b}}, \quad \mathbf{b} = -\mathbf{R}_{M-1}^{-1} \mathbf{r} \\ \mathbf{q} &= -(\mathbf{R}_{M-1}^{-1} \mathbf{r}) q = \mathbf{b} q \\ \mathbf{Q}_{M-1} &= \mathbf{R}_{M-1}^{-1} + q \mathbf{b} \mathbf{b}^H \end{aligned}$$

Let $P = 1/q$ which gives $\mathbf{q} = \mathbf{b}/P$, $\mathbf{b} = \mathbf{q}P$ and $\mathbf{Q}_{M-1} = \mathbf{R}_{M-1}^{-1} + P \mathbf{q} \mathbf{q}^H$ or

$$\langle \mathbf{Q}_{M-1} \rangle_{ij} = \langle \mathbf{R}_{M-1}^{-1} \rangle_{ij} + P \mathbf{q}_i \mathbf{q}_j^* \quad (1)$$

Using the fact that \mathbf{R}_{M-1} is Hermitian Toeplitz, that is, $\mathbf{J} \mathbf{R}_{M-1} \mathbf{J} = \mathbf{R}_{M-1}^*$ or $\mathbf{R}_{M-1}^{-1} = \mathbf{J} (\mathbf{R}_{M-1}^{-1})^* \mathbf{J}$ (Per-Hermitian), we obtain

$$\langle \mathbf{R}_{M-1}^{-1} \rangle = \left(\langle \mathbf{R}_{M-1}^{-1} \rangle_{M-j, M-i} \right)^*$$

Substituting in (1)

$$\langle \mathbf{Q}_{M-1} \rangle_{ij} = \left(\langle \mathbf{R}_{M-1}^{-1} \rangle_{M-j, M-i} \right)^* + P \mathbf{q}_i \mathbf{q}_j^* \quad (2)$$

But also from (1)

$$\left(\langle \mathbf{Q}_{M-1} \rangle_{M-j, M-i} \right)^* = \left(\langle \mathbf{R}_{M-1}^{-1} \rangle_{M-j, M-i} \right)^* + P (\mathbf{q}_{M-j} \mathbf{q}_{M-i}^*)^* \quad (3)$$

Subtracting (3) from (2)

$$\langle \mathbf{Q}_{M-1} \rangle_{ij} - \left(\langle \mathbf{Q}_{M-1} \rangle_{M-j, M-i} \right)^* = P \mathbf{q}_i \mathbf{q}_j^* - P (\mathbf{q}_{M-j} \mathbf{q}_{M-i}^*)^*$$

or

$$\langle \mathbf{Q}_{M-1} \rangle_{ij} = \left(\langle \mathbf{Q}_{M-1} \rangle_{M-j, M-i} \right)^* + P (\mathbf{q}_i \mathbf{q}_j^* - \mathbf{q}_{M-j}^* \mathbf{q}_{M-i})$$

The Matlab function `invtoeplC` is shown below.

```
function Q = invtoeplC(r,M)
% Computes inverse of a Hermitian Toeplitz matrix R
% Q = invtoeplC(r,M)
% Input: First column of Hermitian Toeplitz matrix R
%         r(m), 0 <= m <= M-1
```

```

% Output: Q = Inverse of R using Section 7.7.3
%
% Programmed by: Vinay K. Ingle, 1999
%
%-----
% Copyright 2000, by Dimitris G. Manolakis, Vinay K. Ingle,
% and Stephen M. Kogon. For use with the book
%
% "Statistical and Adaptive Signal Processing"
%
% McGraw-Hill Higher Education.
%
%-----
%
% Requires the function sympersym

M = length(r);

% Determine vector b using the durbin function
[a,k,Po] = durbin(r,M-1);
b = flipud(a(2:M));
%R = toeplitz(r,conj(r)); b = -R(1:M-1,1:M-1)\flipud(conj(r(2:M)));

% Determine P and q
P = r(1) + (conj(r(M:-1:2)))'*b;
q = b/P;

% Computation of Q matrix using eq. (7.7.24)
QM = zeros(M); % Initialize QM
QM(1:M-1,M) = q; QM(M,M) = 1/P; % Last column of QM
QM = sympersymC(QM,QM(:,M)); % symmetry and persymmetry operations
Q = QM(1:M-1,1:M-1); % Extract Q

Mend = ceil((M+1)/2);
for j = M-1:-1:Mend
    for i = (M-j+1):1:j
        % Use eq. (7.7.24)
        Q(i,j) = conj(Q(M-j,M-i)) - P*(conj(q(M-j))*q(M-i)-q(i)*conj(q(j)));
    end
    % Perform symmetry and persymmetry operations
    QM = sympersymC(QM,Q((M-j+1):1:j,j));
    Q = QM(1:M-1,1:M-1);
end
Q = QM;

```

7.41 Consider the estimation of a constant α from its noisy observations. The signal and observation models are

given by

$$\begin{aligned} y(n+1) &= y(n) \quad n > 0 \quad y(0) = \alpha \\ x(n) &= y(n) + v(n) \quad v(n) \sim \text{WGN}(0, \sigma_v^2) \end{aligned}$$

- (a) Develop scalar Kalman filter equations, assuming the initial condition on the a posteriori error variance $R_{\tilde{y}}(0|0)$ equal to r_0 .

Signal model:

$$y(n) = y(n-1), \quad n > 0, \quad y(0) = \alpha \quad \Rightarrow \quad \mathbf{A}(n-1) \triangleq A = 1, \quad \mathbf{B}(n) \triangleq B = 0$$

Observation model:

$$x(n) = y(n) + v(n) \quad \Rightarrow \quad \mathbf{H}(n) \triangleq H = 1$$

Signal Prediction:

$$\hat{y}(n|n-1) = A\hat{y}(n-1|n-1) = \hat{y}(n-1|n-1), \quad \hat{y}(1|0) = y(0) = \alpha$$

Signal filter:

$$\begin{aligned} \hat{y}(n|n) &= \hat{y}(n|n-1) + K(n) \{x(n) - H\hat{y}(n|n-1)\} \\ &= \hat{y}(n|n-1) + K(n) \{x(n) - \hat{y}(n|n-1)\} \end{aligned}$$

A-priori error covariance:

$$\begin{aligned} R_{\tilde{y}}(n|n-1) &= AR_{\tilde{y}}(n|n-1)A + BR_{\eta}(n)B \\ &= R_{\tilde{y}}(n-1|n-1), \quad R_{\tilde{y}}(0|0) \triangleq r_0 \end{aligned}$$

Kalman gain:

$$\begin{aligned} K(n) &= R_{\tilde{y}}(n|n-1)H [R_{\tilde{y}}(n|n-1) + R_v(n)]^{-1} \\ &= \frac{R_{\tilde{y}}(n|n-1)}{R_{\tilde{y}}(n|n-1) + R_v(n)} = \frac{R_{\tilde{y}}(n|n-1)}{R_{\tilde{y}}(n|n-1) + \sigma_v^2} \end{aligned}$$

A-posteriori error covariance:

$$R_{\tilde{y}}(n|n) = [1 - K(n)H] R_{\tilde{y}}(n|n-1) = [1 - K(n)] R_{\tilde{y}}(n|n-1)$$

- (b) Show that the a posteriori error variance $R_{\tilde{y}}(n|n)$ is given by

$$R_{\tilde{y}}(n|n) = \frac{r_0}{1 + (r_0/\sigma_v^2)n} \tag{1}$$

From part (a) we have

$$K(n) = \frac{R_{\tilde{y}}(n|n-1)}{R_{\tilde{y}}(n|n-1) + \sigma_v^2} = \frac{R_{\tilde{y}}(n-1|n-1)}{R_{\tilde{y}}(n-1|n-1) + \sigma_v^2}$$

with

$$\begin{aligned} R_{\tilde{y}}(n|n) &= [1 - K(n)] R_{\tilde{y}}(n|n-1) = \left[1 - \frac{R_{\tilde{y}}(n-1|n-1)}{R_{\tilde{y}}(n-1|n-1) + \sigma_v^2}\right] R_{\tilde{y}}(n-1|n-1) \\ &= \frac{\sigma_v^2 R_{\tilde{y}}(n-1|n-1)}{R_{\tilde{y}}(n-1|n-1) + \sigma_v^2} \end{aligned}$$

Thus for $n = 1$:

$$K(1) = \frac{r_0}{r_0 + \sigma_v^2} = \frac{\frac{r_0}{\sigma_v^2}}{1 + \frac{r_0}{\sigma_v^2}}, \quad R_{\tilde{y}}(1|1) = \frac{r_0}{1 + \frac{r_0}{\sigma_v^2}}$$

$n = 2$:

$$K(2) = \frac{r_0}{2r_0 + \sigma_v^2} = \frac{\frac{r_0}{\sigma_v^2}}{1 + 2\frac{r_0}{\sigma_v^2}}, \quad R_{\tilde{y}}(2|2) = \frac{r_0}{1 + 2\frac{r_0}{\sigma_v^2}}$$

Hence

$$K(n) = \frac{r_0}{nr_0 + \sigma_v^2} = \frac{\frac{r_0}{\sigma_v^2}}{1 + n\frac{r_0}{\sigma_v^2}}, \quad R_{\tilde{y}}(n|n) = \frac{r_0}{1 + n\frac{r_0}{\sigma_v^2}}$$

(c) Show that the optimal filter for the estimation of the constant α is given by

$$\hat{y}(n) = \hat{y}(n-1) + \frac{r_0/\sigma_v^2}{1 + (r_0/\sigma_v^2)n}[x(n) - \hat{y}(n-1)]$$

Using $K(n)$ from the above part we obtain the desired result.

7.42 Consider a random process with PSD given by

$$R_s(e^{j\omega}) = \frac{4}{2.4661 - 1.629 \cos \omega + 0.81 \cos 2\omega}$$

(a) Using MATLAB, plot the PSD $R_s(e^{j\omega})$ and determine the resonant frequency ω_0 .

```
% Given PSD
w = [0:1:500]*pi/500;
Rs = 4./(2.4661-1.629*cos(w)+0.81*cos(2*w));

% (a) Plot of PSD and Resonant frequency
[maxRs,I_max] = max(Rs); w_max = w(I_max);

Hf_1 = figure('units','SCRUN','position',SCRPOS,...
    'paperunits','PAPUN','paperposition',[0,0,5,3],...
    'numbertitle','off','name','P0742a');
plot(w/pi,Rs,'g','linewidth',1.5); hold on; axis([0,1,0,3.5]);
plot([w_max,w_max]/pi,[0,maxRs],'r:',[w_max/pi],[maxRs],'md');
xlabel('Digital frequency in \pi units','fontsize',8);
ylabel('|\it{R}_\{\it{its}\}( e^{j\omega} )|','fontsize',8);
title('Power Spectral Density','fontsize',10,'fontweight','bold');
text(w_max/pi+0.02,0.5,['Resonant frequency: ',num2str(w_max),' radians']);
```

The plot is shown in Figure 7.42a.

(b) Using spectral factorization, develop a signal model for the process of the form

$$\begin{aligned} \mathbf{y}(n) &= \mathbf{A}\mathbf{y}(n-1) + \mathbf{B}\eta(n) \\ s(n) &= [1 \ 0]\mathbf{y}(n) \end{aligned}$$

where $\mathbf{y}(n)$ is a 2×1 vector, $\eta(n) \sim \text{WGN}(0, 1)$, and \mathbf{A} and \mathbf{B} are matrices with appropriate dimensions.

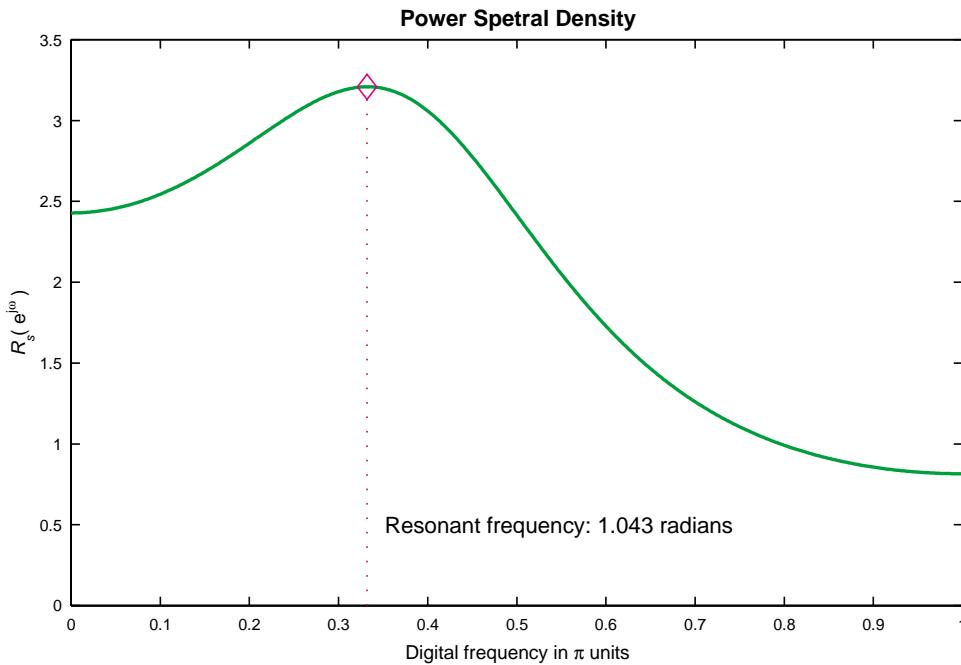


Figure 7.42a: Plot of the PSD $R_s(e^{j\omega})$

Substituting $\cos \omega = \frac{e^{j\omega} + e^{-j\omega}}{2} = \frac{z+z^{-1}}{2} \Big|_{z=e^{j\omega}}$ and $\cos 2\omega = \frac{e^{j2\omega} + e^{-j2\omega}}{2} = \frac{z^2+z^{-2}}{2} \Big|_{z=e^{j\omega}}$ in $R_s(e^{j\omega})$

$$\begin{aligned} R_s(z) &= \frac{4}{2.4661 - 1.629 \frac{z+z^{-1}}{2} + 0.81 \frac{z^2+z^{-2}}{2}} \\ &= \frac{40000z^2}{4050z^4 - 8145z^3 + 24661z^2 - 8145z + 4050} \\ &= \left(\frac{3.1427}{1 - 0.3158z^{-1} + 0.1863z^{-2}} \right) \left(\frac{3.1427}{z^2 - 1.6953z + 5.3674} \right) \end{aligned}$$

Thus the AR model is

$$y(n) = 0.3158 y(n-1) - 0.1863 y(n-2) + 3.1427 \eta(n)$$

or the state equations are:

$$\begin{bmatrix} y(n) \\ y(n-1) \end{bmatrix} = \begin{bmatrix} 0.3158 & -0.1863 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} y(n-1) \\ y(n-2) \end{bmatrix} + \begin{bmatrix} 3.1427 \\ 0 \end{bmatrix} \eta(n)$$

$$\mathbf{y}(n) \triangleq \mathbf{A} \mathbf{y}(n) + \mathbf{B} \eta(n)$$

$$s(n) = y(n) = [1 \ 0] \begin{bmatrix} y(n) \\ y(n-1) \end{bmatrix} \triangleq \mathbf{H} \mathbf{y}(n)$$

```
% (b) Spetctal factorization
B = 8; Rs_den = [0.81,-1.629,4.9322,-1.629,.81];
B = B/Rs_den(1); Rs_den = Rs_den/Rs_den(1);
poles = roots(Rs_den);
a_out = poly(poles(1:2)); a_in = poly(poles(3:4)); b = sqrt(B);
```

```
% Verification
if 0
    N = 8192; eta = randn(N,1);
    sn = filter(b,a_in,eta);
    Nfft = 512; L = 16; w_ham = hamming(L)';
    Fs = 1;
    Rs_hat = psd(sn,Nfft,Fs,w_ham,L/2,'none');
    figure;
    w = [0:1:Nfft/2]*Fs/Nfft; plot(w,Rs_hat); axis([0,Fs/2,0,18]);
end
```

- (c) Let $x(n)$ be the observed values of $s(n)$ given by

$$x(n) = s(n) + v(n) \quad v(n) \sim \text{WGN}(0, 1)$$

Assuming reasonable initial conditions, develop Kalman filter equations and implement them, using MATLAB. Study the performance of the filter by simulating a few sample functions of the signal process $s(n)$ and its observation $x(n)$.

```
% (c) Simulation
```

```
% Generate the AR(2) process s(n)
a = a_in; b = sqrt(B);
N = 100; n = [0:N];
eta = randn(N+1,1);
sn = filter(b,a,eta);
s_std = std(sn);

% Generate vector process y(n)
yn = sn; y_std = s_std;

% Generate the observation process x(n)
v = randn(N+1,1);
yn1 = [0;yn(1:N)];
h1 = 1.0; h2 = 0.0; g1 = sqrt(1);
xn = h1*yn + h2*yn1 + g1*v;
x_std = std(xn);
in_snr = 20*log10(y_std/g1)

% Design KF parameters
A = [-a(2),-a(3);1,0]; B = [b;0];
H = [h1,h2]; G = g1;
Rv = g1*g1; R_eta = eye(1);

% Initialization
y_post = [0;0]; R_post = zeros(2); I = eye(2);
y_hat = zeros(N+1,1); gain = zeros(N+1,2); mse = y_hat;

% Tracking
for n = 0:N
    R_pri = A*R_post*A' + B*R_eta*B';
    y_pri = A*y_post;
```

```

x_pri = H*y_pri;
Rw = H*R_pri*H'+Rv;
K = R_pri*H'*inv(Rw);
y_post = y_pri + K*(xn(n+1) - x_pri);
R_post = (I-K*H)*R_pri;
y_hat(n+1) = y_post(2);
gain(n+1,:) = K';
mse(n+1) = R_post(1,1);
end
y_hat = [y_hat(2:N+1);y_hat(N+1)];
error = yn - y_hat; e_std = std(error);
out_snr = 20*log10(y_std/e_std)

%Plots
Hf_2 = figure('units',SCRUN,'position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',[0,0,5,3],...
    'numbertitle','off','name','P0742c1');

n = 0:N;
plot(n,yn,'g',n,xn,'w:',n,y_hat,'c--','linewidth',1);
ylabel('Amplitude','fontsize',8,'fontname','Helv');
xlabel('n','fontsize',8,'fontname','Helv');
legend('y(n)','x(n)','yhat(n)',0);
set(gca,'xtick',[0:20:N],'ytick',[-10:5:10],'fontname','times','fontsize',6);
title('Estimation of AR(2) Process','fontname','Helv','Fontsize',10);

Hf_3 = figure('units',SCRUN,'position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',[0,0,4,3],...
    'numbertitle','off','name','P0742c2');

subplot(2,1,1);
plot(n,gain(:,1),'g',n,gain(:,2),'r--','linewidth',1); axis([0,N,0,.1]);
legend('K_1(n)','K_2(n)',0);
ylabel('Gains','fontsize',8,'fontname','Helv');
set(gca,'xtick',[0:20:N],'ytick',[0,0.1],'fontname','symbol','fontsize',6);
title('Kalman Gain Values','fontname','Helv','Fontsize',10);

subplot(2,1,2);
plot(n,mse,'g','linewidth',1); axis([0,N,0,1]);
ylabel('MSE','fontsize',8,'fontname','Helv');
set(gca,'xtick',[0:20:N],'ytick',[0,1],'fontname','symbol','fontsize',6);
title('Mean Squared Error','fontname','Helv','Fontsize',10);
xlabel('n','fontsize',8,'fontname','Helv');

```

The plots are shown in Figures 7.42c1 and 7.42c2.

- 7.43** Alternative form of the Kalman filter. A number of different identities and expressions can be obtained for the quantities defining the Kalman filter.

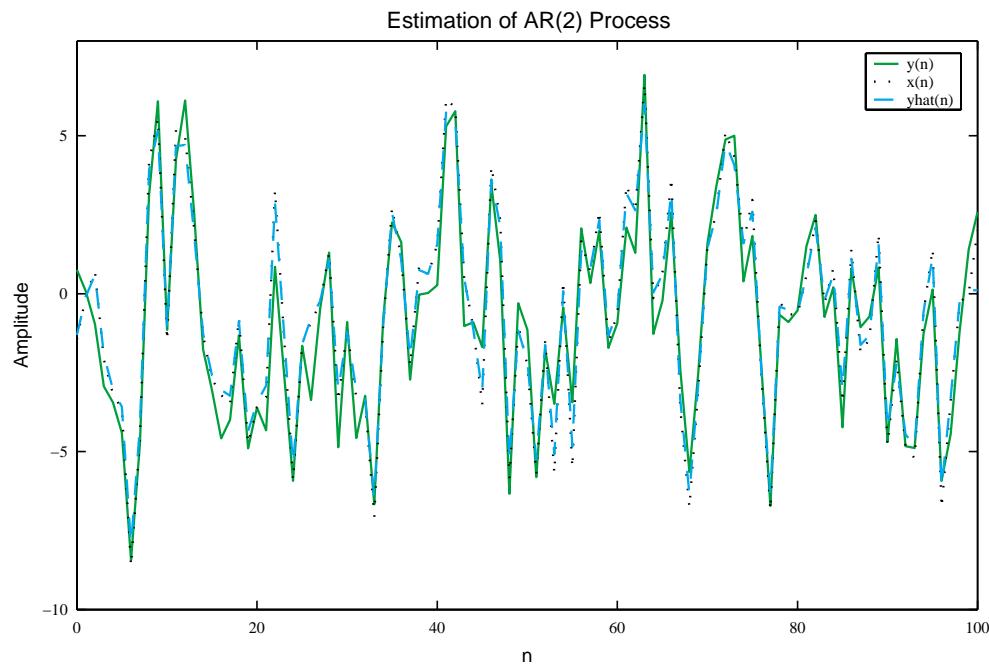


Figure 7.42c1: Plots of true and estimated values

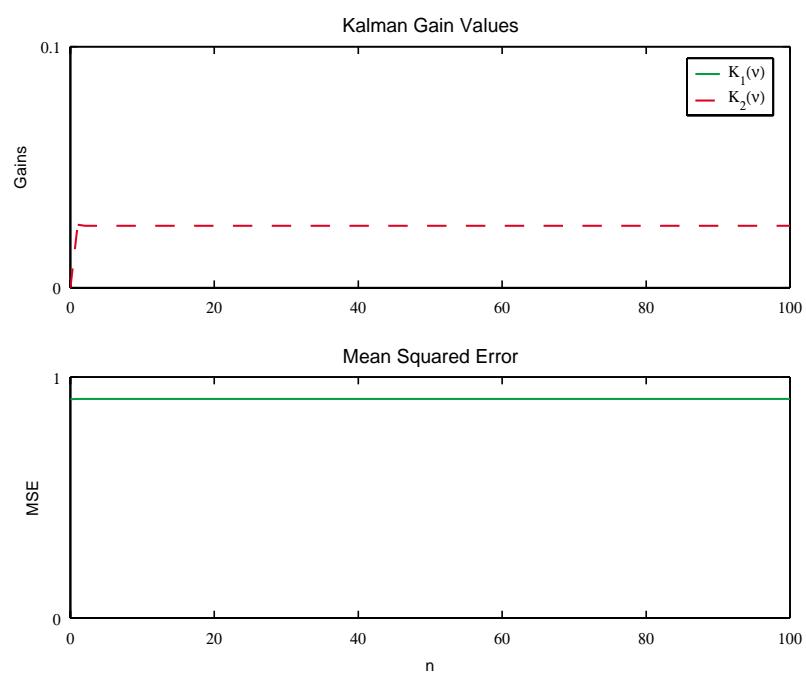


Figure 7.42c2: Plots of gains and MSE

(a) By manipulating the last two equations in (7.8.39) show that

$$\begin{aligned}\mathbf{R}_{\tilde{y}}(n|n) &= \mathbf{R}_{\tilde{y}}(n|n-1) - \mathbf{R}_{\tilde{y}}(n|n-1)\mathbf{H}^H(n) \\ &\quad \times [\mathbf{H}(n)\mathbf{R}_{\tilde{y}}(n|n-1)\mathbf{H}^H(n) + \mathbf{R}_v(n)]^{-1}\mathbf{H}\mathbf{R}_{\tilde{y}}(n|n-1)\end{aligned}\quad (1)$$

Consider

$$\mathbf{R}_{\tilde{y}}(n|n) = [\mathbf{I} - \mathbf{K}(n)\mathbf{H}(n)]\mathbf{R}_{\tilde{y}}(n|n-1)$$

where $\mathbf{K}(n) = \mathbf{R}_{\tilde{y}}(n|n-1)\mathbf{H}^H(n)\mathbf{R}_w^{-1}(n)$. Substituting $\mathbf{K}(n)$ in the above equation

$$\begin{aligned}\mathbf{R}_{\tilde{y}}(n|n) &= [\mathbf{I} - \mathbf{R}_{\tilde{y}}(n|n-1)\mathbf{H}^H(n)\mathbf{R}_w^{-1}(n)\mathbf{H}(n)]\mathbf{R}_{\tilde{y}}(n|n-1) \\ &= \mathbf{R}_{\tilde{y}}(n|n-1) - \mathbf{R}_{\tilde{y}}(n|n-1)\mathbf{H}^H(n)\mathbf{R}_w^{-1}(n)\mathbf{H}(n)\mathbf{R}_{\tilde{y}}(n|n-1)\end{aligned}$$

Now substituting $\mathbf{R}_w(n) = \mathbf{H}(n)\mathbf{R}_{\tilde{y}}(n|n-1)\mathbf{H}^H(n) + \mathbf{R}_v(n)$ from (7.8.29) we obtain the desired result.

(b) If the inverses of $\mathbf{R}_{\tilde{y}}(n|n)$, $\mathbf{R}_{\tilde{y}}(n|n-1)$, and \mathbf{R}_v exist, then show that

$$\mathbf{R}_{\tilde{y}}^{-1}(n|n) = \mathbf{R}_{\tilde{y}}^{-1}(n|n-1) + \mathbf{H}^H(n)\mathbf{R}_v^{-1}(n)\mathbf{H}(n) \quad (2)$$

Consider

$$\mathbf{R}_{\tilde{y}}^{-1}(n|n) = [\mathbf{R}_{\tilde{y}}(n|n-1) - \mathbf{R}_{\tilde{y}}(n|n-1)\mathbf{H}^H(n)\mathbf{R}_w^{-1}(n)\mathbf{H}(n)\mathbf{R}_{\tilde{y}}(n|n-1)]^{-1}$$

Using matrix inversion result

$$[\mathbf{A} - \mathbf{A}\mathbf{B}\mathbf{A}]^{-1} = \mathbf{A}^{-1} + [\mathbf{B}^{-1} - \mathbf{A}]^{-1}$$

we obtain

$$\mathbf{R}_{\tilde{y}}^{-1}(n|n) = \mathbf{R}_{\tilde{y}}^{-1}(n|n-1) + [\mathbf{H}^{-1}(n)\mathbf{R}_w(n)\mathbf{H}^{-H}(n) - \mathbf{R}_{\tilde{y}}(n|n-1)]^{-1}$$

Substituting $\mathbf{R}_w(n) = \mathbf{H}(n)\mathbf{R}_{\tilde{y}}(n|n-1)\mathbf{H}^H(n) + \mathbf{R}_v(n)$, we obtain

$$\begin{aligned}\mathbf{R}_{\tilde{y}}^{-1}(n|n) &= \mathbf{R}_{\tilde{y}}^{-1}(n|n-1) + [\mathbf{H}^{-1}(n)\mathbf{R}_v(n)\mathbf{H}^{-H}(n)]^{-1} \\ &= \mathbf{R}_{\tilde{y}}^{-1}(n|n-1) + \mathbf{H}^H(n)\mathbf{R}_v^{-1}(n)\mathbf{H}^1(n)\end{aligned}$$

This shows that the update of the error covariance matrix does not require the Kalman gain matrix (but does require matrix inverses).

(c) Finally show that the gain matrix is given by

$$\mathbf{K}(n) = \mathbf{R}_{\tilde{y}}(n|n)\mathbf{H}^H(n)\mathbf{R}_v^{-1}(n) \quad (3)$$

which is computed by using the a posteriori error covariance matrix.

Premultiplying (2) by $\mathbf{R}_{\tilde{y}}(n|n)$, we obtain

$$\mathbf{I} = \mathbf{R}_{\tilde{y}}(n|n)\mathbf{R}_{\tilde{y}}^{-1}(n|n-1) + \mathbf{R}_{\tilde{y}}(n|n)\mathbf{H}^H(n)\mathbf{R}_v^{-1}(n)\mathbf{H}^1(n)$$

Postmultiplying by $\mathbf{R}_{\tilde{y}}(n|n-1)$

$$\mathbf{R}_{\tilde{y}}(n|n-1) = \mathbf{R}_{\tilde{y}}(n|n) + \mathbf{R}_{\tilde{y}}(n|n)\mathbf{H}^H(n)\mathbf{R}_v^{-1}(n)\mathbf{H}^1(n)\mathbf{R}_{\tilde{y}}(n|n-1)$$

Comparing with (7.8.39)

$$\mathbf{K}(n) = \mathbf{R}_{\tilde{y}}(n|n)\mathbf{H}^H(n)\mathbf{R}_v^{-1}(n)$$

7.44 In Example 7.8.3 we assumed that only the position measurements were available for estimation. In this problem we will assume that we also have a noisy sensor to measure velocity measurements. Hence the observation model is

$$\mathbf{x}(n) \triangleq \begin{bmatrix} x_p(n) \\ x_v(n) \end{bmatrix} = \begin{bmatrix} y_p(n) + v_1(n) \\ y_v(n) + v_2(n) \end{bmatrix} \quad (1)$$

where $v_1(n)$ and $v_2(n)$ are two independent zero-mean white Gaussian noise sources with variances $\sigma_{v_1}^2$ and $\sigma_{v_2}^2$, respectively.

- (a) Using the state vector model given in Example 7.8.3 and the observation model in (1), develop Kalman filter equations to estimate position and velocity of the object at each n .

The vector state equations are:

$$\begin{aligned} \begin{bmatrix} y_p(n) \\ y_v(n) \end{bmatrix} &= \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} y_p(n-1) \\ y_v(n-1) \end{bmatrix} + \begin{bmatrix} T^2/2 \\ T \end{bmatrix} \eta(n) \\ \mathbf{y}(n) &\triangleq \mathbf{A} \mathbf{y}(n-1) + \mathbf{B} \eta(n) \end{aligned}$$

The output vector equation is:

$$\begin{aligned} \begin{bmatrix} x_p(n) \\ x_v(n) \end{bmatrix} &= \begin{bmatrix} y_p(n) \\ y_v(n) \end{bmatrix} + \begin{bmatrix} v_1(n) \\ v_2(n) \end{bmatrix} = \\ \mathbf{x}(n) &\triangleq \mathbf{H} \mathbf{y}(n) + \mathbf{D} \mathbf{v}(n) \end{aligned}$$

- (b) Using the parameter values

$$T = 0.1 \quad \sigma_{v_1}^2 = \sigma_{v_2}^2 = \sigma_\eta^2 = 0.25 \quad y_p(-1) = 0 \quad y_v(-1) = 1$$

simulate the true and observed positions and velocities of the object. Using your Kalman filter equations, generate plots similar to the ones given in Figures 7.14 and 7.15.

```
% Target and sensor models
T = 0.1;
A = [1 T; 0 1]; % PHI
B = [T*T/2; T]; % G
vareta = 0.25;
H = eye(2);
D = eye(2);
varv1 = 0.25;
varv2 = 0.25;
Rv = diag([varv1,varv2]);

% Generate State and Observation signals
Nt = 100;
eta = sqrt(vareta)*randn(1,1,Nt); y = zeros(2,1,Nt);
y(1:2,1,1) = A*[0;1] + B*eta(1,1,1);
for n = 2:Nt
    y(1:2,1,n) = A*y(1:2,1,n-1) + B*eta(1,1,n);
end
```

```

v1 = sqrt(varv1)*randn(1,Nt);
v2 = sqrt(varv2)*randn(1,Nt);
v = zeros(2,1,Nt); v(1,1,:) = v1(:); v(2,1,:) = v2(:);
for n = 1:Nt
    x(1:2,1,n) = H*y(1:2,1,n) + D*v(1:2,1,n);
end

% A-priori estimates
yhat_ini = [0;0];
R_y = 2*eye(2,2);

% Kalman Filter
[y_hat,GainK,Re_pre,Re_post] = SKF(A,B,vareta,H,D,Rv,x,yhat_ini,R_y);

% Extract quantities for plotting
time = [0:Nt]*T;
yp = y(1,1,:); yp = [0;yp(:)]; % True position
yv = y(2,1,:); yv = [1;yv(:)]; % True velocity
xp = x(1,1,:); xp = [0;xp(:)]; % Noisy position
xv = x(2,1,:); xv = [1;xv(:)]; % Noisy velocity
yp_hat = y_hat(1,1,:); yp_hat = [0;yp_hat(:)]; % Estimated position
yv_hat = y_hat(2,1,:); yv_hat = [1;yv_hat(:)]; % Estimated velocity
Kg1 = GainK(1,1,:); Kg1 = Kg1(:,1); % Kalman Gain (1,1)
Kg2 = GainK(2,2,:); Kg2 = Kg2(:,2); % Kalman Gain (2,2)
for n = 1:Nt
    Rpre(n) = trace(Re_pre(:,:,n));
    Rpost(n) = trace(Re_post(:,:,n));
end

% Plotting
Hf_1 = figure('units','SCRUN','position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',papos);

subplot(2,1,1);
plot(time,yp,'g',time,xp,'r:',time,yp_hat,'m--');
axis([time(1),time(end),-5,15]);
ylabel('Position (meters)', 'fontsize',8);
title('True, Noisy, and Estimated Positions', 'fontsize',10);
legend('True','Noisy','Estimate',4);
set(gca,'xtick',[0:2:10], 'fontsize',6)
set(gca,'ytickmode','auto', 'fontsize',6);

subplot(2,1,2);
plot(time,yv,'g',time,xv,'r:',time,yv_hat,'m--');
axis([0,10,-3,3]);
xlabel('t (sec)', 'fontsize',8); ylabel('velocity (m/sec)', 'fontsize',8);
legend('True','Noisy','Estimate',4);
title('True, Noisy, and Estimated Velocities', 'fontsize',10);

```

```

set(gca,'xtick',[0:2:10],'fontsize',6);
set(gca,'ytickmode','auto','fontsize',6);

Hf_2 = figure('units',SCRUN,'position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',papos);
subplot(2,1,1);
H_kg = plot(time(2:end),Kg1,'go',time(2:end),Kg2,'rd');
set(H_kg,'markersize',3); axis([0,10,0,1]);
title('Kalman gain components','fontsize',10);
legend('K_p','K_v',1); xlabel('t (sec)','fontsize',8);
set(gca,'xtickmode','auto','fontsize',6);
set(gca,'ytick',[0:0.2:1],'fontsize',6);
set(gca,'yticklabel',[0.0';0.2';0.4';0.6';0.8';1.0']);

subplot(2,2,3);
H_pt1 = plot(time(2:41),Rpre(1:40),'ro',time(2:41),Rpost(1:40),'gd');
set(H_pt1,'markersize',3);
legend('a-priori','a-posteriori',1); xlabel('t (sec)','fontsize',8);
axis([0,4.0,0,1]); xlabel('t (sec)','fontsize',8);
%subplot(2,2,3), plot(time,tPm,'.',time,tP,'g.')
title('Trace of covariance matrix','fontsize',10);
set(gca,'xtickmode','auto','fontsize',6);
set(gca,'ytick',[0:0.5:1],'fontsize',6);

subplot(2,2,4);
H_pt2 = plot(time(61:72),Rpre(60:71),'ro',time(61:72),Rpost(60:71),'gd');
set(H_pt2,'markersize',3); hold on;
timenew = [1;1]*time(61:72); timenew = (timenew(:))';
tpmtp = zeros(1,23);
tpmtp(1:2:23) = Rpre(60:71); tpmtp(2:2:24) = Rpost(60:71);
plot(timenew,tpmtp,'w:'); hold off;
legend('a-priori','a-posteriori',1); xlabel('t (sec)','fontsize',8);
axis([5.95,7.05,0.04,0.05]);
xlabel('t (sec)','fontsize',8);
title('Trace of covariance matrix','fontsize',10);
set(gca,'xtick',[6,7],'fontsize',6);
set(gca,'ytick',[0.04:0.005:0.05],'fontsize',6);

```

The plots are shown in Figures 7.44b1 and 7.44b2.

- (c) Discuss the effects of velocity measurements on the estimates.

Clearly, the velocity estimates follow the true velocity values. The error is due to noisy observations.

- 7.45** In this problem, we will assume that the acceleration $y_a(n)$ is an AR(1) process rather than a white noise process. Let $y_a(n)$ be given by

$$y_a(n) = \alpha y_a(n-1) + \eta(n) \quad \eta(n) \sim \text{WGN}(0, \sigma_\eta^2) \quad y_a(-1) = 0 \quad (1)$$

- (a) Augment the state vector $\mathbf{y}(n)$ in (7.8.48), using variable $y_a(n)$, and develop the state vector as well as the observation model, assuming that only the position is measured.

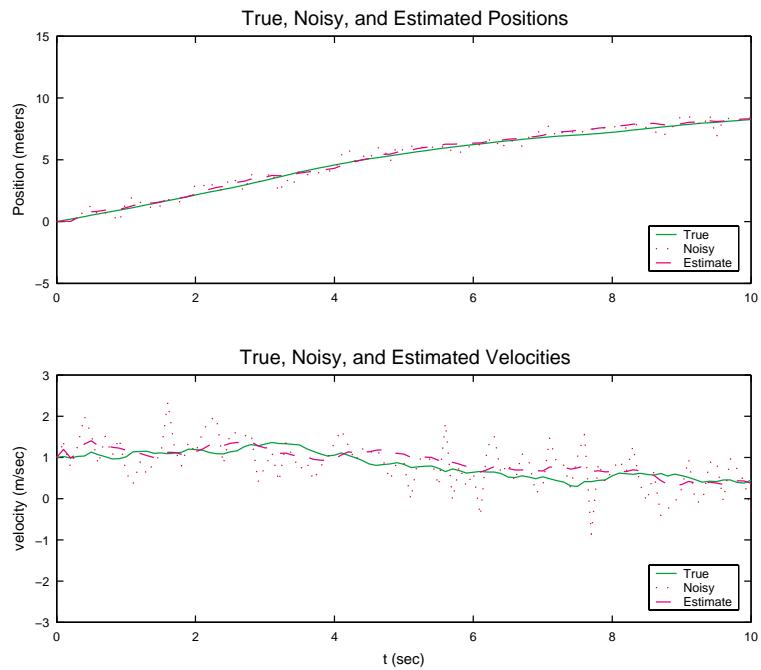


Figure 7.44b1: Plots of position, velocity and acceleration

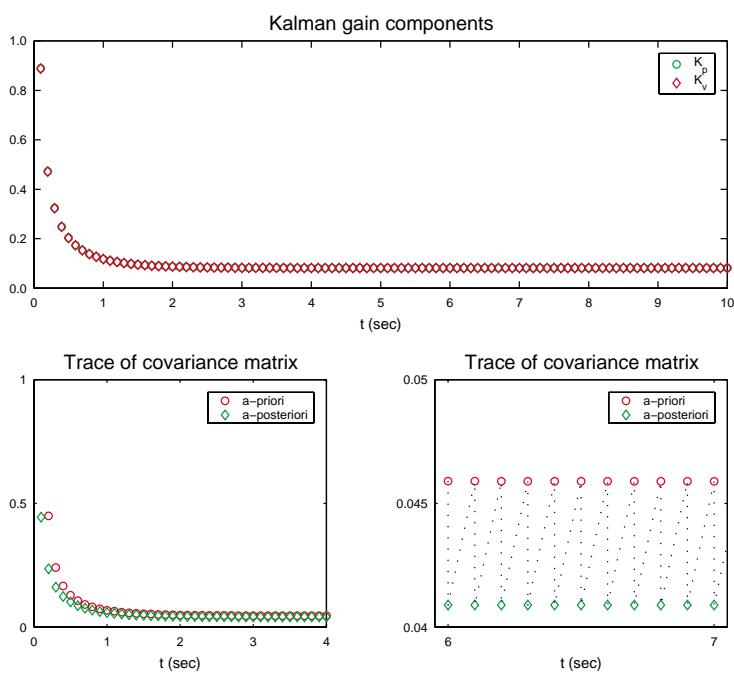


Figure 7.44b2: Plots of gains and covariance traces

The scalar state equations are:

$$\begin{aligned} y_p(n) &= y_p(n-1) + y_v(n-1)T + \frac{1}{2}y_a(n-1)T^2 \\ y_v(n) &= y_v(n-1) + y_a(n-1)T \\ y_a(n) &= \alpha y_a(n-1) + \eta(n) \end{aligned}$$

Hence the vector state equations are:

$$\begin{bmatrix} y_p(n) \\ y_v(n) \\ y_a(n) \end{bmatrix} = \begin{bmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & \alpha \end{bmatrix} \begin{bmatrix} y_p(n-1) \\ y_v(n-1) \\ y_a(n-1) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \eta(n)$$

$$\mathbf{y}(n) \triangleq \mathbf{A} \mathbf{y}(n-1) + \mathbf{B} \eta(n)$$

The output equation is:

$$\begin{aligned} x(n) &= y_p(n) + v(n) = [1 \ 0 \ 0] \mathbf{y}(n) + v(n) \\ &\triangleq \mathbf{H} \mathbf{y}(n) + \mathbf{D} v(n) \end{aligned}$$

(b) Using the above model and the parameter values

$$\begin{aligned} T &= 0.1 & \alpha &= 0.9 & \sigma_v^2 &= \sigma_\eta^2 = 0.25 \\ y_p(-1) &= 0 & y_v(-1) &= 1 \end{aligned}$$

simulate the linear motion of the object. Using Kalman filter equations, estimate the position, velocity, and acceleration values of the object at each n . Generate performance plots similar to the ones given in Figures 7.14 and 7.15.

```
%% (b) AR(1) model for acceleration and observation of position
% Target and sensor models
T = 0.1; alpha = 0.9;

A = [1,T,T*T/2; 0,1,T;0,0,alpha]; K = length(A);
B = [0;0;1];
vareta = 0.25; stdeta = sqrt(vareta); L = 1;
H = [1,0,0]; [M,K] = size(H);
D = eye(1);
varv = 0.25; stdv = sqrt(varv);
%varv2 = 0.25;
Rv = varv*eye(1); P = length(Rv);

% Generate State and Observation signals
Nt = 100; y = zeros(K,1,Nt); x = zeros(M,1,Nt);
eta = stdeta*randn(1,1,Nt); y = zeros(2,1,Nt);
y(1:K,1,1) = A*[0;1;0] + B*eta(1:L,1,1);
for n = 2:Nt
    y(1:K,1,n) = A*y(1:K,1,n-1) + B*eta(1:L,1,n);
end

v1 = stdv*randn(1,Nt);
```

```

v = zeros(1,1,Nt); v(1,1,:) = v1(:);
for n = 1:Nt
    x(1:M,1,n) = H*y(1:K,1,n) + D*v(1:P,1,n);
end

% A-priori estimates
yhat_ini = zeros(K,1); %[0;0;0];
R_y = 2*eye(K);

% Kalman Filter
[y_hat,GainK,Re_pre,Re_post] = SKF(A,B,vareta,H,D,Rv,x,yhat_ini,R_y);

% Extract quantities for plotting
time = [0:Nt]*T;
yp = y(1,1,:); yp = [0;yp(:)]; % True position
yv = y(2,1,:); yv = [1;yv(:)]; % True velocity
ya = y(3,1,:); ya = [0;ya(:)]; % True acceleration
xp = x(1,1,:); xp = [0;xp(:)]; % Noisy position
xv = x(2,1,:); xv = [1;xv(:)]; % Noisy velocity
yp_hat = y_hat(1,1,:); yp_hat = [0;yp_hat(:)]; % Estimated position
yv_hat = y_hat(2,1,:); yv_hat = [1;yv_hat(:)]; % Estimated velocity
ya_hat = y_hat(3,1,:); ya_hat = [1;ya_hat(:)]; % Estimated acceleration

Kg1 = GainK(1,1,:); Kg1 = Kg1(:); % Kalman Gain 1
Kg2 = GainK(2,1,:); Kg2 = Kg2(:); % Kalman Gain 2
Kg3 = GainK(3,1,:); Kg3 = Kg3(:); % Kalman Gain 3

for n = 1:Nt
    Rpre(n) = trace(Re_pre(:,:,n));
    Rpost(n) = trace(Re_post(:,:,n));
end

% Plotting
Hf_1 = figure('units','SCRUN','position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',papos);
xaxismin = time(1); xaxismax = time(end);

subplot(3,1,1);
plot(time,yp,'g',time,xp,'r:',time,yp_hat,'m--');
yaxismin = floor(min(min([yp,xp,yp_hat]))/5)*5;
yaxismax = ceil(max(max([yp,xp,yp_hat]))/5)*5;
axis([xaxismin,xaxismax,yaxismin,yaxismax]);
ylabel('Position (meters)', 'fontsize',8);
title('True, Noisy, and Estimated Positions', 'fontsize',10);
legend('True','Noisy','Estimate',4);
set(gca,'xtick',[0:2:10], 'fontsize',6)
set(gca,'ytickmode','auto', 'fontsize',6);

subplot(3,1,2);

```

```

plot(time,yv,'g',time,yv_hat,'m--');
yaxismin = floor(min(min([yv,yv_hat]))/5)*5;
yaxismax = ceil(max(max([yv,yv_hat]))/5)*5;
axis([xaxismin,xaxismax,yaxismin,yaxismax]);
ylabel('velocity (m/sec)','fontsize',8);
legend('True','Estimate',4);
title('True and Estimated Velocities','fontsize',10);
set(gca,'xtick',[0:2:10],'fontsize',6);
set(gca,'ytickmode','auto','fontsize',6);

subplot(3,1,3);
plot(time,ya,'g',time,ya_hat,'m--');
yaxismin = floor(min(min([ya,ya_hat]))/5)*5;
yaxismax = ceil(max(max([ya,ya_hat]))/5)*5;
axis([xaxismin,xaxismax,yaxismin,yaxismax]);
xlabel('t (sec)','fontsize',8); ylabel('velocity (m/sec)','fontsize',8);
legend('True','Estimate',4);
title('True and Estimated Accelerations','fontsize',10);
set(gca,'xtick',[0:2:10],'fontsize',6);
set(gca,'ytickmode','auto','fontsize',6);

Hf_2 = figure('units',SCRUN,'position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',papos);
subplot(2,1,1);
H_kg = plot(time(2:end),Kg1,'mo',time(2:end),Kg2,'rd',time(2:end),Kg3,'g*');
set(H_kg,'markersize',3); axis([0,10,0,1]);
title('Kalman gain components','fontsize',10);
legend('K_p','K_v','K_a',1); xlabel('t (sec)','fontsize',8);
set(gca,'xtickmode','auto','fontsize',6);
set(gca,'ytick',[0:0.2:1],'fontsize',6);
set(gca,'yticklabel',[0.0';0.2';0.4';0.6';0.8';1.0']);

subplot(2,2,3);
H_pt1 = plot(time(2:41),Rpre(1:40),'mo',time(2:41),Rpost(1:40),'gd');
set(H_pt1,'markersize',3);
legend('a-priori','a-posteriori',1); xlabel('t (sec)','fontsize',8);
%axis([0,4.0,1,6]);
xlabel('t (sec)','fontsize',8);
%subplot(2,2,3), plot(time,tPm,'.',time,tP,'g.')
title('Trace of covariance matrix','fontsize',10);
set(gca,'xtickmode','auto','fontsize',6);
set(gca,'ytick',[0:1:6],'fontsize',6);

subplot(2,2,4);
H_pt2 = plot(time(61:72),Rpre(60:71),'mo',time(61:72),Rpost(60:71),'gd');
set(H_pt2,'markersize',3); hold on;
timenew = [1;1]*time(61:72); timenew = (timenew(:));
tpmtp = zeros(1,23);
tpmtp(1:2:23) = Rpre(60:71); tpmtp(2:2:24) = Rpost(60:71);

```

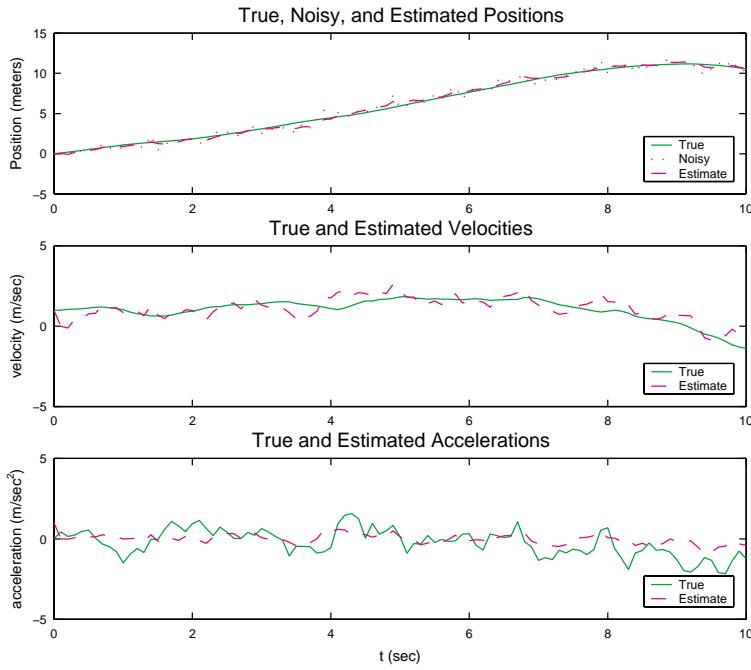


Figure 7.45b1: Plots of position, velocity and acceleration

```

plot(timenew,tpmtp,'w:'); hold off;
legend('a-priori','a-posteriori',1); xlabel('t (sec)', 'fontsize',8);
axis([5.95,7.05,1.55,1.9]);
xlabel('t (sec)', 'fontsize',8);
title('Trace of covariance matrix', 'fontsize',10);
set(gca,'xtick',[6,7], 'fontsize',6);
set(gca,'ytick',[1.5:0.1:1.9], 'fontsize',6);

```

The plots are shown in Figures 7.45b1 and 7.45b2.

- (c) Now assume that noisy measurements of $y_v(n)$ and $y_a(n)$ are also available, that is, the observation model is

$$\mathbf{x}(n) \triangleq \begin{bmatrix} x_p(n) \\ x_v(n) \\ x_a(n) \end{bmatrix} = \begin{bmatrix} y_p(n) + v_1(n) \\ y_v(n) + v_2(n) \\ y_a(n) + v_3(n) \end{bmatrix} \quad (2)$$

where $v_1(n)$, $v_2(n)$, and $v_3(n)$ are IID zero-mean white Gaussian noise sources with variance σ_v^2 . Repeat parts (a) and (b) above.

```

%% (c) AR(1) model for acceleration and
% observations of position, velocity, and acceleration
% Target and sensor models
T = 0.1; alpha = 0.9;

A = [1,T,T*T/2; 0,1,T;0,0,alpha]; K = length(A);
B = [0;0;1];
vareta = 0.25; stdeta = sqrt(vareta); L = length(vareta);
H = eye(3); [M,K] = size(H);
D = eye(3);

```

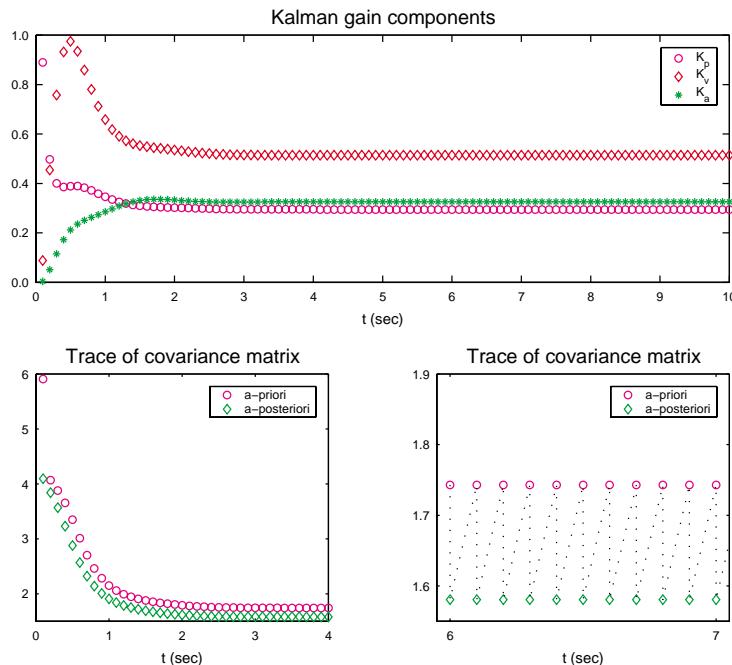


Figure 7.45b2: Plots of gains and covariance traces

```

varv = 0.25; stdv = sqrt(varv);
%varv2 = 0.25;
Rv = varv*eye(3); P = length(Rv);

% Generate State and Observation signals
Nt = 100; y = zeros(K,1,Nt); x = zeros(M,1,Nt);
eta = stdeta*randn(1,1,Nt); y = zeros(2,1,Nt);
y(1:K,1,1) = A*[0;1;0] + B*eta(1:L,1,n);
for n = 2:Nt
    y(1:K,1,n) = A*y(1:K,1,n-1) + B*eta(1:L,1,n);
end

v1 = stdv*randn(1,Nt);
v2 = stdv*randn(1,Nt);
v3 = stdv*randn(1,Nt);
v = zeros(1,1,Nt);
v(1,1,:) = v1(:); v(2,1,:) = v2(:); v(3,1,:) = v3(:);
for n = 1:Nt
    x(1:M,1,n) = H*y(1:K,1,n) + D*v(1:P,1,n);
end

% A-priori estimates
yhat_ini = zeros(K,1); %[0;0;0];
R_y = 2*eye(K);

% Kalman Filter
[y_hat,GainK,Re_pre,Re_post] = skf(A,B,vareta,H,D,Rv,x,yhat_ini,R_y);

```

```
% Extract quantities for plotting
time = [0:Nt]*T;
yp = y(1,1,:); yp = [0;yp(:)]; % True position
yv = y(2,1,:); yv = [1;yv(:)]; % True velocity
ya = y(3,1,:); ya = [0;ya(:)]; % True acceleration
xp = x(1,1,:); xp = [0;xp(:)]; % Noisy position
xv = x(2,1,:); xv = [1;xv(:)]; % Noisy velocity
xa = x(3,1,:); xa = [0;xa(:)]; % Noisy Acceleration
yp_hat = y_hat(1,1,:); yp_hat = [0;yp_hat(:)]; % Estimated position
yv_hat = y_hat(2,1,:); yv_hat = [1;yv_hat(:)]; % Estimated velocity
ya_hat = y_hat(3,1,:); ya_hat = [1;ya_hat(:)]; % Estimated acceleration

Kg1 = GainK(1,1,:); Kg1 = Kg1(:); % Kalman Gain 1
Kg2 = GainK(2,2,:); Kg2 = Kg2(:); % Kalman Gain 2
Kg3 = GainK(3,3,:); Kg3 = Kg3(:); % Kalman Gain 3

for n = 1:Nt
    Rpre(n) = trace(Re_pre(:,:,n));
    Rpost(n) = trace(Re_post(:,:,n));
end

% Plotting
Hf_3 = figure('units','SCRUN','position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',papos);
xaxismin = time(1); xaxismax = time(end);

subplot(3,1,1);
plot(time,yp,'g',time,xp,'r:',time,yp_hat,'m--');
yaxismin = floor(min(min([yp,xp,yp_hat]))/5)*5;
yaxismax = ceil(max(max([yp,xp,yp_hat]))/5)*5;
axis([xaxismin,xaxismax,yaxismin,yaxismax]);
ylabel('Position (meters)', 'fontsize',8);
title('True, Noisy, and Estimated Positions', 'fontsize',10);
legend('True','Noisy','Estimate',4);
set(gca,'xtick',[0:2:10], 'fontsize',6)
set(gca,'ytickmode','auto', 'fontsize',6);

subplot(3,1,2);
plot(time,yv,'g',time,xv,'r:',time,yv_hat,'m--');
yaxismin = floor(min(min([yv,xv,yv_hat]))/5)*5;
yaxismax = ceil(max(max([yv,xv,yv_hat]))/5)*5;
axis([xaxismin,xaxismax,yaxismin,yaxismax]);
xlabel('t (sec)', 'fontsize',8); ylabel('velocity (m/sec)', 'fontsize',8);
legend('True','Noisy','Estimate',4);
title('True, Noisy, and Estimated Velocities', 'fontsize',10);
set(gca,'xtick',[0:2:10], 'fontsize',6);
set(gca,'ytickmode','auto', 'fontsize',6);
```

```

subplot(3,1,3);
plot(time,ya,'g',time,xa,'r:',time,ya_hat,'m--');
yaxismin = floor(min(min([ya,ya_hat]))/5)*5;
yaxismax = ceil(max(max([ya,ya_hat]))/5)*5;
axis([xaxismin,xaxismax,yaxismin,yaxismax]);
xlabel('t (sec)', 'fontsize',8); ylabel('velocity (m/sec)', 'fontsize',8);
legend('True', 'Noisy', 'Estimate',4);
title('True, Noisy, and Estimated Accelerations', 'fontsize',10);
set(gca,'xtick',[0:2:10], 'fontsize',6);
set(gca,'ytickmode','auto', 'fontsize',6);

Hf_4 = figure('units',SCRUN,'position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',papos);
subplot(2,1,1);
H_kg = plot(time(2:end),Kg1,'mo',time(2:end),Kg2,'rd',time(2:end),Kg3,'g*');
set(H_kg,'markersize',3); axis([0,10,0,1]);
title('Kalman gain components', 'fontsize',10);
legend('K_p', 'K_v', 'K_a',1); xlabel('t (sec)', 'fontsize',8);
set(gca,'xtickmode','auto', 'fontsize',6);
set(gca,'ytick',[0:0.2:1], 'fontsize',6);
set(gca,'yticklabel',[0.0';0.2';0.4';0.6';0.8';1.0']);

subplot(2,2,3);
H_pt1 = plot(time(2:41),Rpre(1:40), 'mo', time(2:41),Rpost(1:40), 'gd');
set(H_pt1,'markersize',3);
legend('a-priori', 'a-posteriori',1); xlabel('t (sec)', 'fontsize',8);
axis([0,4.0,0,7]);
xlabel('t (sec)', 'fontsize',8);
title('Trace of covariance matrix', 'fontsize',10);
set(gca,'xtickmode','auto', 'fontsize',6);
set(gca,'ytick',[0:1:7], 'fontsize',6);

subplot(2,2,4);
H_pt2 = plot(time(61:72),Rpre(60:71), 'mo', time(61:72),Rpost(60:71), 'gd');
set(H_pt2,'markersize',3); hold on;
timenew = [1;1]*time(61:72); timenew = (timenew(:))';
tpmtp = zeros(1,23);
tpmtp(1:2:23) = Rpre(60:71); tpmtp(2:2:24) = Rpost(60:71);
plot(timenew,tpmtp, 'w:'); hold off;
legend('a-priori', 'a-posteriori',1); xlabel('t (sec)', 'fontsize',8);
axis([5.95,7.05,0.15,0.6]);
xlabel('t (sec)', 'fontsize',8);
title('Trace of covariance matrix', 'fontsize',10);
set(gca,'xtick',[6,7], 'fontsize',6);
set(gca,'ytick',[0.2:0.1:0.6], 'fontsize',6);

```

The plots are shown in Figures 7.45c1 and 7.45c2.

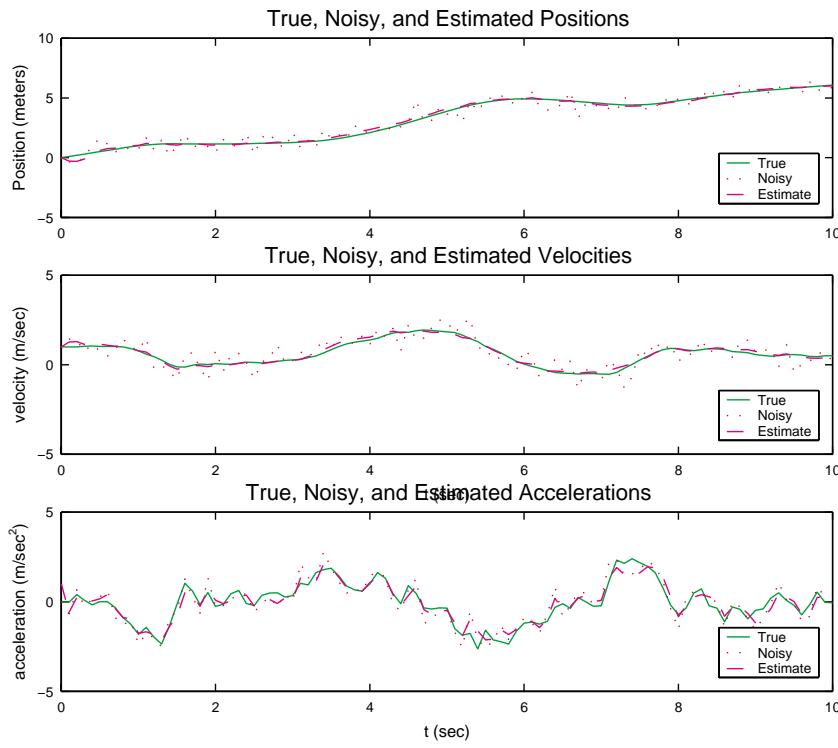


Figure 7.45c1: Plots of position, velocity and acceleration

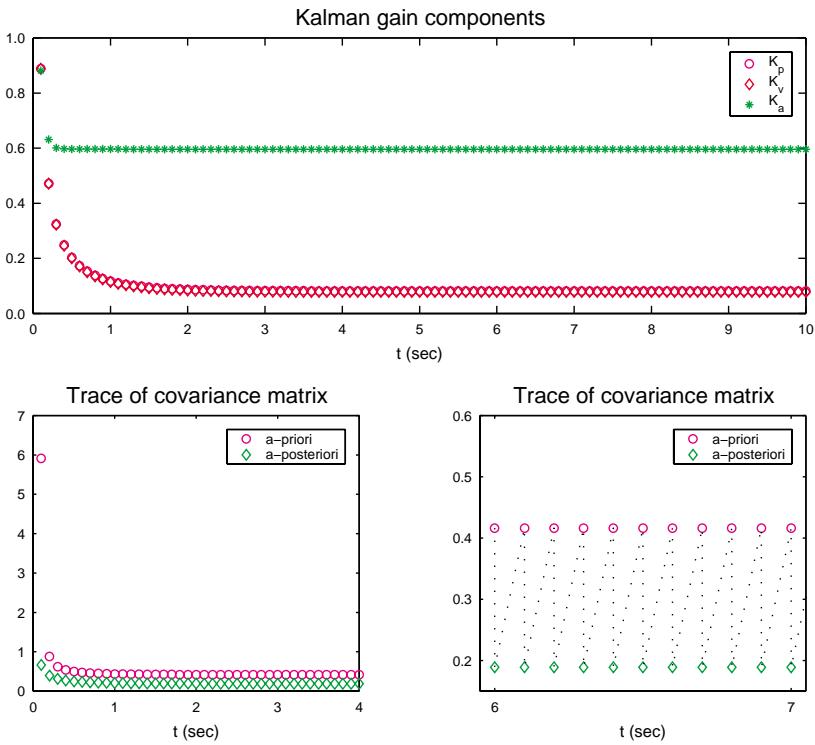


Figure 7.45c2: Plots of gains and covariance traces

Least Squares Filtering and Prediction

- 8.1** By differentiating (8.2.8) with respect to the vector \mathbf{c} , show that the LSE estimator \mathbf{c}_{ls} is given by the solution of the normal equations (8.2.12).

From (8.2.8)

$$E = E_y - \mathbf{c}^H \hat{\mathbf{d}} - \hat{\mathbf{d}}^H \mathbf{c} + \mathbf{c}^H \hat{\mathbf{R}} \mathbf{c}$$

Hence (see Appendix B)

$$0 = \frac{\partial}{\partial \mathbf{c}} E = \frac{\partial}{\partial \mathbf{c}} \left(E_y - \mathbf{c}^H \hat{\mathbf{d}} - \hat{\mathbf{d}}^H \mathbf{c} + \mathbf{c}^H \hat{\mathbf{R}} \mathbf{c} \right) = -\hat{\mathbf{d}}^H + \mathbf{c}^H \hat{\mathbf{R}}$$

gives the desired result.

- 8.2** Let the weighted LSE be given by $E_w = \mathbf{e}^H \mathbf{W} \mathbf{e}$, where \mathbf{W} is a Hermitian positive definite matrix.

- (a) By minimizing E_w with respect to the vector \mathbf{c} , show that the weighted LSE estimator is given by (8.2.35).

Using $\mathbf{e} \triangleq \mathbf{y} - \mathbf{X}\mathbf{c}$, we have

$$\begin{aligned} E_w &= \mathbf{e}^H \mathbf{W} \mathbf{e} = (\mathbf{y}^H - \mathbf{c}^H \mathbf{X}^H) \mathbf{W} (\mathbf{y} - \mathbf{X}\mathbf{c}) = (\mathbf{y}^H - \mathbf{c}^H \mathbf{X}^H) (\mathbf{W}\mathbf{y} - \mathbf{W}\mathbf{X}\mathbf{c}) \\ &= \mathbf{y}^H \mathbf{W}\mathbf{y} - \mathbf{y}^H \mathbf{W}\mathbf{X}\mathbf{c} - \mathbf{c}^H \mathbf{X}^H \mathbf{W}\mathbf{y} + \mathbf{c}^H \mathbf{X}^H \mathbf{W}\mathbf{X}\mathbf{c} \\ &\triangleq E_{wy} - \hat{\mathbf{d}}_w^H \mathbf{c} - \mathbf{c}^H \hat{\mathbf{d}}_w + \mathbf{c}^H \hat{\mathbf{R}}_w \mathbf{c} \end{aligned}$$

where $E_{wy} \triangleq \mathbf{y}^H \mathbf{W}\mathbf{y}$, $\hat{\mathbf{d}}_w \triangleq \mathbf{X}^H \mathbf{W}\mathbf{y}$, and $\hat{\mathbf{R}}_w \triangleq \mathbf{X}^H \mathbf{W}\mathbf{X}$. Since \mathbf{W} is Hermitian and positive-definite, $\hat{\mathbf{R}}_w$ is also Hermitian and positive-definite. Thus the weighted LSE estimator \mathbf{c}_{wls} is provided by the solution of

$$\hat{\mathbf{R}}_w \mathbf{c}_{wls} = \hat{\mathbf{d}}_w$$

or

$$\mathbf{c}_{wls} = \hat{\mathbf{R}}_w^{-1} \hat{\mathbf{d}}_w = (\mathbf{X}^H \mathbf{W}\mathbf{X})^{-1} \mathbf{X}^H \mathbf{W}\mathbf{y}$$

- (b) Using the LDL^H decomposition $\mathbf{W} = \mathbf{L}\mathbf{D}\mathbf{L}^H$, show that the weighted LS criterion corresponds to prefiltering the error or the data.

Using $\mathbf{W} = \mathbf{L}\mathbf{D}\mathbf{L}^H$, $E_w = \mathbf{e}^H \mathbf{L}\mathbf{D}\mathbf{L}^H \mathbf{e}$. Consider $\mathbf{L}^H \mathbf{e}$ which due to the upper triangular structure of \mathbf{L}^H is a prefiltering operation on the error. Similarly from

$$\mathbf{c}_{wls} = (\mathbf{X}^H \mathbf{L}\mathbf{D}\mathbf{L}^H \mathbf{X})^{-1} \mathbf{X}^H \mathbf{L}\mathbf{D}\mathbf{L}^H \mathbf{y}$$

we note that $\mathbf{L}^H \mathbf{X}$ or $\mathbf{L}^H \mathbf{y}$ are prefiltering operations.

- 8.3** Using direct substitution of (8.4.4) into (8.4.5), show that the LS estimator $\mathbf{c}_{ls}^{(i)}$ and the associated LS error $E_{ls}^{(i)}$ are determined by (8.4.5).

Using $\bar{\mathbf{X}} = [\mathbf{X}_1 \quad \mathbf{y} \quad \mathbf{X}_2]$ from (8.4.4), we have in (8.4.5)

$$\begin{bmatrix} \mathbf{0} \\ E_{ls}^{(i)} \\ \mathbf{0} \end{bmatrix} = (\bar{\mathbf{X}}^H \bar{\mathbf{X}}) \mathbf{c}_{ls}^{(i)} = \begin{bmatrix} \mathbf{X}_1^H \\ \mathbf{y}^H \\ \mathbf{X}_2^H \end{bmatrix} [\mathbf{X}_1 \quad \mathbf{y} \quad \mathbf{X}_2] \mathbf{c}_{ls}^{(i)}$$

$$= \begin{bmatrix} \mathbf{X}_1^H \mathbf{X}_1 & \mathbf{X}_1^H \mathbf{y} & \mathbf{X}_1^H \mathbf{X}_2 \\ \mathbf{y}^H \mathbf{X}_1 & \mathbf{y}^H \mathbf{y} & \mathbf{y}^H \mathbf{X}_2 \\ \mathbf{X}_2^H \mathbf{X}_1 & \mathbf{X}_2^H \mathbf{y} & \mathbf{X}_2^H \mathbf{X}_2 \end{bmatrix} \triangleq \begin{bmatrix} \mathbf{R}_{11} & \mathbf{r}_1 & \mathbf{R}_{12} \\ \mathbf{r}_1^H & E_y & \mathbf{r}_2^H \\ \mathbf{R}_{12}^H & \mathbf{r}_2 & \mathbf{R}_{22} \end{bmatrix}$$

Express $\mathbf{c}_{ls}^{(i)}$ as $\mathbf{c}_{ls}^{(i)} = [c_{ls_1} \quad 1 \quad c_{ls_2}]^T$ so that

$$\begin{bmatrix} \mathbf{R}_{11} & \mathbf{r}_1 & \mathbf{R}_{12} \\ \mathbf{r}_1^H & E_y & \mathbf{r}_2^H \\ \mathbf{R}_{12}^H & \mathbf{r}_2 & \mathbf{R}_{22} \end{bmatrix} \begin{bmatrix} c_{ls_1} \\ 1 \\ c_{ls_2} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ E_{ls}^{(i)} \\ \mathbf{0} \end{bmatrix}$$

Following the parallels of (6.5.4) and (6.5.6) with the above equation (or using orthogonality) we obtain

$$\begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{R}_{12}^H & \mathbf{R}_{22} \end{bmatrix} \begin{bmatrix} c_{ls_1} \\ c_{ls_2} \end{bmatrix} = -\begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \end{bmatrix}$$

and

$$E_{ls}^{(i)} = \mathbf{r}_1^H \mathbf{c}_{ls_1} + E_y + \mathbf{r}_2^H \mathbf{c}_{ls_2}$$

which are equations for the optimal LS estimator and design.

8.4 Consider a linear system described by the difference equation $y(n) = 0.9y(n-1) + 0.1x(n-1) + v(n)$, where $x(n)$ is the input signal, $y(n)$ is the output signal, and $v(n)$ is an output disturbance. Suppose that we have collected $N = 1000$ samples of input-output data and that we wish to estimate the system coefficients, using the LS criterion with no windowing. Determine the coefficients of the model $y(n) = ay(n-1) + dx(n-1)$ and their estimated covariance matrix $\hat{\sigma}_e^2 \hat{\mathbf{R}}^{-1}$ when

(a) $x(n) \sim \text{WGN}(0, 1)$ and $v(n) \sim \text{WGN}(0, 1)$

```
% part a
sigmaX=1; sigmaV=1;

B=[0 0.1]; A=[1 0.9];

x = sqrt(sigmaX)*randn(1000,1);
y = filter(B,A,x) + sqrt(sigmaV)*randn;

[R_hat,d_hat] = lsmatvec('nowi',x,2,y)
R_hat =
890.9536 11.7168
11.7168 891.0335
d_hat =
-31.4221
-28.2493
cls = inv(R_hat)*d_hat
c_ls =
```

```

-0.0349
-0.0312

Ey = y'*y;
Els = Ey-d_hat'*cls;

N = 1000; M = 2;
sigmaE = Els/(N-M)
sigmaE =
0.5557
cov = sigmaE*inv(R_hat)
cov =
1.0e-03 *
0.6238 -0.0082
-0.0082 0.6237

```

- (b) $x(n) \sim \text{WGN}(0, 1)$ and $v(n) = 0.8v(n-1) + w(n)$ is an AR(1) process with $w(n) \sim \text{WGN}(0, 1)$. Comment upon the quality of the obtained estimates by comparing the matrices $\hat{\sigma}_e^2 \hat{\mathbf{R}}^{-1}$ obtained in each case.

```

% part b
sigmaW = 1;
w = sqrt(sigmaW)*randn(1000,1);
v = filter([1], [1 0.8], w);
y = filter(B, A, x) + v;

[R_hat, d_hat] = lsmatvec('nowi', x, 2, y)
R_hat =
890.9536 11.7168
11.7168 891.0335
d_hat =
-39.7950
-40.2742
cls = inv(R_hat)*d_hat
cls =
-0.0441
-0.0446
Ey = y'*y;
Els = Ey-d_hat'*cls;

N = 1000; M=2;
sigmaE = Els/(N-M)
sigmaE =
2.6554
cov = sigmaE*inv(R_hat)
cov =
0.0030 -0.0000
-0.0000 0.0030

```

- 8.5** Use Lagrange multipliers to show that Equation (8.4.13) provides the minimum of (8.4.8) under the constraint

(8.4.9).

- 8.6** If full windowing is used in LS, then the autocorrelation matrix is Toeplitz. Using this fact, show that in the combined FBLP predictor is given by

$$\mathbf{a}^{\text{fb}} = \frac{1}{2} (\mathbf{a} + \mathbf{j}\mathbf{b}^*)$$

- 8.7** Consider the noncausal “middle” sample linear signal estimator specified by (8.4.1) with $M = 2L$ and $i = L$. Then

$$e^{(L)}(n) = \sum_{k=0}^{2L} c_k^{(L)*} x(n-k) = \mathbf{c}^{(L)\text{H}} \bar{\mathbf{X}}(n), \quad c_L^{(L)} = 1$$

- (a) Show that if we apply full windowing to the data matrix, the resulting signal estimator is conjugate symmetric, that is, $\mathbf{c}^{(L)} = \mathbf{J}\mathbf{c}^{(L)*}$. This property does not hold for any other windowing method.

We have the vector equation

$$\mathbf{e}^{(L)} = \bar{\mathbf{X}}\mathbf{c}^{(L)}$$

in which $\bar{\mathbf{X}}$ is a data matrix with full windowing. Hence $\bar{\mathbf{R}} \triangleq \bar{\mathbf{X}}^H \bar{\mathbf{X}}$ is a Hermitian Toeplitz matrix, that is,

$$\bar{\mathbf{R}}\mathbf{J} = \mathbf{J}\bar{\mathbf{R}}^* \text{ or } \bar{\mathbf{R}} = \mathbf{J}\bar{\mathbf{R}}^*\mathbf{J}$$

Thus in (8.4.5) we have

$$\bar{\mathbf{R}}\mathbf{c}_{\text{ls}}^{(L)} = \mathbf{J}\bar{\mathbf{R}}^*\mathbf{J}\mathbf{c}_{\text{ls}}^{(L)} = \begin{bmatrix} \mathbf{0} \\ E_{\text{ls}}^{(L)} \\ \mathbf{0} \end{bmatrix}$$

Multiplying both sides by \mathbf{J} and taking complex-conjugate, we obtain

$$\bar{\mathbf{R}}^*\mathbf{J}\mathbf{c}_{\text{ls}}^{(L)} = \begin{bmatrix} \mathbf{0} \\ E_{\text{ls}}^{(L)} \\ \mathbf{0} \end{bmatrix} = \bar{\mathbf{R}}\mathbf{c}_{\text{ls}}^{(L)}$$

since $\mathbf{J}\mathbf{J}^* = \mathbf{I}$ and $E_{\text{ls}}^{(L)}$ is real-valued. Thus $\mathbf{c}_{\text{ls}}^{(L)} = \mathbf{J}\mathbf{c}_{\text{ls}}^{(L)}$ and this is possible only with a full windowing.

- (b) Derive the normal equations for the signal estimator that minimizes the total squared error $E^{(L)} = \|\mathbf{e}^{(L)}\|^2$ under the constraint $\mathbf{c}^{(L)} = \mathbf{J}\mathbf{c}^{(L)*}$.

In this case we assume that $\bar{\mathbf{R}} = \bar{\mathbf{X}}^H \bar{\mathbf{X}}$ is not Toeplitz but only Hermitian. Thus we want to minimize

$$E^{(L)} = \mathbf{e}^{(L)\text{H}} \mathbf{e}^{(L)}, \quad \mathbf{e}^{(L)} = \bar{\mathbf{X}}\mathbf{c}^{(L)}$$

subject to the constraint $\mathbf{c}^{(L)} = \mathbf{J}\mathbf{c}^{(L)}$. We will prove the real-valued case. Consider partitiong of $\mathbf{e}^{(L)} = \bar{\mathbf{X}}\mathbf{c}^{(L)}$ as

$$\mathbf{e}^{(L)} = \begin{bmatrix} \bar{\mathbf{X}}_1 & \mathbf{y} & \bar{\mathbf{X}}_2 \end{bmatrix} \begin{bmatrix} \mathbf{c}_1^{(L)} \\ 1 \\ \mathbf{c}_2^{(L)} \end{bmatrix}$$

where $\mathbf{y} = \bar{\mathbf{x}}(n - L)$ and $\mathbf{c}_2^{(L)} = \mathbf{J}\mathbf{c}_1^{(L)}$ (using the constraint). Thus

$$\begin{aligned}\mathbf{e}^{(L)} &= \bar{\mathbf{X}}_1\mathbf{c}_1^{(L)} + \mathbf{y} + \bar{\mathbf{X}}_2\mathbf{c}_2^{(L)} = \bar{\mathbf{X}}_1\mathbf{c}_1^{(L)} + \mathbf{y} + \bar{\mathbf{X}}_2\mathbf{J}\mathbf{c}_1^{(L)} = (\bar{\mathbf{X}}_1 + \bar{\mathbf{X}}_2\mathbf{J})\mathbf{c}_1^{(L)} + \mathbf{y} \\ &\triangleq \mathbf{W}\mathbf{c}_1^{(L)} + \mathbf{y}\end{aligned}$$

Thus normal equations are given by

$$\mathbf{W}^T\mathbf{e}^{(L)} = \mathbf{W}^T\mathbf{W}\mathbf{c}_1^{(L)} + \mathbf{W}^T\mathbf{y} = \mathbf{0}$$

or

$$(\bar{\mathbf{X}}_1 + \bar{\mathbf{X}}_2\mathbf{J})^T(\bar{\mathbf{X}}_1 + \bar{\mathbf{X}}_2\mathbf{J})\mathbf{c}_1^{(L)} + (\bar{\mathbf{X}}_1 + \bar{\mathbf{X}}_2\mathbf{J})^T\mathbf{y} = \mathbf{0} = (\bar{\mathbf{X}}_1^T + \mathbf{J}\bar{\mathbf{X}}_2^T)(\bar{\mathbf{X}}_1 + \bar{\mathbf{X}}_2\mathbf{J})\mathbf{c}_1^{(L)} + (\bar{\mathbf{X}}_1^T + \bar{\mathbf{X}}_2^T\mathbf{J})\mathbf{y}$$

or

$$(\bar{\mathbf{X}}_1^T\bar{\mathbf{X}}_1 + \bar{\mathbf{X}}_1^T\bar{\mathbf{X}}_2\mathbf{J} + \mathbf{J}\bar{\mathbf{X}}_2^T\bar{\mathbf{X}}_1 + \mathbf{J}\bar{\mathbf{X}}_2^T\bar{\mathbf{X}}_2\mathbf{J})\mathbf{c}_1^{(L)} + (\bar{\mathbf{X}}_1 + \bar{\mathbf{X}}_2\mathbf{J})\mathbf{y} = \mathbf{0} \quad (1)$$

Using another form of normal equations

$$\bar{\mathbf{X}}^T\bar{\mathbf{X}} \begin{bmatrix} \mathbf{c}_1^{(L)} \\ 1 \\ \mathbf{J}\mathbf{c}_1^{(L)} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{X}}_1^T\bar{\mathbf{X}}_1\mathbf{c}_1^{(L)} + \bar{\mathbf{X}}_1^T\bar{\mathbf{X}}_2\mathbf{J}\mathbf{c}_1^{(L)} + \bar{\mathbf{X}}_1^T\mathbf{y} \\ E_{ls}^{(L)} \\ \bar{\mathbf{X}}_2^T\bar{\mathbf{X}}_1\mathbf{c}_1^{(L)} + \bar{\mathbf{X}}_2^T\mathbf{y} + \bar{\mathbf{X}}_2^T\bar{\mathbf{X}}_2\mathbf{J}\mathbf{c}_1^{(L)} \end{bmatrix}$$

and since the first and the last elements (after multiplied by \mathbf{J}) of the right-hand side vector add to zero due to (1), we obtain

$$\bar{\mathbf{X}}^T\bar{\mathbf{X}}\mathbf{c}^{(L)} + \mathbf{J}\bar{\mathbf{X}}^T\bar{\mathbf{X}}\mathbf{c}^{(L)} = \begin{bmatrix} \mathbf{0} \\ 2E^{(L)} \\ \mathbf{0} \end{bmatrix}$$

or

$$[\bar{\mathbf{X}}^T\bar{\mathbf{X}} + \mathbf{J}\bar{\mathbf{X}}^T\bar{\mathbf{X}}]\mathbf{c}^{(L)} = \begin{bmatrix} \mathbf{0} \\ 2E^{(L)} \\ \mathbf{0} \end{bmatrix}$$

which are the normal equations. The complex-valued proof is similar.

- (c) Show that if we enforce the normal equation matrix to be centro-Hermitian, that is, we use the normal equations

$$[\bar{\mathbf{X}}^H\bar{\mathbf{X}} + \mathbf{J}\bar{\mathbf{X}}^T\bar{\mathbf{X}}^*\mathbf{J}]\mathbf{c}^{(L)} = \begin{bmatrix} \mathbf{0} \\ 2E^{(L)} \\ \mathbf{0} \end{bmatrix}$$

then the resulting signal smoother is conjugate symmetric.

In this case $\mathbf{W} \triangleq (\bar{\mathbf{X}}^H\bar{\mathbf{X}} + \mathbf{J}\bar{\mathbf{X}}^T\bar{\mathbf{X}}^*\mathbf{J})$ is Hermitian Toeplitz, that is,

$$\mathbf{J}\mathbf{W}\mathbf{J} = (\mathbf{J}\bar{\mathbf{X}}^H\bar{\mathbf{X}}\mathbf{J} + \bar{\mathbf{X}}^T\bar{\mathbf{X}}^*\mathbf{J}) = (\mathbf{J}\bar{\mathbf{X}}^T\bar{\mathbf{X}}^*\mathbf{J} + \bar{\mathbf{X}}^H\bar{\mathbf{X}})^* = \mathbf{W}^*$$

Then from part (a) above, we have $\mathbf{c}_{ls}^{(L)} = \mathbf{J}\mathbf{c}_{ls}^{(L)}$.

(d) Illustrate parts (a) to (c), using the data matrix

$$\mathbf{X} = \begin{bmatrix} 3 & 2 & 1 \\ 4 & 3 & 2 \\ \vdots & 4 & 3 \\ 1001 & \vdots & 4 \\ \vdots & 1001 & \vdots \\ 3 & \vdots & 1001 \\ 2 & 3 & \vdots \\ 1 & 2 & 3 \end{bmatrix}$$

and check which smoother provides the smallest total squared error. Try to justify the obtained answer.

```
L = 2; M = 2*L;
N = 1000; x = [1:N,N+1,N:-1:1]'; N = length(x); %x = x + 0.5*(rand(N,1)-0.5);

% Symmetric Case (Full Windowing)
disp('Symmetric Case (Full Windowing)');
Xbar = toeplitz([conj(x);zeros(M,1)], [conj(x(1)),zeros(1,M)]);
y = Xbar(:,L+1); X = [Xbar(:,1:L),Xbar(:,L+2:M+1)];

R = X'*X;
r = X'*y;
c = -R\r;
c = [c(1:L);1;c(L+1:end)];
Els = Xbar'*Xbar*c, Els = real(Els(L+1));
e = Xbar*c; Els1 = e'*e;

% Nonsymmetric case (No windowing)
disp('Nonsymmetric case (No windowing)');
Xbar = toeplitz(x(M+1:end),fliplr(x(1:M+1)));
y = Xbar(:,L+1); X = [Xbar(:,1:L),Xbar(:,L+2:M+1)];

R = X'*X;
r = X'*y;
c = -R\r;
c = [c(1:L);1;c(L+1:end)];
Els = Xbar'*Xbar*c, Els = real(Els(L+1));
e = Xbar*c; Els2 = e'*e;

% No windowing but forced coefficient conjugate symmetry
disp('No windowing but forced coefficient conjugate symmetry');
y = Xbar(:,L+1); X = [Xbar(:,1:L)+(fliplr(Xbar(:,L+2:M+1)))];

R = X'*X;
r = X'*y;
c = -R\r;
c = [c(1:L);1;(c(L:-1:1))]; %c = [c(1:L);1;c(L+1:end)]
```

```

Els = (Xbar'*Xbar + flipud(Xbar'*Xbar))*c/2, Els = real(Els(L+1));
e = Xbar*c; Els3 = e'*e;
%% Complex-valued approach
y = Xbar(:,L+1);
X = [Xbar(:,1:L)+(fliplr(Xbar(:,L+2:M+1))), ...
      j*(Xbar(:,1:L)-(fliplr(Xbar(:,L+2:M+1))))];
R = X'*X;
r = X'*y;
c = -R\r; c = c(1:L) + j*c(L+1:end);
c = [c(1:L);1;conj(c(L:-1:1))]; %c = [c(1:L);1;c(L+1:end)]
e = Xbar*c; Els3 = e'*e;

% No windowing but forced centro-Hermitian normal equation matrix
disp('No windowing but forced centro-Hermitian normal equation matrix');
Xbar1 = fliplr(conj(Xbar));
R = Xbar'*Xbar + Xbar1'*Xbar1;
R1 = [R(1:L,1:L),R(1:L,L+2:end);R(L+2:end,1:L),R(L+2:end,L+2:end)];
r1 = [R(1:L,L+1);R(L+2:end,L+1)];
c = -R1\r1;
c = [c(1:L);1;c(L+1:end)];
Els = R*c/2, Els = real(Els(L+1));
e = Xbar*c; Els4 = e'*e;

[Els,I] = sort([Els1,Els2,Els3,Els4])
Els =
    0.6667    0.6667    0.6667    1.0000
I =
    3        2        4        1

```

Thus the full-windowing gives the worst error.

8.8 A useful impulse response for some geophysical signal processing applications is the Mexican hat wavelet

$$g(t) = \frac{2}{\sqrt{3}}\pi^{-1/4}(1-t^2)e^{-t^2/2}$$

which is the second derivative of a Gaussian pulse.

- (a) Plot the wavelet $g(t)$ and the magnitude and phase of its Fourier transform.

```

% Mexican-Hat function:
tmax = 5; t = -tmax:0.05:tmax; t2 = t.*t;
gt = (2/sqrt(3))*pi^(-1/4)*(1-t2).*exp(-t2/2);

% Fourier Transform of g(t): G(F)
F = -1:0.01:1; w = 2*pi*F; w2 = w.*w;
Gf = sqrt(8/3)*(pi^(1/4))*w2.*exp(-w2/2);
I1 = max(find(Gf(1:51) <= 0.001)); F1 = F(I1);
I2 = min(find(Gf(151:201) <= 0.001)); F2 = F(I2+150);

```

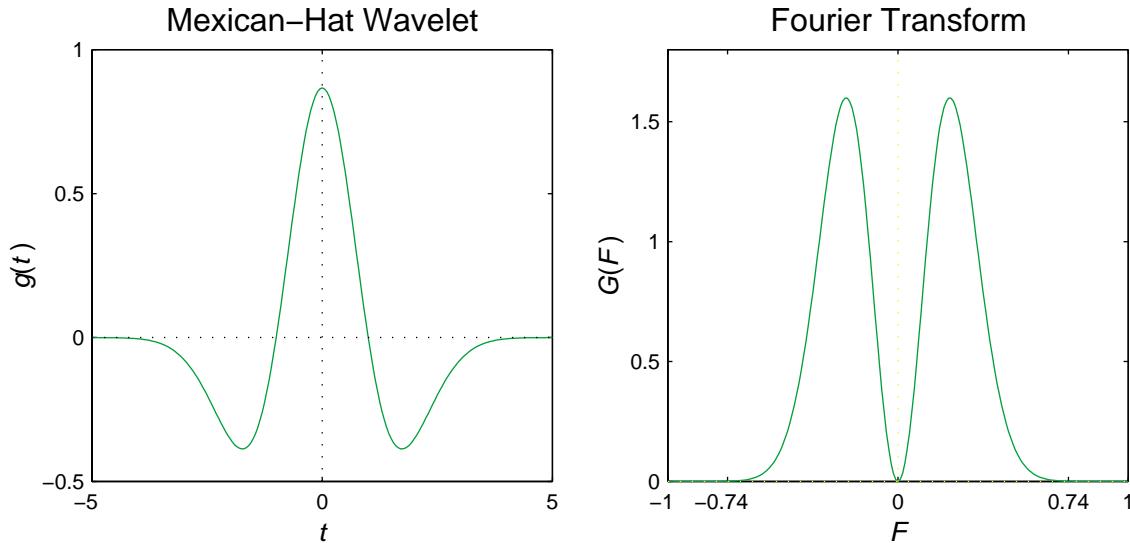


Figure 8.8a: Plots of $g(t)$ and $G(f)$

```
% (a) Plot of g(t) and its Fourier Transform G(F)
Hf_1 = figure('units','inches','position',[1,1,6,3],...
    'paperunits','inches','paperposition',[0,0,6,3],...
    'numbertitle','off','name','P0808a');

subplot('position',[0.08,0.15,0.4,0.75]);
plot(t,gt,'g',[-tmax,tmax],[0,0],[0,0],[-.5,1],'w:');
axis([-tmax,tmax,-0.5,1]);
title('Mexican-Hat Wavelet','fontsize',12);
xlabel('\it{t}', 'fontsize',10); ylabel('{\it{g}}({\it{t}})', 'fontsize',10);

subplot('position',[0.58,0.15,0.4,0.75]);
plot(F,Gf,'g',[-1,1],[0,0],[0,0],[0,1.8],'w:');
axis([-1,1,0,1.8]);
title('Fourier Transform','fontsize',12);
xlabel('\it{F}', 'fontsize',10); ylabel('{\it{G}}({\it{F}})', 'fontsize',10);
set(gca,'xtick',[-1,F1,0,F2,1],'ytick',[0:0.5:2]);
```

The plot is shown in Figure 8.8a.

- (b) By examining the spectrum of the wavelet, determine a reasonable sampling frequency F_s .

```
%% (b) A reasonable sampling frequency (for  $G(F) \leq 0.001$ )
Fs = 2*F2; Ts = 1/Fs;
Fs =
1.4800
```

- (c) Design an optimum LS inverse FIR filter for the discrete-time wavelet $g(nT)$, where $T = 1/F_s$. Determine a reasonable value for M by plotting the LSE E_M as a function of order M . Investigate whether we can improve the inverse filter by introducing some delay n_0 . Determine the best value of n_0 and plot the impulse response of the resulting filter and the combined impulse response $g(n) * h(n - n_0)$, which should resemble an impulse.

```
%%(c) Optimum LS Inverse filter
M = 14; % order of the filter
```

```
% Generate sample of g(t) as data
t = 0:Ts:5; t = [-fliplr(t(2:end)),t]; t2 = t.*t;
g = (2/sqrt(3))*pi^(-1/4)*(1-t2).*exp(-t2/2); g = g';
n = 0:length(g)-1; tmax = n(end)*Ts;
G = toeplitz([g;zeros(M,1)], [g(1),zeros(1,M)]);
[L,N] = size(G); n0 = floor(L/2);
y = [zeros(n0,1);1;zeros(n0,1)];
R = G'*G; r = G'*y;
c_ls = R\r;
e = y - G*c_ls; format long; Els = e'*e, format short;
Els =
0.09340142175215
y_ls = conv(g,c_ls); m = 0:length(y_ls)-1; mmax = m(end)*Ts;

Hf_2 = figure('units','inches','position',[1,1,6,6],...
    'paperunits','inches','paperposition',[0,0,6,6],...
    'numbertitle','off','name','P0808c');

subplot('position',[0.08,0.6,0.4,0.3]);
stem(n*Ts,g,'g'); hold on;
plot([0,tmax],[0,0],'w:',[0,0],[-.5,1],'w:'); axis([0,tmax,-0.5,1]);
title('sampled Wavelet','fontsize',12);
xlabel('\itn','fontsize',10); ylabel('{\itg}({\itn})','fontsize',10);
set(gca,'xtick',n*Ts,'xticklabel',...
    [' 0'; ' 1'; ' 2'; ' 3'; ' 4'; ' 5'; ' 6'; ' 7'; ' 8'; ' 9'; '10'; '11';...
    '12'; '13'; '14']);

subplot('position',[0.58,0.6,0.4,0.3]);
stem(n*Ts,c_ls,'g'); hold on;
plot([0,tmax],[0,0],'w:'); axis([0,tmax,-20,20]);
title('LS Inverse Filter','fontsize',12);
xlabel('\itn','fontsize',10); ylabel('{\itc}_ls({\itn})','fontsize',10);
set(gca,'xtick',n*Ts,'xticklabel',...
    [' 0'; ' 1'; ' 2'; ' 3'; ' 4'; ' 5'; ' 6'; ' 7'; ' 8'; ' 9'; '10'; '11'; '12';...
    '13'; '14']);

subplot('position',[0.08,0.1,0.9,0.3]);
stem(m*Ts,y_ls,'g'); hold on;
plot([0,mmax],[0,0],'w:'); axis([0,mmax,-1,1]);
title('Verification','fontsize',12);
xlabel('\itn','fontsize',10);
ylabel('{\itc}_ls({\itn})*{\itg}({\itn})','fontsize',10);
set(gca,'xtick',n*M*Ts,'xticklabel',...
    [' 0'; '14'; '29']);
```

The plot is shown in Figure 8.8c.

- (d) Repeat part (c) by increasing the sampling frequency by a factor of 2 and comparing with the results obtained in part (c).

```
%%(d) Optimum LS Inverse filter (Twice the sampling rate)
```

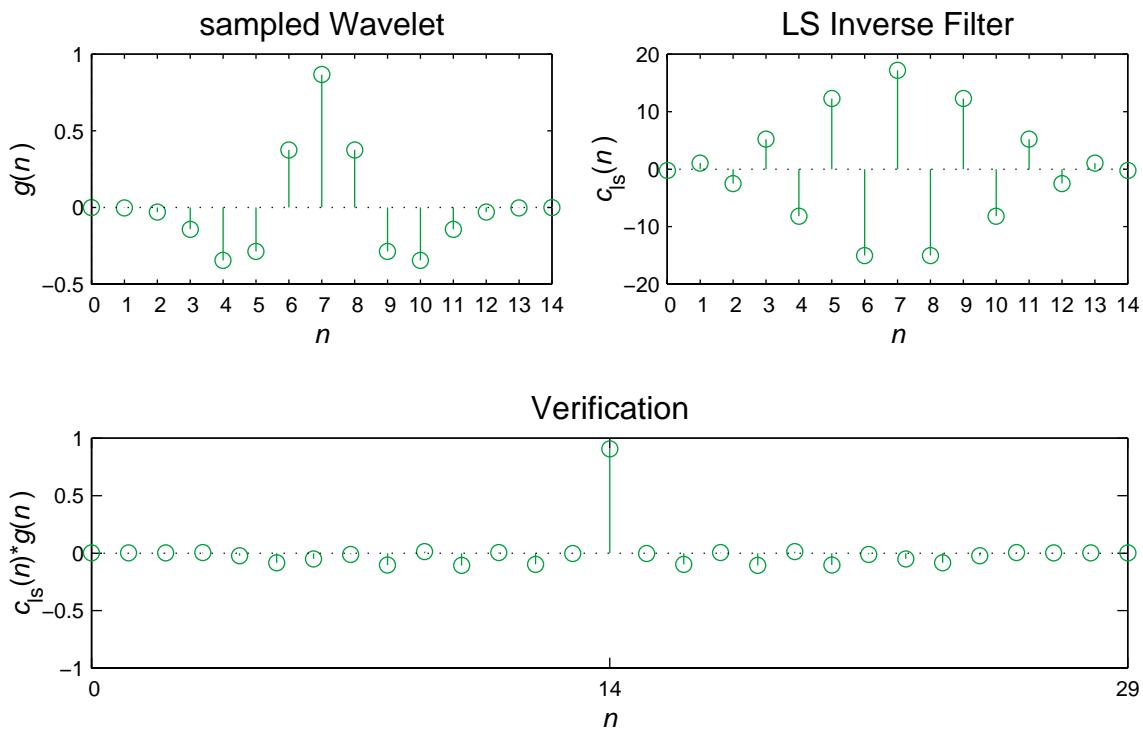


Figure 8.8c: Plots of sampled wavelet and its inverse filter

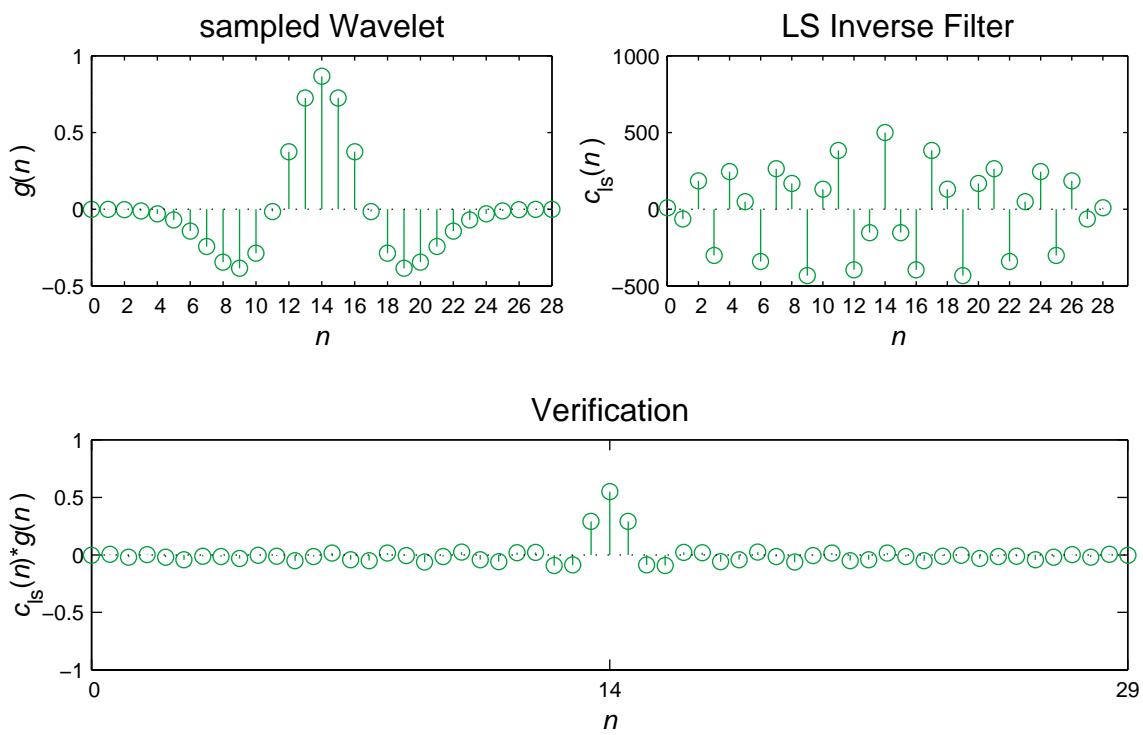


Figure 8.8d: Plots of sampled wavelet and its inverse filter for twice the sampling rate

```

Fs = 2*Fs; Ts = 1/Fs;

M = 28; % order of the filter
% Generate sample of g(t) as data
t = 0:Ts:5; t = [-fliplr(t(2:end)),t]; t2 = t.*t;
g = (2/sqrt(3))*pi^(-1/4)*(1-t2).*exp(-t2/2); g = g';
n = 0:length(g)-1; tmax = n(end)*Ts;
G = toeplitz([g;zeros(M,1)], [g(1),zeros(1,M)]);
[L,N] = size(G); n0 = floor(L/2);
y = [zeros(n0,1);1;zeros(n0,1)];
R = G'*G; r = G'*y;
c_ls = R\r; k = 0:length(c_ls)-1; kmax = k(end)*Ts;
e = y - G*c_ls; format long; Els = e'*e, format short;
Els =
0.45036127872114
y_ls = conv(g,c_ls); m = 0:length(y_ls)-1; mmax = m(end)*Ts;

Hf_3 = figure('units','inches','position',[1,1,6,6],...
    'paperunits','inches','paperposition',[0,0,6,6],...
    'numbertitle','off','name','P0808d');

subplot('position',[0.08,0.6,0.4,0.3]);
stem(n*Ts,g,'g'); hold on;
plot([0,tmax],[0,0],'w:',[0,0],[-.5,1],'w:'); axis([0,tmax,-0.5,1]);
title('sampled Wavelet','fontsize',12);
xlabel('\itn','fontsize',10); ylabel('{\itg}({\itn})','fontsize',10);
set(gca,'xtick',n*2*Ts,'xticklabel',...
    ['0';'2';'4';'6';'8';'10';'12';'14';'16';'18';'20';'22';...
    '24';'26';'28']);

subplot('position',[0.58,0.6,0.4,0.3]);
stem(k*Ts,c_ls,'g'); hold on;
plot([0,tmax],[0,0],'w:'); %axis([0,tmax,-20,20]);
title('LS Inverse Filter','fontsize',12);
xlabel('\itn','fontsize',10); ylabel('{\itc}_ls({\itn})','fontsize',10);
set(gca,'xtick',n*2*Ts,'xticklabel',...
    ['0';'2';'4';'6';'8';'10';'12';'14';'16';'18';'20';'22';...
    '24';'26';'28']);

subplot('position',[0.08,0.1,0.9,0.3]);
stem(m*Ts,y_ls,'g'); hold on;
plot([0,mmax],[0,0],'w:'); axis([0,mmax,-1,1]);
title('Verification','fontsize',12);
xlabel('\itn','fontsize',10);
ylabel('{\itc}_ls({\itn})*{\itg}({\itn})','fontsize',10);
set(gca,'xtick',n*M*Ts,'xticklabel',...
    ['0';'14';'29']);

```

The plot is shown in Figure 8.8d.

- 8.9** (a) Prove Equation (8.5.4) regarding the \mathbf{LDL}^H decomposition of the augmented matrix $\bar{\mathbf{R}}$.

Using direct substitution

$$\begin{bmatrix} \mathbf{L} & \mathbf{0} \\ \mathbf{k}^H & 1 \end{bmatrix} \begin{bmatrix} \mathbf{D} & \mathbf{0} \\ \mathbf{0} & E_{ls} \end{bmatrix} \begin{bmatrix} \mathbf{L}^H & \mathbf{k} \\ \mathbf{0}^H & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{LDL}^H & \mathbf{LDk} \\ \mathbf{k}^H \mathbf{DL}^H & \mathbf{k}^H \mathbf{Dk} + E_{ls} \end{bmatrix}$$

$$= \begin{bmatrix} \hat{\mathbf{R}} & \hat{\mathbf{d}} \\ \hat{\mathbf{d}}^H & E_y \end{bmatrix} = \bar{\mathbf{R}}$$

- (b) Solve the LS estimation problem in Example 8.5.1, using the \mathbf{LDL}^H decomposition of $\bar{\mathbf{R}}$ and the partitionings in (8.5.4).

```
% part b
X=[1 1 1; 2 2 1; 3 1 3; 1 0 1];
y=[1 2 4 3]';
R_hat=X'*X;
d_hat=X'*y;
Ey=y'*y;
R_bar=[R_hat d_hat; d_hat' Ey];
[L_bar D_bar] = ldlt(R_bar);
k=L_bar(end,1:end-1)';
L=L_bar(1:end-1,1:end-1);
Cls=L'\k
Cls =
    3.0000
   -1.5000
   -1.0000
```

- 8.10** Prove the order-recursive algorithm described by the relations given in (8.5.12). Demonstrate the validity of this approach, using the data in Example 8.5.1.

- 8.11** In this problem, we wish to show that the statistical interpretations of innovation and partial correlation for $w_m(n)$ and k_{m+1} in (8.5.12) hold in a deterministic LSE sense. To this end, suppose that the “partial correlation” between $\tilde{\mathbf{y}}$ and $\tilde{\mathbf{x}}_{m+1}$ is defined using the residual records $\tilde{\mathbf{e}}_m = \tilde{\mathbf{y}} - \mathbf{X}_m \mathbf{c}_m$ and $\tilde{\mathbf{e}}_m = \tilde{\mathbf{x}}_{m+1} + \mathbf{X}_m \mathbf{b}_m$, where \mathbf{b}_m is the LSE BLP. Show that $k_{k+1} = \beta_{m+1}/\xi_{m+1}$, where $\beta_{m+1} \triangleq \tilde{\mathbf{e}}_m^H \tilde{\mathbf{e}}_m$ and $\xi_{m+1} = \tilde{\mathbf{e}}_m^H \tilde{\mathbf{e}}_m$. Demonstrate the validity of these formulas using the data in Example 8.5.1.

- 8.12** Show that the Cholesky decomposition of a Hermitian positive definite matrix \mathbf{R} can be computed by using the following algorithm

```
for j = 1 to M
    lij = (rij - Σk=1j-1 |ljk|2)1/2
    for i = j + 1 to M
        lij = (rij - Σk=1j-1 lik* ljk) / ljj
    end i
end j
```

and write a MATLAB function for its implementation. Test your code using the built-in MATLAB function `chol`.

8.13 Compute the LDL^T and Cholesky decompositions of the following matrices:

$$\mathbf{X}_1 = \begin{bmatrix} 9 & 3 & -6 \\ 3 & 4 & 1 \\ -6 & 1 & 9 \end{bmatrix} \quad \text{and} \quad \mathbf{X}_2 = \begin{bmatrix} 6 & 4 & -2 \\ 4 & 5 & 3 \\ -2 & 3 & 6 \end{bmatrix}$$

```
% Part 1. Matrix X1
X1 = [9 3 -6; 3 4 1; -6 1 9];

% R1 = X1'*X1;
chX1 = chol(X1)
chX1 =
3.0000    1.0000   -2.0000
          0    1.7321    1.7321
          0        0    1.4142
[L1 D1] = ldlt(X1)
L1 =
1.0000        0        0
0.3333    1.0000        0
-0.6667    1.0000    1.0000
D1 =
9
3
2
A1 = L1*diag(sqrt(D1))
A1 =
3.0000        0        0
1.0000    1.7321        0
-2.0000    1.7321    1.4142

% Part 2. Matrix X2
X2 = [6 4 2; 4 5 3; 2 3 6];
%R2=X2'*X2;
[L2,D2] = ldlt(X2)
L2 =
1.0000        0        0
0.6667    1.0000        0
0.3333    0.7143    1.0000
D2 =
6.0000
2.3333
4.1429
chX2 = chol(X2)
chX2 =
2.4495    1.6330    0.8165
          0    1.5275    1.0911
          0        0    2.0354
A2 = L2*diag(sqrt(D2))
A2 =
```

2.4495	0	0
1.6330	1.5275	0
0.8165	1.0911	2.0354

8.14 Solve the LS problem in Example 8.6.1,

(a) using the QR decomposition of the augmented data matrix $\bar{\mathbf{X}} = [\mathbf{X} \mathbf{y}]$

```

x=[1 1 1; 2 2 1; 3 1 3; 1 0 1];
y=[1 2 4 3]';

x_hat=[x y];
[N M]=size(x_hat);

[q,r]=qr(x_hat)
q =
    -0.2582   -0.3545    0.8006    0.4082
    -0.5164   -0.7089   -0.4804    0.0000
    -0.7746    0.4557    0.1601   -0.4082
    -0.2582    0.4051   -0.3203    0.8165
r =
    -3.8730   -2.0656   -3.3566   -5.1640
        0    -1.3166    0.7089    1.2659
        0        0    0.4804   -0.4804
        0        0        0    1.2247
Rqr=r(1:3,1:3);

Zqr=q'*y
Zqr =
    -5.1640
    1.2659
   -0.4804
    1.2247
z1qr=Zqr(1:3);
z2qr=Zqr(4);
Cls_qr=Rqr\z1qr
Cls_qr =
    3.0000
   -1.5000
   -1.0000
Els_qr=z2qr'*z2qr
Els_qr =
    1.5000
eqr=q*[zeros(1,3) z2qr'];
eqr =
    0.5000
    0.0000
   -0.5000
    1.0000

```

(b) using the Cholesky decomposition of the matrix $\bar{\mathbf{R}} = \bar{\mathbf{X}}^H \bar{\mathbf{X}}$.

```
[R,p]=chol(x_hat'*x_hat);
```

```
Rchol=R(1:3,1:3);
```

```
Cls_ch=Rchol'\R(1:3,4)
```

```
Cls_ch =
```

```
    1.3333
```

```
   -3.0534
```

```
  -14.8224
```

```
Els_ch=R(4,4)
```

```
Els_ch =
```

```
    1.2247
```

We have $\mathbf{H} = \mathbf{I} - 2\mathbf{w}\mathbf{w}^H$ where $\|\mathbf{w}\| = \mathbf{w}^H \mathbf{w} = 1$.

- 8.15** (a) Show that a unit vector \mathbf{w} is an eigenvector of the matrix $\mathbf{H} = \mathbf{I} - 2\mathbf{w}\mathbf{w}^H$. What is the corresponding eigenvalue?

Consider

$$(\mathbf{I} - 2\mathbf{w}\mathbf{w}^H)\mathbf{w} = \mathbf{w} - 2\mathbf{w}\mathbf{w}^H\mathbf{w} = \mathbf{w} - 2\mathbf{w} = -\mathbf{w}$$

Hence \mathbf{w} is an eigenvector woth $\lambda = -1$.

- (b) If a vector \mathbf{z} is orthogonal to \mathbf{w} , show that \mathbf{z} is an eigenvector of \mathbf{H} . What is the corresponding eigenvalue?

Consider

$$(\mathbf{I} - 2\mathbf{w}\mathbf{w}^H)\mathbf{z} = \mathbf{z} - 2\mathbf{w}\underbrace{\mathbf{w}^H\mathbf{z}}_0 = \mathbf{z}$$

Hence \mathbf{z} is an eigenvector woth $\lambda = 1$.

- 8.16** Solve the LS problem

$$\mathbf{X} = \begin{bmatrix} 1 & 2 \\ 1 & 3 \\ 1 & 2 \\ 1 & -1 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} -3 \\ 10 \\ 3 \\ 6 \end{bmatrix}$$

using the Householder transformation.

```
X=[1 2; 1 3; 1 2; 1 -1]; Y=[-3 10 3 6]';
x=X(:,1);
s=norm(x);
sn=sqrt(2*s*(s+abs(x(1))));
w1=(x+[1;0;0;0]*sign(x(1))*s)/sn;
H1=eye(length(w1))-2*w1*w1';
X1=H1*X
X1 =
-2.0000 -3.0000
```

```

-0.0000    1.3333
-0.0000    0.3333
-0.0000   -2.6667
x=X1(2:end,2);
s=norm(x);
sn=sqrt(2*s*(s+abs(x(1)))); 
w22=(x+[1;0;0]*sign(x(1))*s)/sn;
w2=[0;w22];
H22=eye(length(w22))-2*w22*w22';
H2=eye(4,4);H2(2:end,2:end)=H22;
X2=H2*X1
X2 =
-2.0000   -3.0000
-0.0000   -3.0000
-0.0000      0
-0.0000      0
R=X2
R =
-2.0000   -3.0000
-0.0000   -3.0000
-0.0000      0
-0.0000      0
Q=H1*H2
Q =
-0.5000   -0.1667   -0.4744   -0.7051
-0.5000   -0.5000   -0.2692    0.6538
-0.5000   -0.1667    0.8333   -0.1667
-0.5000    0.8333   -0.0897    0.2179
Q*R
ans =
1.0000   2.0000
1.0000   3.0000
1.0000   2.0000
1.0000  -1.0000
w1
w1 =
0.8660
0.2887
0.2887
0.2887
w2
w2 =
0
0.8498
0.0654
-0.5230
% Solve the LS problem
z=Q'*Y;

```

```

cls=R(1:2,1:2) '\z(1:2)
cls =
    4.0000
   -4.0000
Els=z(3:4)'*z(3:4)
Els =
    90.0000

```

8.17 Solve Problem 8.16 by using the Givens transformation.

```

% Given's Rotation
X = [1,2;1,3;1,2;1,-1], A = X;
X =
    1      2
    1      3
    1      2
    1     -1
y = [-3;10;3;6];

% Step-1: Zero out X(4,1) => rotate rows 3 and 4 of Col 1
G41 = eye(4);
[c,s] = rotate(X(3,1),X(4,1));
G41(:,3:4) = G41(:,3:4)*[c,s;-s,c];
X = G41*X,
X =
    1.0000    2.0000
    1.0000    3.0000
    1.4142    0.7071
    0     -2.1213

% Step-2: Zero out X(3,1) => rotate rows 2 and 3 of Col 1
G31 = eye(4);
[c,s] = rotate(X(2,1),X(3,1));
G31(:,2:3) = G31(:,2:3)*[c,s;-s,c];
X = G31*X,
X =
    1.0000    2.0000
    1.7321    2.3094
   -0.0000   -2.0412
    0     -2.1213

% Step-3: Zero out X(2,1) => rotate rows 1 and 2 of Col 1
G21 = eye(4);
[c,s] = rotate(X(1,1),X(2,1));
G21(:,1:2) = G21(:,1:2)*[c,s;-s,c];
X = G21*X,
X =
    2.0000    3.0000
   -0.0000   -0.5774

```

```

-0.0000   -2.0412
      0    -2.1213

% Step-4: Zero out X(4,2) => rotate rows 3 and 4 of Col 2
G42 = eye(4);
[c,s] = rotate(X(3,2),X(4,2));
G42(:,3:4) = G42(:,3:4)*[c,s;-s,c];
X = G42*X,
X =
    2.0000    3.0000
   -0.0000   -0.5774
   -0.0000   -2.9439
    0.0000        0

% Step-5: Zero out X(3,2) => rotate rows 2 and 3 of Col 2
G32 = eye(4);
[c,s] = rotate(X(2,2),X(3,2));
G32(:,2:3) = G32(:,2:3)*[c,s;-s,c];
X = G32*X,
X =
    2.0000    3.0000
   -0.0000   -3.0000
    0.0000        0
    0.0000        0

% The G, Q and R matrices
G = G32*G42*G21*G31*G41
G =
    0.5000    0.5000    0.5000    0.5000
   -0.1667   -0.5000   -0.1667    0.8333
    0.8498   -0.3922   -0.3269   -0.1307
        0     0.5883   -0.7845    0.1961
Q = G'
Q =
    0.5000   -0.1667    0.8498        0
    0.5000   -0.5000   -0.3922    0.5883
    0.5000   -0.1667   -0.3269   -0.7845
    0.5000    0.8333   -0.1307    0.1961
R = triu(X)
R =
    2.0000    3.0000
        0   -3.0000
        0        0
        0        0

% LS solution
R1 = R(1:2,:)
R1 =
    2.0000    3.0000

```

```

          0    -3.0000
r = G*y
r =
      8.0000
      0
-8.2369
  4.7068
r1 = r(1:2)
r1 =
      8.0000
      0
c = R1\r1
c =
      4.0000
      0
LSE = norm(r(3:4))^2
LSE =
      90

% check using Matlab's LS tools
c = A\y
c =
      4.0000
      0.0000

```

8.18 Compute the QR decomposition of the data matrix

$$\mathbf{X} = \begin{bmatrix} 4 & 2 & 1 \\ 2 & 0 & 1 \\ 2 & 0 & -1 \\ 1 & 2 & 1 \end{bmatrix}$$

using the GS and MGS methods, and compare the obtained results.

```

% QR decomposition using Gram-Schmidt (GS) and Modified GS (MGS) methods
X = [4,2,1;2,0,1;2,0,-1;1,2,1]
X =
      4      2      1
      2      0      1
      2      0     -1
      1      2      1
[N,M] = size(X);

%% GS method
R = zeros(M); Q = zeros(N,M);

% vector q1
R(1,1) = norm(X(:,1));
Q(:,1) = X(:,1)/R(1,1);

```

```

% vector q2
R(1,2) = Q(:,1)'*X(:,2);
p2 = X(:,2) - R(1,2)*Q(:,1); R(2,2) = norm(p2);
Q(:,2) = p2/R(2,2);

% vector q3
R(1,3) = Q(:,1)'*X(:,3); R(2,3) = Q(:,2)'*X(:,3);
p3 = X(:,3) - R(1,3)*Q(:,1) - R(2,3)*Q(:,2); R(3,3) = norm(p3);
Q(:,3) = p3/R(3,3);

% QR decomposition
Q, R
Q =
0.8000    0.2000         0
0.4000   -0.4000      0.7071
0.4000   -0.4000     -0.7071
0.2000    0.8000         0
R =
5.0000    2.0000      1.0000
0       2.0000      1.0000
0           0      1.4142

% Check
er_GS = norm(X-Q*R)
er_GS =
2.2204e-016

%% MGS method
R = zeros(M); Q = zeros(N,M); X1 = X;

% vector q1
R(1,1) = norm(X(:,1));
Q(:,1) = X(:,1)/R(1,1);
R(1,2) = Q(:,1)'*X(:,2); X(:,2) = X(:,2) - R(1,2)*Q(:,1);
R(1,3) = Q(:,1)'*X(:,3); X(:,3) = X(:,3) - R(1,3)*Q(:,1);

% vector q2
R(2,2) = norm(X(:,2));
Q(:,2) = X(:,2)/R(2,2);
R(2,3) = Q(:,2)'*X(:,3); X(:,3) = X(:,3) - R(2,3)*Q(:,2);

% vector q3
R(3,3) = norm(X(:,3));
Q(:,3) = X(:,3)/R(3,3);

% QR decomposition
Q, R
Q =

```

```

0.8000    0.2000   -0.0000
0.4000   -0.4000    0.7071
0.4000   -0.4000   -0.7071
0.2000    0.8000   -0.0000
R =
5.0000    2.0000   1.0000
0        2.0000   1.0000
0        0       1.4142

```

```

% Check
er_MGS = norm(X1-Q*R)
er_MGS =
1.5701e-016

```

8.19 Solve the following LS problem

$$\mathbf{X} = \begin{bmatrix} 1 & -2 & -1 \\ 2 & 0 & 1 \\ 2 & -4 & 2 \\ 4 & 0 & 0 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} -1 \\ 1 \\ 1 \\ -2 \end{bmatrix}$$

by computing the QR decomposition using the GS algorithm.

Consider

$$\mathbf{X} = \begin{bmatrix} 1 & -2 & -1 \\ 2 & 0 & 1 \\ 2 & -4 & 2 \\ 4 & 0 & 0 \end{bmatrix} \triangleq [\mathbf{x}_1 \ \mathbf{x}_2 \ \mathbf{x}_3]$$

Then we have the following steps:

- Compute \mathbf{p}_1, r_{11} , and \mathbf{q}_1 :

$$\mathbf{p}_1 = \mathbf{x}_1 = \begin{bmatrix} 1 \\ 2 \\ 2 \\ 4 \end{bmatrix}, \quad r_{11} = \|\mathbf{p}_1\| = 5, \quad \mathbf{q}_1 = \mathbf{p}_1 / \|\mathbf{p}_1\| = \begin{bmatrix} 1 \\ 2 \\ 2 \\ 4 \end{bmatrix} / \left\| \begin{bmatrix} 1 \\ 2 \\ 2 \\ 4 \end{bmatrix} \right\| = \begin{bmatrix} 0.2 \\ 0.4 \\ 0.4 \\ 0.8 \end{bmatrix}$$

Compute $\mathbf{p}_2, r_{12}, r_{22}$, and \mathbf{q}_2 :

$$\mathbf{p}_2 = \mathbf{x}_2 - (\mathbf{q}_1^T \mathbf{x}_2) \mathbf{q}_1 = \begin{bmatrix} -2 \\ 0 \\ -4 \\ 0 \end{bmatrix} - \left(\begin{bmatrix} .2 & .4 & .4 & .8 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 2 \\ 4 \end{bmatrix} \right) \begin{bmatrix} 0.2 \\ 0.4 \\ 0.4 \\ 0.8 \end{bmatrix} = \begin{bmatrix} -\frac{8}{5} \\ \frac{4}{5} \\ -\frac{16}{5} \\ \frac{8}{5} \end{bmatrix}$$

$$r_{12} = \mathbf{q}_1^T \mathbf{x}_2 = -2, \quad r_{22} = \|\mathbf{p}_2\| = 4, \quad \text{and } \mathbf{q}_2 = \mathbf{p}_2 / \|\mathbf{p}_2\| = \begin{bmatrix} -0.4 \\ 0.2 \\ -0.8 \\ 0.4 \end{bmatrix}$$

- Compute \mathbf{p}_3 , r_{13} , r_{23} , r_{33} , and \mathbf{q}_3 :

$$\mathbf{p}_3 = \mathbf{x}_3 - (\mathbf{q}_1^T \mathbf{x}_3) \mathbf{q}_1 - (\mathbf{q}_2^T \mathbf{x}_3) \mathbf{q}_2 = \begin{bmatrix} -1.6 \\ 0.8 \\ 0.8 \\ -0.4 \end{bmatrix}$$

$$r_{13} = \mathbf{q}_1^T \mathbf{x}_3 = 1, \quad r_{23} = \mathbf{q}_2^T \mathbf{x}_3 = -1, \quad r_{33} = \|\mathbf{p}_3\| = 2, \quad \text{and } \mathbf{q}_3 = \mathbf{p}_3 / \|\mathbf{p}_3\| = \begin{bmatrix} -0.8 \\ 0.4 \\ 0.4 \\ -0.2 \end{bmatrix}$$

Thus the thin QR decomposition is

$$\mathbf{Q}_1 = \begin{bmatrix} 0.2 & -0.4 & -0.8 \\ 0.4 & 0.2 & 0.4 \\ 0.4 & -0.8 & 0.4 \\ 0.8 & 0.4 & -0.2 \end{bmatrix}, \quad \mathcal{R} = \begin{bmatrix} 5 & -2 & 1 \\ 0 & 4 & -1 \\ 0 & 0 & 2 \end{bmatrix}$$

Thus

$$\mathbf{X}\mathbf{c}_{ls} = \mathbf{Q}_1 \mathcal{R} \mathbf{c}_{ls} = \mathbf{y} \Rightarrow \mathcal{R} \mathbf{c}_{ls} = \mathbf{Q}_1^T \mathbf{y} = \begin{bmatrix} 0.2 & 0.4 & 0.4 & 0.8 \\ -0.4 & 0.2 & -0.8 & 0.4 \\ -0.8 & 0.4 & 0.4 & -0.2 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \\ 1 \\ -2 \end{bmatrix} = \begin{bmatrix} -1.0 \\ -1.0 \\ 2.0 \end{bmatrix}$$

or

$$\begin{bmatrix} 5 & -2 & 1 \\ 0 & 4 & -1 \\ 0 & 0 & 2 \end{bmatrix} \mathbf{c}_{ls} = \begin{bmatrix} -1.0 \\ -1.0 \\ 2.0 \end{bmatrix}$$

Finally using backsubstitution, we obtain

$$\mathbf{c}_{ls} = \begin{bmatrix} -0.4 \\ 0 \\ 1.0 \end{bmatrix}$$

- 8.20** Show that the computational organization of the MGS algorithm shown in Table 8.4 can be used to compute the GS algorithm if we replace the step $r_{mi} = \mathbf{q}_m^H \mathbf{x}_i$ by $r_{mi} = \mathbf{q}_m^H \mathbf{q}_i$ where \mathbf{q}_i 's are initialized as $\mathbf{q}_i = \mathbf{x}_i$, $1 \leq i \leq M$.

This follows by observing that the r_{mi} calculations are unchanged while \mathbf{x}_i 's play the role of \mathbf{p}_i 's in the GS algorithm.

- 8.21** Compute the SVD of $\mathbf{X} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 0 \end{bmatrix}$ by computing the eigenvalues and eigenvectors of $\mathbf{X}^H \mathbf{X}$ and $\mathbf{X} \mathbf{X}^H$. Check with the results obtained using the `svd` function.

Consider first $\mathbf{X}^H \mathbf{X}$

$$\mathbf{X}^H \mathbf{X} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}$$

Its eigenvectors are $\lambda_1 = 4$, $\lambda_2 = 0$, and its eigenmatrix is

$$\mathbf{V} = \frac{1}{\sqrt{2}} \left[\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix} \right]$$

Similarly, consider \mathbf{XX}^H

$$\mathbf{XX}^H = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 0 \\ 2 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Its eigenvectors are $\lambda_1 = 4$, $\lambda_2 = 0$, $\lambda_3 = 0$, and its eigenmatrix is

$$\mathbf{U} = \frac{1}{\sqrt{2}} \left[\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right]$$

From Matlab, the function $[U,S,V] = \text{svd}([1,1;1,1;0,0])$ gives

```
[U,S,V] = svd([1,1;1,1;0,0])
U =
    0.7071   -0.7071      0
    0.7071    0.7071      0
    0          0       1.0000
S =
    2      0
    0      0
    0      0
V =
    0.7071   -0.7071
    0.7071    0.7071
```

which agrees with the above results.

8.22 Repeat Problem 8.21 for

$$(a) \mathbf{X} = \begin{bmatrix} 6 & 2 \\ -7 & 6 \end{bmatrix}$$

```
% part a
X=[6 2; -7 6];
XHXa=X'*X;
XXHa=X*X';
[Ua1,Sa1,Va1] = svd(XHXa)
Ua1 =
    0.8944   -0.4472
   -0.4472   -0.8944
Sa1 =
    100.0000      0
            0   25.0000
Va1 =
    0.8944   -0.4472
```

```

-0.4472   -0.8944
[Ua2,Sa2,Va2] = svd(XXHa)
Ua2 =
  0.4472   -0.8944
 -0.8944   -0.4472
Sa2 =
 100.0000      0
      0  25.0000
Va2 =
  0.4472   -0.8944
 -0.8944   -0.4472
(b) X = [0  1  1
          1  1  0].
% part b
X = [0 1 1; 1 1 0];
XHXb=X'*X;
XXHb=X*X';
[Ub1,Sb1,Vb1] = svd(XHXb)
Ub1 =
  0.4082   -0.7071    0.5774
  0.8165    0.0000   -0.5774
  0.4082    0.7071    0.5774
Sb1 =
  3.0000      0      0
      0  1.0000      0
      0      0      0
Vb1 =
  0.4082   -0.7071    0.5774
  0.8165    0.0000   -0.5774
  0.4082    0.7071    0.5774
[Ub2,Sb2,Vb2] = svd(XXHb)
Ub2 =
  0.7071   -0.7071
  0.7071    0.7071
Sb2 =
  3.0000      0
      0  1.0000
Vb2 =
  0.7071   -0.7071
  0.7071    0.7071

```

- 8.23** Write a MATLAB program to produce the plots in Figure 8.14, using the matrix $\mathbf{X} = \begin{bmatrix} 6 & 2 \\ -7 & 6 \end{bmatrix}$. Hint: Use a parametric description of the circle in polar coordinates.

```

close all;
set(0,'defaultaxesfontsize',8);
Hf_1 = figure('units','inches','position',[0.05,0.3,6,6],...

```

```

% Transformation matrix X
X = [6,2;-7,6];
[U,S,V] = svd(X); V(:,2) = -V(:,2); U(:,2) = -U(:,2);
disp(sprintf('\n The singular values are %2i and %2i',S(1,1),S(2,2)));

The singular values are 10 and 5

Pv = [1,0;0,1]; % test point after transformation by V'
P = V*Pv; % test point before transformation by V'

% Unit circle: C = r*exp(j*theta) = exp(j*theta); (since r = 1)
theta = 0:1:360; % angle in degrees
x = cos(theta*pi/180); y = sin(theta*pi/180);
C = [x;y];

subplot(2,2,1);
plot(x,y,'g',[ -4,4],[0,0],'w',[0,0],[-4,4],'w:',P(1,1),P(2,1),'yo',...
    P(1,2),P(2,2),'yd');
axis([-4,4,-4,4]); axis('equal');
set(gca,'xtick',[-3:2:3], 'ytick',[-3:2:3]);
title('Unit circle with test points');

% Transformation of Unit circle using X
Cx = X*C; Px = X*P;
subplot(2,2,2);
plot(Cx(1,:),Cx(2,:),'g',[ -11,11],[0,0],'w',[0,0],[-11,11],'w:',...
    Px(1,1),Px(2,1),'yo',Px(1,2),Px(2,2),'yd');
axis([-11,11,-11,11]); axis('equal');
title('Transformation using X');

%% Transformation using SVD components
% X = U*S*V';

% Transformation using V'
Cv = V'*C;
subplot(2,2,3);
plot(Cv(1,:),Cv(2,:),'g',[ -4,4],[0,0],'w',[0,0],[-4,4],'w:',...
    Pv(1,1),Pv(2,1),'yo',Pv(1,2),Pv(2,2),'yd');
axis([-4,4,-4,4]); axis('equal');
set(gca,'xtick',[-3:2:3], 'ytick',[-3:2:3]);
title('Transformation using V^H');

% Transformation using S
Cs = S*Cv; Ps = S*Pv;
subplot(2,2,4);
plot(Cs(1,:),Cs(2,:),'g',[ -11,11],[0,0],'w',[0,0],[-11,11],'w:',...
    Ps(1,1),Ps(2,1),'yo',Ps(1,2),Ps(2,2),'yd');
axis([-11,11,-11,11]); axis('equal');

```

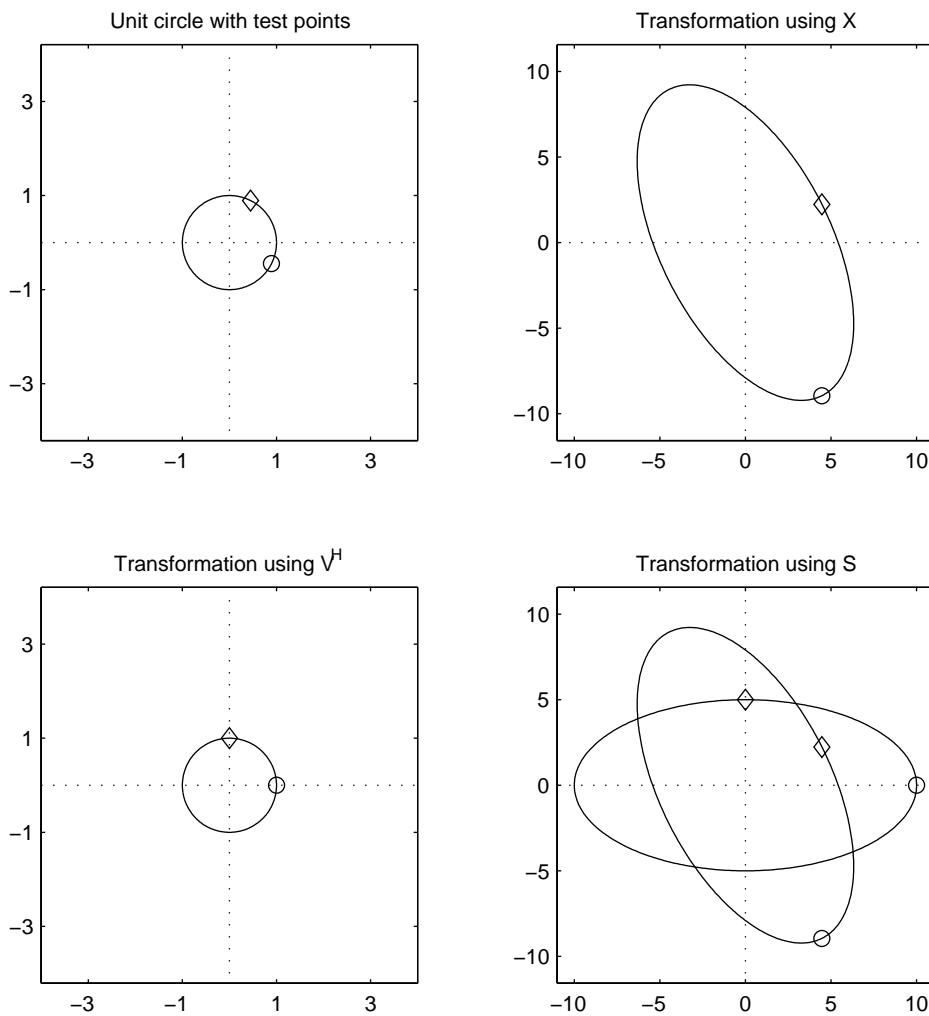


Figure 8.23: Geometric interpretation of SVD

```

title('Transformation using S');

% Transformation using U
Cu = U*Cs; Pu = U*Ps;
hold on;
plot(Cu(1,:),Cu(2,:),'r',Pu(1,1),Pu(2,1),'yo',Pu(1,2),Pu(2,2),'yd');

```

The plots are shown in Figure 8.23.

- 8.24** For the matrix $\mathbf{X} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}^T$ determine \mathbf{X}^+ and verify that \mathbf{X} and \mathbf{X}^+ satisfy the four Moore-Penrose conditions (8.7.22).

```

X=[0 1 1; 1 1 0]';
[U,S,V]=svd(X);
Splus=zeros(size(S'));
Splus(1:2,1:2)=diag(1./diag(S));
Xplus=V*Splus*U';
disp('A is')

```

```

A is
disp(Xplus)
-0.3333    0.3333    0.6667
  0.6667    0.3333   -0.3333

% Verify four Moore-Penrose Conditions

MP1=X*Xplus*X;
disp('XAX = X')
XAX = X
disp(MP1)
-0.0000    1.0000
  1.0000    1.0000
  1.0000    0.0000

MP2=Xplus*X*Xplus;
disp('AXA = A')
AXA = A
disp(MP2)
-0.3333    0.3333    0.6667
  0.6667    0.3333   -0.3333

MP3=(X*Xplus)';
disp('(XA)^(H) = XA')
(XA)^(H) = XA
disp(MP3)
  0.6667    0.3333   -0.3333
  0.3333    0.6667    0.3333
 -0.3333    0.3333    0.6667

MP4=(Xplus*X)';
disp('(AX)^(H) = AX')
(AX)^(H) = AX
disp(MP4)
  1.0000   -0.0000
  0.0000    1.0000

```

- 8.25** Prove the four Moore-Penrose conditions in (8.7.22) and explain why $\mathbf{X}\mathbf{X}^+$ and $\mathbf{X}^+\mathbf{X}$ are orthogonal projections onto the range space of \mathbf{X} and \mathbf{X}^H .

Using $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^H$, $\mathbf{X}^+ = \mathbf{V}\Sigma^+\mathbf{U}^H$, and $\mathbf{A} = \mathbf{X}^+$

- Consider

$$\begin{aligned}\mathbf{X}\mathbf{A}\mathbf{X} &= (\mathbf{U}\Sigma\mathbf{V}^H)(\mathbf{V}\Sigma^+\mathbf{U}^H)(\mathbf{U}\Sigma\mathbf{V}^H) = \mathbf{U}\Sigma\mathbf{V}^H\mathbf{V}\Sigma^+\mathbf{U}^H\mathbf{U}\Sigma\mathbf{V}^H \\ &= \mathbf{U}\Sigma\Sigma^+\Sigma\mathbf{V}^H = \mathbf{U}\Sigma\mathbf{V}^H = \mathbf{X}\end{aligned}$$

since

$$\Sigma^+\Sigma = \Sigma\Sigma^+ = \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \Sigma_r^{-1} & 0 \\ 0 & 0 \end{bmatrix} = \mathbf{I}$$

- Consider

$$\begin{aligned}\mathbf{AXA} &= (\mathbf{V}\Sigma^+\mathbf{U}^H)(\mathbf{U}\Sigma\mathbf{V}^H)(\mathbf{V}\Sigma^+\mathbf{U}^H) = \mathbf{V}\Sigma^+\mathbf{U}^H\mathbf{U}\Sigma\mathbf{V}^H\mathbf{V}\Sigma^+\mathbf{U}^H \\ &= \mathbf{V}\Sigma^+\Sigma\Sigma^+\mathbf{U}^H = \mathbf{X}^+ = \mathbf{A}\end{aligned}$$

- Consider

$$\begin{aligned}(\mathbf{XA})^H &= \mathbf{A}^H\mathbf{X}^H = (\mathbf{V}\Sigma^+\mathbf{U}^H)^H(\mathbf{U}\Sigma\mathbf{V}^H)^H = \mathbf{U}\Sigma^{+H}\mathbf{V}^H\mathbf{V}\Sigma^H\mathbf{U}^H \\ &= (\mathbf{U}\Sigma^{+H}\mathbf{V}^H)(\mathbf{V}\Sigma^H\mathbf{U}^H) = \mathbf{XA}\end{aligned}$$

- Similarly

$$\begin{aligned}(\mathbf{AX})^H &= \mathbf{X}^H\mathbf{A}^H = (\mathbf{U}\Sigma\mathbf{V}^H)^H(\mathbf{V}\Sigma^+\mathbf{U}^H)^H = \mathbf{V}\Sigma^H\mathbf{U}^H\mathbf{U}\Sigma^{+H}\mathbf{V}^H \\ &= (\mathbf{V}\Sigma^H\mathbf{U}^H)(\mathbf{U}\Sigma^{+H}\mathbf{V}^H) = \mathbf{AX}\end{aligned}$$

Consider

$$\begin{aligned}\mathbf{XX}^+ &= \mathbf{U}\Sigma\mathbf{V}^H\mathbf{V}\Sigma^+\mathbf{U}^H = \mathbf{U}\Sigma\Sigma^+\mathbf{U}^H \\ &= \mathbf{U}\begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} \Sigma_r^{-1} & 0 \\ 0 & 0 \end{bmatrix}\mathbf{U}^H = \mathbf{U}\begin{bmatrix} \mathbf{I}_r & 0 \\ 0 & 0 \end{bmatrix}\mathbf{U}^H \\ &= \mathbf{U}_r\mathbf{U}_r^H\end{aligned}$$

which by definition are orthogonal projections onto the range space of \mathbf{X} (Golub & Van Loan). Similarly $\mathbf{X}^+\mathbf{X}$ are orthogonal projections onto the range space of \mathbf{X}^+ .

8.26 In this problem we examine in greater detail the radio-frequency interference cancellation experiment discussed in Section 8.4.3. We first explain the generation of the various signals and then proceed with the design and evaluation of the LS interference canceler.

- (a) The useful signal is a pointlike target defined by

$$s(t) = \frac{d}{dt} \left(\frac{1}{e^{-\alpha t/t_r} + e^{\alpha t/t_f}} \right) \triangleq \frac{dg(t)}{dt}$$

where $\alpha = 2.3$, $t_r = 0.4$, and $t_f = 2$. Given that $F_s = 2$ GHz, determine $s(n)$ by computing the samples $g(n) = g(nT)$ in the interval $-2 \leq nT \leq 6$ ns and then computing the first difference $s(n) = g(n) - g(n-1)$. Plot the signal $s(n)$ and its Fourier transform (magnitude and phase), and check whether the pointlike and wideband assumptions are justified.

- (b) Generate $N = 4096$ samples of the narrowband interference using the formula

$$z(n) = \sum_{i=1}^L A_i \sin(\omega_i n + \phi_i)$$

and the following information:

```
Fs=2; % All frequencies are measured in GHz.
F=0.1*[0.6 1 1.8 2.1 3 4.8 5.2 5.7 6.1 6.4 6.7 7 7.8 9.3]';
L=length(F);
om=2*pi*F/Fs;
A=[0.5 1 1 0.5 0.1 0.3 0.5 1 1 1 0.5 0.3 1.5 0.5];
rand('seed',1954);
phi=2*pi*rand(L,1);
```

- (c) Compute and plot the periodogram of $z(n)$ to check the correctness of your code.
- (d) Generate N samples of white Gaussian noise $v(n) \sim \text{WGN}(0, 0.1)$ and create the observed signal $x(n) = 5s(n - n_0) + z(n) + v(n)$, where $n_0 = 1000$. Compute and plot the periodogram of $x(n)$.
- (e) Design a one-step ahead ($D = 1$) linear predictor with $M = 100$ coefficients using the FBLP method with no windowing. Then use the obtained FBLP to clean the corrupted signal $x(n)$ as shown in Figure 8.7. To evaluate the performance of the canceler, generate the plots shown in Figures 8.8 and 8.9.

8.27 Careful inspection of Figure 8.9 indicates that the D -step prediction error filter, that is, the system with input $x(n)$ and output $e^f(n)$, acts as a whitening filter. In this problem, we try to solve Problem 8.26 by designing a practical whitening filter using a power spectral density (PSD) estimate of the corrupted signal $x(n)$.

- (a) Estimate the PSD $\hat{R}_x^{(\text{PA})}(e^{j\omega_k})$, $\omega_k = 2\pi k/N_{\text{FFT}}$, of the signal $x(n)$, using the method of averaged periodograms. Use a segment length of $L = 256$ samples, 50 percent overlap, and $N_{\text{FFT}} = 512$.

```
% Common parameters
Fs = 2; % All frequencies are in GHz
N = 4096; n = 0:N-1; % Length of sequence

%% Pointlike Target Signal s(n)
alpha = 2.3; tr = 0.4; tf = 2;
n = [-2*Fs:6*Fs]; t = n/(Fs*1e9);
g = 1./exp(-alpha*t/tr)+exp(alpha*t/tf));
s = diff(g);

%% Given Narrowband interference parameters
% Frequencies of Interfering sinusoids
Fk = 0.1*[0.6,1,1.8,2.1,3,4.8,5.2,5.7,6.1,6.4,6.7,7,7.8,9.3]';
K = length(Fk);
omk = 2*pi*Fk/Fs;
% Amplitudes of Interfering sinusoids
Ak = [0.5,1,1,0.5,0.1,0.3,0.5,1,1,1,0.5,0.3,1.5,0.5]';
% Phases of Interfering sinusoids
rand('seed',1954); phik = 2*pi*rand(K,1);
% Narrowband Interference signal z(n)
z = sum((Ak*ones(1,N)).*sin(omk*n + phik*ones(1,N)));

%% Additive WGN
varv = 0.1; sigv = sqrt(varv);
v = sigv*randn(1,N);

%% Observed signal x(n)
x = 5*s+z+v;

% (a) PSD Rx using the method of averaged periodogram
Nfft = 512; L = 256; w_ham = hamming(L)';
Rx = psd(x,Nfft,Fs,w_ham,L/2,'none');
```

- (b) Since the PSD does not provide any phase information, we shall design a whitening FIR filter with linear

phase by

$$\tilde{H}(k) = \frac{1}{\sqrt{\hat{R}_x^{(\text{PA})}(e^{j\omega_k})}} e^{-j\frac{2\pi}{N_{\text{FFT}}} \frac{N_{\text{FFT}}-1}{2} k}$$

where $\tilde{H}(k)$ is the DFT of the impulse response of the filter, that is,

$$\tilde{H}(k) = \sum_{n=0}^{N_{\text{FFT}}-1} h(n) e^{-j\frac{2\pi}{N_{\text{FFT}}} nk}$$

with $0 \leq k \leq N_{\text{FFT}} - 1$.

% (b) Whitening FIR filter

```
magH = 1./sqrt(Rx'); magH = [magH,fliplr(magH(2:end-1))];  
k = 0:Nfft/2; phaH = 2*pi*(Nfft-1)/(Nfft*2)*k;  
phaH = [phaH,fliplr(phaH(2:end-1))];  
H = magH.*exp(-j*phaH);  
h = real(ifft(H));
```

- (c) Use the obtained whitening filter to clean the corrupted signal $x(n)$, and compare its performance with the FBLP canceler by generating plots similar to those shown in Figures 8.8 and 8.9.

% (c) Filtering using the Whitening filter
y = filter(h,1,x);

- (d) Repeat part (c) with $L = 128$, $N_{\text{FFT}} = 512$ and $L = 512$, $N_{\text{FFT}} = 1024$ and check whether spectral resolution has any effect upon the performance.

% (d1) L = 128, Nfft = 512
Nfft = 512; L = 128; w_ham = hamming(L);
Rx = psd(x,Nfft,Fs,w_ham,L/2,'none');
magH = 1./sqrt(Rx'); magH = [magH,fliplr(magH(2:end-1))];
k = 0:Nfft/2; phaH = 2*pi*(Nfft-1)/(Nfft*2)*k;
phaH = [phaH,fliplr(phaH(2:end-1))];
H = magH.*exp(-j*phaH);
h = real(ifft(H));
y = filter(h,1,x);

% (d2) L = 512, Nfft = 1024
Nfft = 1024; L = 512; w_ham = hamming(L);
Rx = psd(x,Nfft,Fs,w_ham,L/2,'none');
magH = 1./sqrt(Rx'); magH = [magH,fliplr(magH(2:end-1))];
k = 0:Nfft/2; phaH = 2*pi*(Nfft-1)/(Nfft*2)*k;
phaH = [phaH,fliplr(phaH(2:end-1))];
H = magH.*exp(-j*phaH);
h = real(ifft(H));
y = filter(h,1,x);

- 8.28** Repeat Problem 8.27, using the multitaper method of PSD estimation.

- (a) Estimate the PSD $\hat{R}_x^{(\text{PA})}(e^{j\omega_k})$, $\omega_k = 2\pi k/N_{\text{FFT}}$, of the signal $x(n)$, using the multitaper method. Use $N_{\text{FFT}} = 512$.

```

% Common parameters
Fs = 2; % All frequencies are in GHz
N = 4096; n = 0:N-1; % Length of sequence

%% Pointlike Target Signal s(n)
alpha = 2.3; tr = 0.4; tf = 2;
n = [-2*Fs:6*Fs]; t = n/(Fs*1e9);
g = 1./exp(-alpha*t/tr)+exp(alpha*t/tf));
s = diff(g);

%% Given Narrowband interference parameters
% Frequencies of Interfering sinusoids
Fk = 0.1*[0.6,1,1.8,2.1,3,4.8,5.2,5.7,6.1,6.4,6.7,7,7.8,9.3]';
K = length(Fk);
omk = 2*pi*Fk/Fs;
% Amplitudes of Interfering sinusoids
Ak = [0.5,1,1,0.5,0.1,0.3,0.5,1,1,1,0.5,0.3,1.5,0.5]';
% Phases of Interfering sinusoids
rand('seed',1954); phik = 2*pi*rand(K,1);
% Narrowband Interference signal z(n)
z = sum((Ak*ones(1,N)).*sin(omk*n + phik*ones(1,N)));

%% Additive WGN
varv = 0.1; sigv = sqrt(varv);
v = sigv*randn(1,N);

%% Observed signal x(n)
x = 5*s+z+v;

% (a) PSD Rx using the Multitaper method
Nfft = 512;
Rx = pmtm(x,4,Nfft,Fs);

```

- (b) Since the PSD does not provide any phase information, we shall design a whitening FIR filter with linear phase by

$$\tilde{H}(k) = \frac{1}{\sqrt{\hat{R}_x^{(\text{PA})}(e^{j\omega_k})}} e^{-j \frac{2\pi}{N_{\text{FFT}}} \frac{N_{\text{FFT}}-1}{2} k}$$

where $\tilde{H}(k)$ is the DFT of the impulse response of the filter, that is,

$$\tilde{H}(k) = \sum_{n=0}^{N_{\text{FFT}}-1} h(n) e^{-j \frac{2\pi}{N_{\text{FFT}}} nk}$$

with $0 \leq k \leq N_{\text{FFT}} - 1$.

```

% (b) Whitening FIR filter
magH = 1./sqrt(Rx'); magH = [magH,fliplr(magH(2:end-1))];
k = 0:Nfft/2; phaH = 2*pi*(Nfft-1)/(Nfft*2)*k;
phaH = [phaH,fliplr(phaH(2:end-1))];

```

```
H = magH.*exp(-j*phaH);
h = real(ifft(H));
```

- (c) Use the obtained whitening filter to clean the corrupted signal $x(n)$, and compare its performance with the FBLP canceler by generating plots similar to those shown in Figures 8.8 and 8.9.

```
% (c) Filtering using the Whitening filter
y = filter(h,1,x);
```

- (d) Repeat part (c) with $N_{FFT} = 512$ and $N_{FFT} = 1024$ and check whether spectral resolution has any effect upon the performance.

```
% (d1) Nfft = 512
Nfft = 512;
Rx = pmtm(x,4,Nfft,Fs);
magH = 1./sqrt(Rx'); magH = [magH,fliplr(magH(2:end-1))];
k = 0:Nfft/2; phaH = 2*pi*(Nfft-1)/(Nfft*2)*k;
phaH = [phaH,fliplr(phaH(2:end-1))];
H = magH.*exp(-j*phaH);
h = real(ifft(H));
y = filter(h,1,x);

% (d2) Nfft = 1024
Nfft = 1024;
Rx = psd(x,4,Nfft,Fs);
magH = 1./sqrt(Rx'); magH = [magH,fliplr(magH(2:end-1))];
k = 0:Nfft/2; phaH = 2*pi*(Nfft-1)/(Nfft*2)*k;
phaH = [phaH,fliplr(phaH(2:end-1))];
H = magH.*exp(-j*phaH);
h = real(ifft(H));
y = filter(h,1,x);
```

- 8.29** In this problem we develop an RFI canceler using a symmetric linear smoother with guard samples defined by

$$e(n) = x(n) - \hat{x}(n) \triangleq x(n) + \sum_{k=D}^M c_k[x(n-k) + x(n+k)]$$

where $1 \leq D < M$ prevents the use of the D adjacent samples to the estimation of $x(n)$.

- (a) Following the approach used in Section 8.4.3, demonstrate whether such a canceler can be used to mitigate RFI and under what conditions.

```
a=2.3; r=0.4;
f=2; Fs=2; T=1/Fs;
n=[-4:12];

for i=1:length(n)
    g(i)=1/(exp(-a*n(i)*T/r)+exp(a*n(i)*T/f));
end

s(1)=g(1);
for i=2:length(n)
```

```
s(i)=g(i)-g(i-1);
end
```

```
S=fft(s,501);
S=fftshift(S);
```

- (b) If there is theoretical justification for such a canceler, estimate its coefficients, using the method of LS with no windowing for $M = 50$ and $D = 1$ for the situation described in Problem 8.26.

```
% part(b)
F=0.1*[0.6 1 1.8 2.1 3 4.8 5.2 5.7 6.1 6.4 6.7 7 7.8 9.3]';
L=length(F);
om=2*pi*F/Fs;
A=[0.5 1 1 0.5 0.1 0.3 0.5 1 1 1 0.5 0.3 1.5 0.5]';
rand('seed',1954);
phi=2*pi*rand(L,1);

N=4096;
for i=1:N
    z(i)=A'*(sin(om.*i+phi));
end
z=z';
```

- (c) Use the obtained filter to clean the corrupted signal $x(n)$, and compare its performance with the FBLP canceler by generating plots similar to those shown in Figures 8.8 and 8.9.

```
% part(d)
sigmaV=0.1;
v=sqrt(sigmaV)*randn(size(z));
s_delay=zeros(size(z));
for i=1:length(s)
    s_delay(i+1000)=s(i);
end

x=5*s_delay+z+v;
```

```
M=50;
D=1;
R=lsmatvec('nowi',x,2*M+2);
```

```
Rb=fliplr(R);
```

```
R_bar=R+Rb;
```

```
c_bar=R_bar\R_bar(:,50);
```

- (d) Repeat part (c) for $D = 2$.

- 8.30** In Example 6.7.1 we studied the design and performance of an optimum FIR inverse system. In this problem, we design and analyze the performance of a similar FIR LS inverse filter, using training input-output data.

- (a) First, we generate $N = 100$ observations of the input signal $y(n)$ and the noisy output signal $x(n)$. We assume that $x(n) \sim \text{WGN}(0, 1)$ and $v(n) \sim \text{WGN}(0, 0.1)$. To avoid transient effects, we generate 200 samples and retain the last 100 samples to generate the required data records.

```
% (a)
M=10; sigmaX=1;
sigmaV=0.1; sigmaY=1;

y=sqrt(sigmaY)*randn(250,100);
v=sqrt(sigmaV)*randn(250,100);

s=filter([-3/5 7/5 -2/5], [1], [zeros(1,100); y]);

x=s(2:end,:)+v;
```

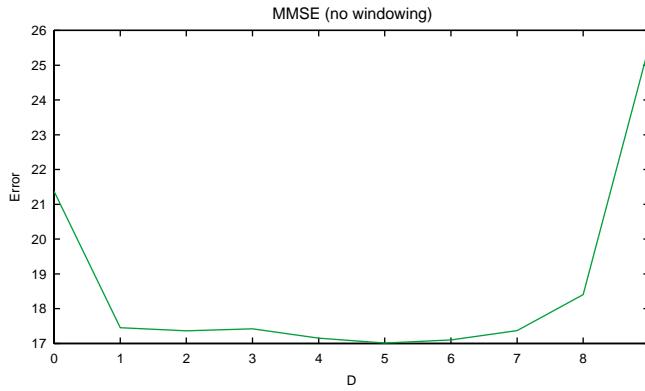
- (b) Design an LS inverse filter with $M = 10$ for $0 \leq D < 10$, using no windowing, and choose the best value of delay D .

```
% (b) No Windowing
X=x(101:200,:);

for j=1:100
    for i=1:10
        [R dd]=lsmatvec('nowi',X(:,j),10,y(99+i:198+i,j));
        d(:,i)=dd;
        h(:,i,j)=R\d(:,i);
        Py=y(99+i:198+i,j)'*y(99+i:198+i,j);
        P(i,j)=Py-d(:,i)'*h(:,i,j);
    end
end

Hf_1 = figure('units','SCRUN','position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',[0,0,4,2]);
plot([0:9],mean(P,2),'g'),
title('MMSE (no windowing)', 'fontsize', title_fontsize);
xlabel('D', 'fontsize', label_fontsize);
ylabel('Error', 'fontsize', label_fontsize);
exportfig(gcf,'p0830a.eps','fontmode','scaled',...
    'linemode','scaled','color','cmyk');

Hf_2 = figure('units','SCRUN','position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',[0,0,5,4]);
subplot(411);
stem([0:9],mean(h(:,2,:),3),'g');line([0:9],zeros(10,1));
axis([0 9 -.25 1]),
text(0.85, .8,['D=1'], 'units', 'normalized');
title('Filter Coefficients h_{D} (no windowing)', 'fontsize', title_fontsize);
subplot(412)
stem([0:9],mean(h(:,4,:),3),'g');line([0:9],zeros(10,1));
axis([0 9 -.25 1]),
text(0.85, .8,['D=3'], 'units', 'normalized', 'fontsize', title_fontsize);
```

**Figure 8.30a:** Plot of error variance vs D for no windowing

```

subplot(413)
stem([0:9],mean(h(:,6,:),3),'g');line([0:9],zeros(10,1));
axis([0 9 -.25 1]),
text(0.85, .8,['D=5'], 'units','normalized','fontsize',title_fontsize);
subplot(414)
stem([0:9],mean(h(:,8,:),3),'g');line([0:9],zeros(10,1));
axis([0 9 -.25 1]),
text(0.85, .8,['D=7'], 'units','normalized','fontsize',title_fontsize);
xlabel('n','fontsize',label_fontsize);

```

The plots of error variance and filter responses are shown in Figures 8.30a and 8.30b.

- (c) Repeat part (b) using full windowing.

```

% (c) full windowing
for j=1:100
    for i=1:10
        [R dd]=lsmatvec('full',X(:,j),10,y(99+i:198+i,j));
        d(:,i)=dd;
        h(:,i,j)=R\d(:,i);
        Py=y(99+i:198+i,j)'*y(99+i:198+i,j);
        P(i,j)=Py-d(:,i)'*h(:,i,j);
    end
end

Hf_3 = figure('units',SCRUN,'position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',[0,0,4,2]);
plot([0:9],mean(P,2),'g');
title('MMSE (full windowing)', 'fontsize',title_fontsize);
xlabel('D','fontsize',label_fontsize);
ylabel('Error','fontsize',label_fontsize);
exportfig(gcf,'p0830c.eps','fontmode','scaled',...
    'linemode','scaled','color','cmyk');

Hf_4 = figure('units',SCRUN,'position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',[0,0,5,4]);
subplot(411)

```

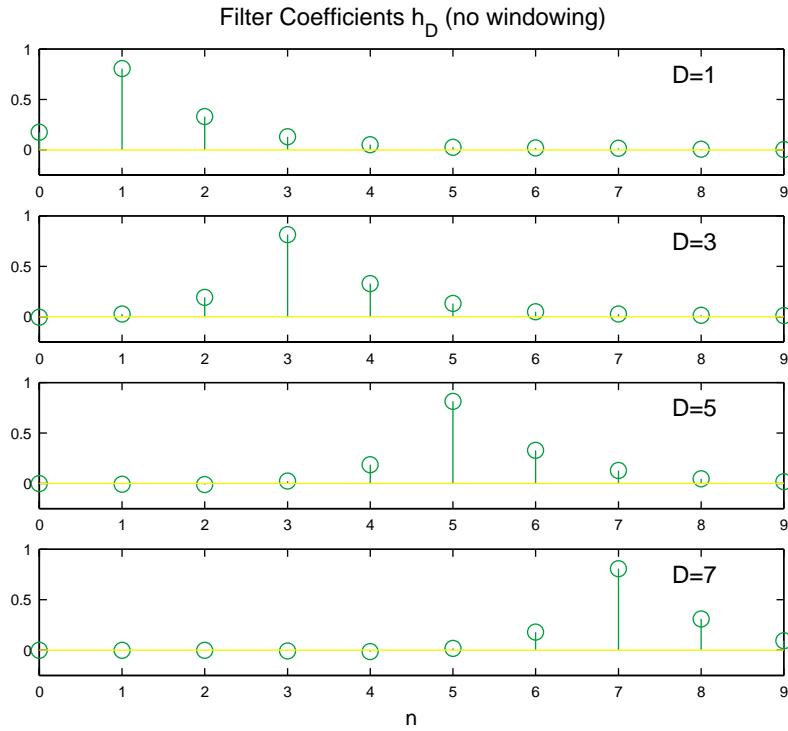


Figure 8.30b: Plots of filter responses for various values of D for no windowing

```

stem([0:9],mean(h(:,2,:)),3), 'g'); line([0:9],zeros(10,1));
axis([0 9 -.25 1]),
text(0.85, .8,['D=1'], 'units','normalized','fontsize',title_fontsize);
title('Filter Coefficients h_{D} (full windowing)', 'fontsize',title_fontsize);
subplot(412)
stem([0:9],mean(h(:,4,:)),3), 'g'); line([0:9],zeros(10,1));
axis([0 9 -.25 1]),
text(0.85, .8,['D=3'], 'units','normalized','fontsize',title_fontsize);
subplot(413)
stem([0:9],mean(h(:,6,:)),3), 'g'); line([0:9],zeros(10,1));
axis([0 9 -.25 1]),
text(0.85, .8,['D=5'], 'units','normalized','fontsize',title_fontsize);
subplot(414)
stem([0:9],mean(h(:,8,:)),3), 'g'); line([0:9],zeros(10,1));
axis([0 9 -.25 1]),
text(0.85, .8,['D=7'], 'units','normalized','fontsize',title_fontsize);
xlabel('n','fontsize',label_fontsize);

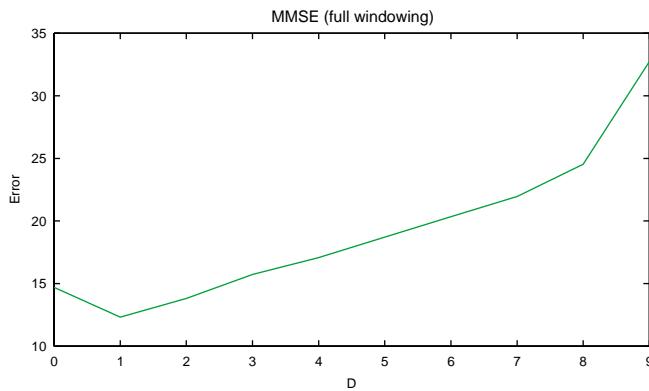
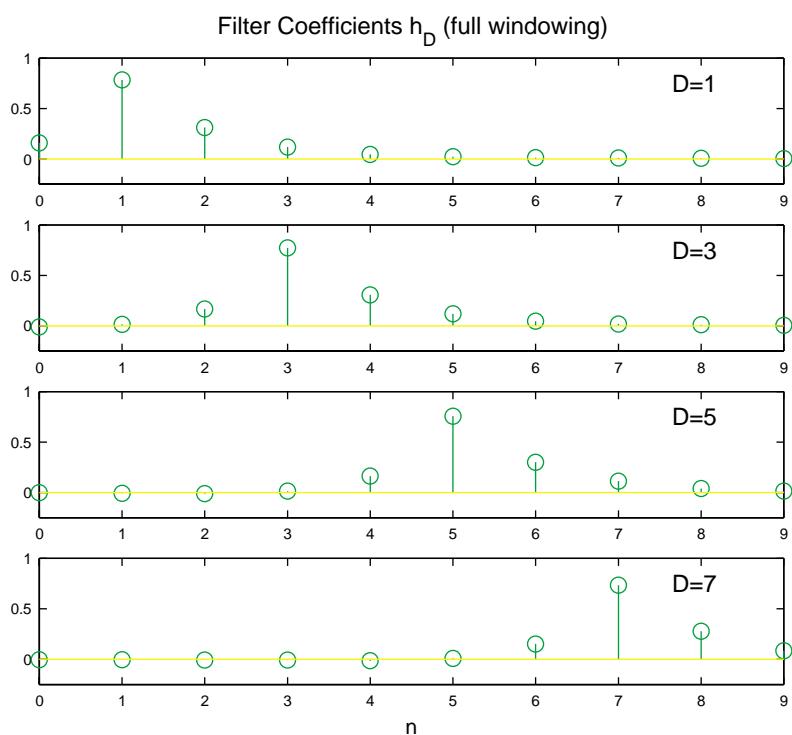
```

The plots of error variance and filter responses are shown in Figures 8.30c and 8.30d.

- (d) Compare the LS filters obtained in parts (b) and (c) with the optimum filter designed in Example 6.7.1. What are your conclusions?

8.31 In this problem we estimate the equalizer discussed in Example 6.8.1, using input-output training data, and we evaluate its performance using Monte Carlo simulation.

- (a) Generate $N = 1000$ samples of input-desired response data $\{x(n), a(n)\}_{0}^{N-1}$ and use them to estimate the

**Figure 8.30a:** Plot of error variance vs D for full windowing**Figure 8.30b:** Plots of filter responses for various values of D for full windowing

correlation matrix $\hat{\mathbf{R}}_x$ and the cross-correlation vector $\hat{\mathbf{d}}$ between $\mathbf{x}(n)$ and $y(n-D)$. Use $D = 7$, $M = 11$, and $W = 2.9$. Solve the normal equations to determine the LS FIR equalizer and the corresponding LSE.

```
% Given Parameters
N = 1000; % Length of sequences
M = 11; % equalizer length
D = 7; % delay in desired signal
varv = 0.001; % Variance of noise
K = 500; % Number of simulations

% Initializations
h = zeros(1,3);

%% W = 2.9: channel impulse response
W = 2.9;
h(1) = 0.5*(1+cos(2*pi*(1-2)/W));
h(2) = 0.5*(1+cos(2*pi*(2-2)/W));
h(3) = 0.5*(1+cos(2*pi*(3-2)/W));
h1 = [0 h(1) h(2) h(3) 0]';

% (a) LS FIR equalizer and LSE

% Generation of Bernoulli sequence a(n)
a = rand(N,1); a = 2*round(a)-1;

% Generation of noise sequence v(n)
v = sqrt(varv)*randn(N,1);

% Generation of received sequence x(n)
x = conv2(a,h1,'same') + v;

% Generation of desired response y(n)
y = [zeros((M-1)/2,1);a;zeros((M-1)/2,1)]; %y = y(1:N-M+1);

% Estimated autocorrelation matrix
Xbar = toeplitz([conj(x);zeros(M-1,1)], [conj(x(1)),zeros(1,M-1)]);
R_hat = Xbar'*Xbar;

% Estimated cross-correlation vector
d_hat = Xbar'*y;

% LS FIR equalizer
c = R_hat\d_hat;

% LSE
LSE = y'*y - c'*d_hat;

% Printout
disp(sprintf('\n *** The LS FIR equalizer is:\n %7.4f %7.4f %7.4f'))
```

```
%7.4f %7.4f %7.4f %7.4f %7.4f %7.4f %7.4f %7.4f ',c));  
*** The LS FIR equalizer is:  
-0.0012 0.0024 -0.0122 0.0604 -0.2563 1.1095 -0.2550  
0.0578 -0.0114 -0.0006 0.0013  
  
disp(sprintf(' *** The LSE is: %7.4f', LSE));  
*** The LSE is: 1.5309
```

- (b) Repeat part (a) 500 times; by changing the seed of the random number generators, compute the average (over the realizations) coefficient vector and average LSE, and compare with the optimum MSE equalizer obtained in Example 6.8.1. What are your conclusions?

```
% (b) Monte-Carlo Analysis  
c = zeros(M,K); LSE = zeros(1,K);  
for k = 1:K  
    rand('state',sum(100*clock)); randn('state',sum(100*clock));  
    a = rand(N,1); a = 2*round(a)-1;  
    v = sqrt(varv)*randn(N,1);  
    x = conv2(a,h1,'same') + v;  
    y = [zeros((M-1)/2,1);a;zeros((M-1)/2,1)]; %y = y(1:N-M+1);  
    Xbar = toeplitz([conj(x);zeros(M-1,1)], [conj(x(1)),zeros(1,M-1)]);  
    R_hat = Xbar'*Xbar;  
    d_hat = Xbar'*y;  
    c(:,k) = R_hat\d_hat;  
    LSE(k) = y'*y - c(:,k)'*d_hat;  
end  
c1 = mean(c,2);  
LSE1 = mean(LSE);
```

- (c) Repeat parts (a) and (b) by setting $W = 3.1$.

```
%% W = 3.5: channel impulse response  
W = 3.5;  
h(1) = 0.5*(1+cos(2*pi*(1-2)/W));  
h(2) = 0.5*(1+cos(2*pi*(2-2)/W));  
h(3) = 0.5*(1+cos(2*pi*(3-2)/W));  
h2 = [0 h(1) h(2) h(3) 0]';  
  
% (c) LS FIR equalizer and LSE  
  
% Generation of Bernoulli sequence a(n)  
a = rand(N,1); a = 2*round(a)-1;  
  
% Generation of noise sequence v(n)  
v = sqrt(varv)*randn(N,1);  
  
% Generation of received sequence x(n)  
x = conv2(a,h2,'same') + v;  
  
% Generation of desired response y(n)
```

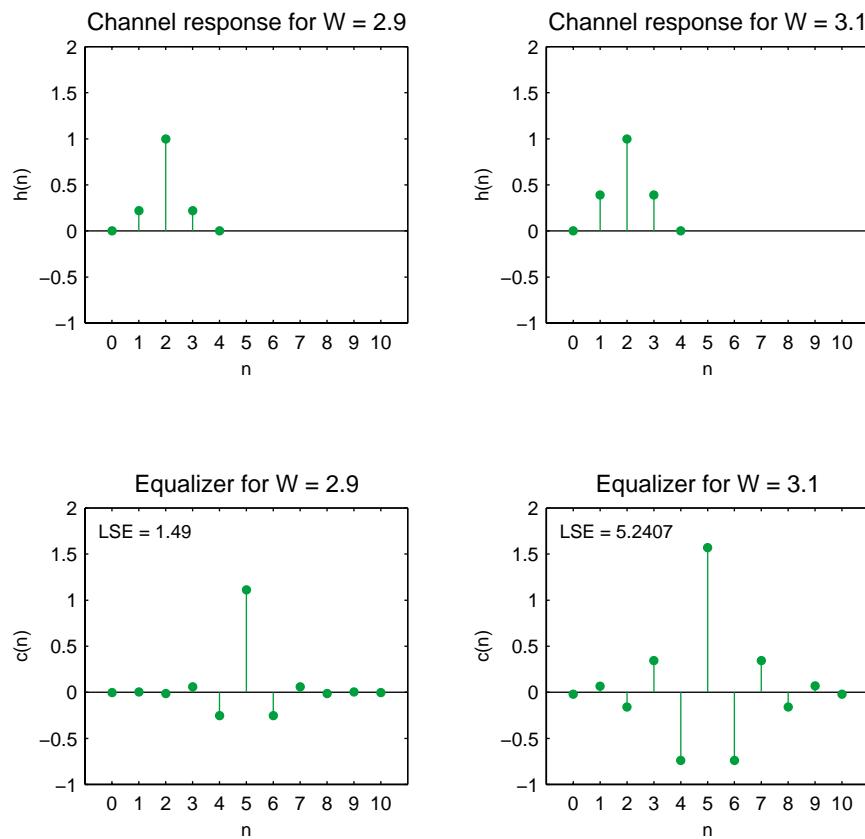


Figure 8.31: Plots of filter responses for various values of D for full windowing

```

y = [zeros((M-1)/2,1);a;zeros((M-1)/2,1)]; %y = y(1:N-M+1);

% Estimated autocorrelation matrix
Xbar = toeplitz([conj(x);zeros(M-1,1)],[conj(x(1)),zeros(1,M-1)]);
R_hat = Xbar'*Xbar;

% Estimated cross-correlation vector
d_hat = Xbar'*y;

% LS FIR equalizer
c = R_hat\d_hat;

% LSE
LSE = y'*y - c'*d_hat;

% Printout
disp(sprintf('\n *** The LS FIR equalizer is:\n %7.4f %7.4f %7.4f
%7.4f %7.4f %7.4f %7.4f %7.4f %7.4f %7.4f ',c));
*** The LS FIR equalizer is:
-0.0237  0.0729 -0.1662  0.3532 -0.7476  1.5752 -0.7450
 0.3513 -0.1629   0.0700 -0.0236

disp(sprintf(' *** The LSE is: %7.4f', LSE));
*** The LSE is:  5.3158

% (d) Monte-Carlo Analysis
c = zeros(M,K); LSE = zeros(1,K);
for k = 1:K
    rand('state',sum(100*clock)); randn('state',sum(100*clock));
    a = rand(N,1); a = 2*round(a)-1;
    v = sqrt(varv)*randn(N,1);
    x = conv2(a,h2,'same') + v;
    y = [zeros((M-1)/2,1);a;zeros((M-1)/2,1)]; %y = y(1:N-M+1);
    Xbar = toeplitz([conj(x);zeros(M-1,1)],[conj(x(1)),zeros(1,M-1)]);
    R_hat = Xbar'*Xbar;
    d_hat = Xbar'*y;
    c(:,k) = R_hat\d_hat;
    LSE(k) = y'*y - c(:,k)'*d_hat;
end
c2 = mean(c,2);
LSE2 = mean(LSE);

```

Plots of Channel responses and their corresponding equalizers are shown in Figure-8.31.

Signal Modeling and Parametric Spectral Estimation

9.1 The random process $x(n)$ is generated by

$$H(z) = \frac{1}{1 - 2.7607z^{-1} + 3.8108z^{-2} - 2.6535z^{-3} + 0.9238z^{-4}}$$

using a WGN(0, 1) process.

(a) The Matlab function armocov:

```
function [ahat,VarP] = armocov(x,P)
% Modofied covariance method to compute AR(P) model coefficients
% [ahat,e,Ep_fb] = armocov(x,P)
% Inputs:
%         x : data vector; P : Model order
% Outputs:
%         ahat : coefficient estimates
%         e : error vector
%         VarP : Error variance (forward_backword)
%
%-----
% Copyright 2000, by Dimitris G. Manolakis, Vinay K. Ingle,
% and Stephen M. Kogon. For use with the book
%
% "Statistical and Adaptive Signal Processing"
%
% McGraw-Hill Higher Education.
%-----

Rbar = lsmatrix(x,P+1);
if isreal(Rbar)
    Rfb = Rbar + flipud(fliplr(Rbar));
else
    Rfb = Rbar + flipud(fliplr(conj(Rbar)));
end

[ahat,VarP] = olsigest(Rfb,1);

Matlab Verification:

% Generate x(n)
bh = 1; ba = [1,-2.7607,3.8108,-2.6535,0.9238];
N = 250; w = randn(N,1);
x = filter(bh,ba,w);

% (a) Verification of Matlab function "armocov"
P = 4;
[ahat,VarP] = armocov(x,P);
```

```

disp(sprintf('ahat = [%6.4f,%6.4f,%6.4f,%6.4f]',ahat));
ahat = [-2.8153,3.9155,-2.7413,0.9480]
disp(sprintf('Error variance: %8.4f',VarP));
Error variance: 400.1909

```

- (b) The plots are generated using the following Matlab script and shown in Figure 9.1.

```

% (b) Plot of error variance and MSC vs P
Pmax = 15; V = zeros(Pmax,1);
FPE = V; AIC = V; MDL = V; CAT = V;

for P = 1:Pmax
    [ahat,VarP] = armodcov(x,P);
    V(P) = VarP;
    FPE(P) = (N+P)/(N-P)*VarP;
    AIC(P) = N*log10(VarP) + 2*P;
    MDL(P) = N*log10(VarP) + P*log10(N);
    CAT(P) = sum([N-1:-1:N-P]'./(N*V(1:P)))/N -(N-P)/(N*VarP);
end
P = 1:Pmax;
Hf_1 = figure('units','SCRUN','position',SCRPOS,...
stem(P,V/max(V),'g'); axis([0,Pmax+1,0,1.1]);
xlabel('Model order P','fontsize',label_fontsize);
ylabel('Normalized error variance','fontsize',label_fontsize);
title('Modified Covariance Method','fontsize',title_fontsize);
set(gca,'xtick',[1:1:Pmax],'ytick',[0:0.2:1]);

Hf_2 = figure('units','SCRUN','position',SCRPOS,...

subplot(2,2,1); plot(P,FPE/max(FPE),'g');
%xlabel('Model order P','fontsize',label_fontsize);
ylabel('Normalized FPE(P)','fontsize',label_fontsize);
title('Final Prediction Error','fontsize',title_fontsize);
set(gca,'xtick',[1:1:Pmax],'xlim',[0,Pmax+1]);

subplot(2,2,2); plot(P,AIC/max(AIC),'g');
%xlabel('Model order P','fontsize',label_fontsize);
ylabel('Normalized AIC(P)','fontsize',label_fontsize);
title('Akaike Information Criteria','fontsize',title_fontsize);
set(gca,'xtick',[1:1:Pmax],'xlim',[0,Pmax+1]);

subplot(2,2,3); plot(P,MDL/max(MDL),'g');
xlabel('Model order P','fontsize',label_fontsize);
ylabel('Normalized MDL(P)','fontsize',label_fontsize);
title('Minimum Description Length','fontsize',title_fontsize);
set(gca,'xtick',[1:1:Pmax],'xlim',[0,Pmax+1]);

subplot(2,2,4); plot(P,CAT/max(-CAT),'g');
xlabel('Model order P','fontsize',label_fontsize);
ylabel('Normalized CAT(P)','fontsize',label_fontsize);

```

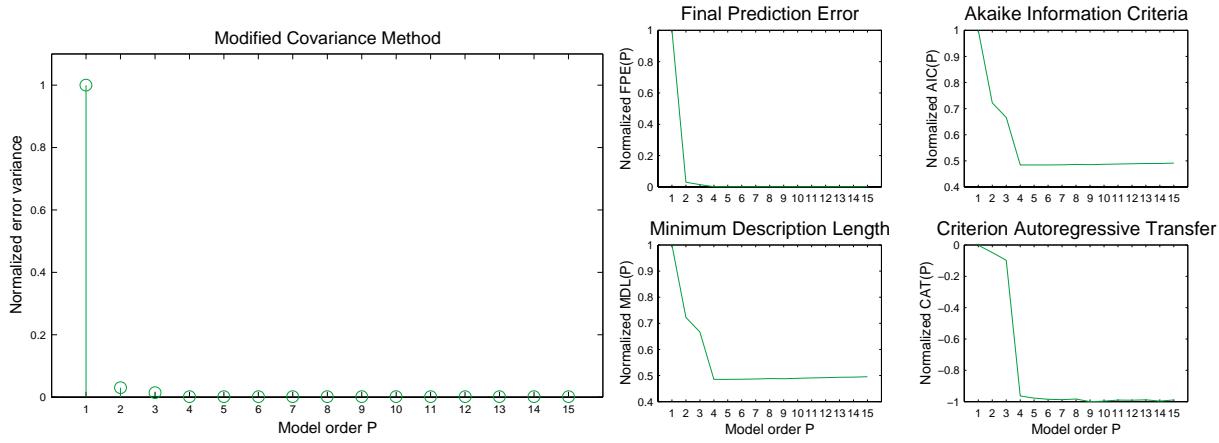


Figure 9.1: Plots of error variance and model selection criteria vs P

```
title('Criterion Autoregressive Transfer','fontsize',title_fontsize);
set(gca,'xtick',[1:Pmax], 'xlim',[0,Pmax+1]);
```

- (c) Comments: Both plots indicate that the correct model order is 4.

9.2 The forward/backward LS error \mathcal{E}_m^{fb} is given by (9.2.33)

$$\mathcal{E}_m^{fb} = \sum_{n=N_i}^{N_f} \{ |e_m^f(n)|^2 + |e_m^b(n)|^2 \}$$

- (a) Using (9.2.26), we obtain

$$\begin{aligned} |e_m^f(n)|^2 &= e_m^f(n)e_m^{f*}(n) \\ &= \{e_{m-1}^f(n) + k_{m-1}^*e_{m-1}^b(n-1)\} \{e_{m-1}^{f*}(n) + k_{m-1}e_{m-1}^{b*}(n-1)\} \\ &= |e_{m-1}^f(n)|^2 + |k_{m-1}|^2 |e_{m-1}^b(n-1)|^2 \\ &\quad + e_{m-1}^f(n)k_{m-1}e_{m-1}^{b*}(n-1) + k_{m-1}^*e_{m-1}^b(n-1)e_{m-1}^{f*}(n) \end{aligned}$$

Similarly, from (9.2.27) we obtain

$$\begin{aligned} |e_m^b(n)|^2 &= |e_{m-1}^b(n-1)|^2 + |k_{m-1}|^2 |e_{m-1}^f(n)|^2 \\ &\quad + e_{m-1}^b(n-1)k_{m-1}^*e_{m-1}^{f*}(n) + k_{m-1}e_{m-1}^{b*}(n-1)e_{m-1}^f(n) \end{aligned}$$

Adding the above two equations and summing over $N_i \leq n \leq N_f$

$$\begin{aligned} \mathcal{E}_m^{fb} &= (1 + |k_{m-1}|^2) \sum_{N_i}^{N_f} |e_{m-1}^f(n)|^2 + (1 + |k_{m-1}|^2) \sum_{N_i}^{N_f} |e_{m-1}^b(n-1)|^2 \\ &\quad + 2k_{m-1} \sum_{N_i}^{N_f} e_{m-1}^f(n)e_{m-1}^{b*}(n-1) + 2k_{m-1}^* \sum_{N_i}^{N_f} e_{m-1}^b(n-1)e_{m-1}^{f*}(n) \end{aligned}$$

Using $\mathcal{E}_{m-1}^f = \sum_{N_i}^{N_f} |e_{m-1}^f(n)|^2$, $\mathcal{E}_{m-1}^b = \sum_{N_i}^{N_f} |e_{m-1}^b(n-1)|^2$, and $\beta_{m-1}^{fb*} = \sum_{N_i}^{N_f} e_{m-1}^f(n)e_{m-1}^{b*}(n-1)$, we obtain

$$\mathcal{E}_m^{fb} = (1 + |k_{m-1}|^2) \{ \mathcal{E}_{m-1}^f + \mathcal{E}_{m-1}^b \} + 4 \operatorname{Re} \{ k_{m-1}^* \beta_{m-1}^{fb*} \}$$

(b) Using $\frac{\partial \mathcal{E}_{m-1}^{\text{fb}}}{\partial k_{m-1}^*} = 0$, we obtain

$$2k_{m-1} \{ \mathcal{E}_{m-1}^{\text{f}} + \mathcal{E}_{m-1}^{\text{b}} \} + 4\beta_{m-1}^{\text{fb}} = 0$$

which gives

$$k_{m-1}^{\text{B}} = -\frac{\beta_{m-1}^{\text{fb}}}{\frac{1}{2} \{ \mathcal{E}_{m-1}^{\text{f}} + \mathcal{E}_{m-1}^{\text{b}} \}} = \frac{2k_{m-1}^{\text{FP}} k_{m-1}^{\text{BP}}}{k_{m-1}^{\text{FP}} + k_{m-1}^{\text{BP}}}$$

where the last step is obtained by using (9.2.29) and (9.2.32) and the superscript B is used to emphasize Burg's contribution.

- (c) Since $|k_{m-1}^{\text{FP}}| \leq 1$ and $|k_{m-1}^{\text{BP}}| \leq 1$ and since k_{m-1}^{B} is the harmonic mean of k_{m-1}^{FP} and k_{m-1}^{BP} , it is clear that $|k_{m-1}^{\text{B}}| \leq 1$.
- (d) From (9.2.36), k_{m-1}^{IS} is the geometric mean of k_{m-1}^{FP} and k_{m-1}^{BP} , that is,

$$|k_{m-1}^{\text{IS}}|^2 = |k_{m-1}^{\text{FP}}| |k_{m-1}^{\text{BP}}| \leq 1 \text{ or } |k_{m-1}^{\text{IS}}| \leq 1$$

9.3 An AR(2) process is generated using the system function

$$H(z) = \frac{1}{1 - 0.9z^{-1} + 0.81z^{-2}}$$

excited by a WGN(0,1) process. The following Matlab script provides the numerical proof.

```
% Generate x(n)
bh = 1; ba = [1,-0.9,0.81];
N = 250; w = randn(N,1);
x = filter(bh,ba,w);

% Implement Table 9.1 (page 461) using full windowing method
P = 2;
kFP = zeros(1,P-1);
kBp = zeros(1,P-1);
kBurg = zeros(1,P-1);
kIS = zeros(1,P-1);

% 1. Input: x(n) for Ni <= n <= Nf
x = [0;x;0]; Lx = length(x); % Full windowing
ef = zeros(Lx,P-1);
eb = zeros(Lx,P-1);

% 2. Initialization
ef0 = x; eb0 = x;
betaFB0 = ef0(2:end)'*eb0(1:end-1);
Ef0 = ef0'*ef0;
Eb0 = eb0(1:end-1)'*eb0(1:end-1);
kFP0 = -betaFB0/Eb0;
kBp0 = -betaFB0/Ef0;
kBurg0 = 2*kFP0*kBp0/(kFP0+kBp0);
kIS0 = -betaFB0/sqrt(Eb0*Ef0);
```

```

for n = 2:length(x)
    ef(n,1) = ef0(n) + kBurg0*eb0(n-1);
    eb(n,1) = eb0(n-1) + kBurg0*ef0(n);
end

% 3. Loop: m = 2,...,P
for m = 2:P
    betaFB(m-1) = ef(2:end,m-1)'*eb(1:end-1,m-1);
    Ef(m-1) = ef(:,m-1)'*ef(:,m-1);
    Eb(m-1) = eb(1:end-1,m-1)'*eb(1:end-1,m-1);
    kFP(m-1) = -betaFB(m-1)/Eb(m-1);
    kBp(m-1) = -betaFB(m-1)/Ef(m-1);
    kBurg(m-1) = 2*kFP(m-1)*kBp(m-1)/(kFP(m-1)+kBp(m-1));
    kIS(m-1) = -betaFB(m-1)/sqrt(Eb(m-1)*Ef(m-1));
    for n = 2:length(x)
        ef(n,m) = ef(n,m-1) + kBurg(m-1)*eb(n-1,m-1);
        eb(n,m) = eb(n-1,m-1) + kBurg(m-1)*ef(n,m-1);
    end
end

% 4. Output:
kFP = [kFP0,kFP]
kFP = -0.4845 0.8483
kBp = [kBp0,kBp]
kBp = -0.4845 0.8407
kBurg = [kBurg0,kBurg]
kBurg = -0.4845 0.8445
kIS = [kISO,kIS]
kIS = -0.4845 0.8445

```

9.4 An AR(2) process is generated using the difference equation

$$x(n) = w(n) - 1.5857x(n-1) - 0.9604x(n-2)$$

where $w(n)$ WGN(0, 1).

(a) Matlab Script for $P = 1$:

```

% Generate x(n)
bh = 1; ba = [1,1.5857,0.9604];
N = 256; n = 0:N-1; w = randn(N,1);
x = filter(bh,ba,w);

% (a) Optimum 1st-order linear predictor
P = 1;
[a,e,V,FPE]=arls(x,P);

Hf_1 = figure('units','SCRUN','position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',[0,0,5,3]);
set(Hf_1,'NumberTitle','off','Name','Pr0904a');

```

```

L = 10; l = 0:L-1;
K = N/2; f = [0:K]/K; k = 1:K+1;
x = e;
subplot('position',[0.1,0.65,0.35,0.3]);
plot(n(1:100),x(1:100),'g'); axis([-1,100,-4,4]);
xlabel('\itn','fontsize',label_fontsize,'fontname','times');
ylabel('\{itx\}{\itn}','fontsize',label_fontsize,'fontname','times');
title('Error samples','Fontsize',title_fontsize);
set(gca,'xtick',[0:20:100],'ytick',[-4:1:4]);

% Autocorrelation test
r = autoc(x,L+1); rho = r(1:L)/r(1);
conf_lim = 1.96/sqrt(N);
subplot('position',[0.6,0.65,0.35,0.3]);
plot([-1,L],[0,0],'w'); hold on;
Hr = stem(l,rho,'filled','g');
set(Hr,'markersize',3); axis([-1,L,-1.2,1.2]);
plot([-1,L],[-conf_lim,-conf_lim],'r:');
plot([-1,L],[conf_lim,conf_lim],'r:');
hold off; conf_lim = round(conf_lim*100)/100;
xlabel('Lag \itl','fontsize',label_fontsize);
ylabel('\rho_{itx\{itl\}}','fontsize',label_fontsize,'fontname','times');
title('Autocorrelation test','Fontsize',title_fontsize);
set(gca,'xtick',[0:5:L],'ytick',[-1,-conf_lim,0,conf_lim,1]);

% PSD test
Rx = psd(x,N,2,boxcar(length(x)), 'none');
Ix = cumsum(Rx(1:K+1)); Ix = Ix/Ix(K+1);
Ibl = -1.36/sqrt(K-1) + (k-1)/(K-1);
Ibu = 1.36/sqrt(K-1) + (k-1)/(K-1);
subplot('position',[0.1,0.15,0.35,0.3]);
plot(f,Ix,'g',f,Ibl,'r:',f,Ibu,'r:'); axis([0,1,-0.2,1.2]);
xlabel('Frequency (cycles/samp)','fontsize',label_fontsize);
ylabel('I_{itx\{itf\}}','fontsize',label_fontsize,'fontname','times');
title('PSD test','Fontsize',title_fontsize);
Ibl(1) = round(Ibl(1)*100)/100;
Ibu(1) = round(Ibu(1)*100)/100;
set(gca,'xtick',[0:0.2:1],'ytick',[Ibl(1),0,Ibu(1),1]);
set(gca,'xticklabel',[ '0';'0.1';'0.2';'0.3';'0.4';'0.5']);

% PARCOR test
Vx=r(1); r=r/Vx;
[a,PACS,var]=durbin(r,L); PACS = [1;PACS(1:L-1)];
conf_lim = 1.96/sqrt(N);
subplot('position',[0.6,0.15,0.35,0.3]);
plot([-1,L],[0,0],'w'); hold on;
Hr = stem(l(2:L),PACS(2:L),'filled','g');
set(Hr,'markersize',3); axis([-1,L,-1.2,1.2]);

```

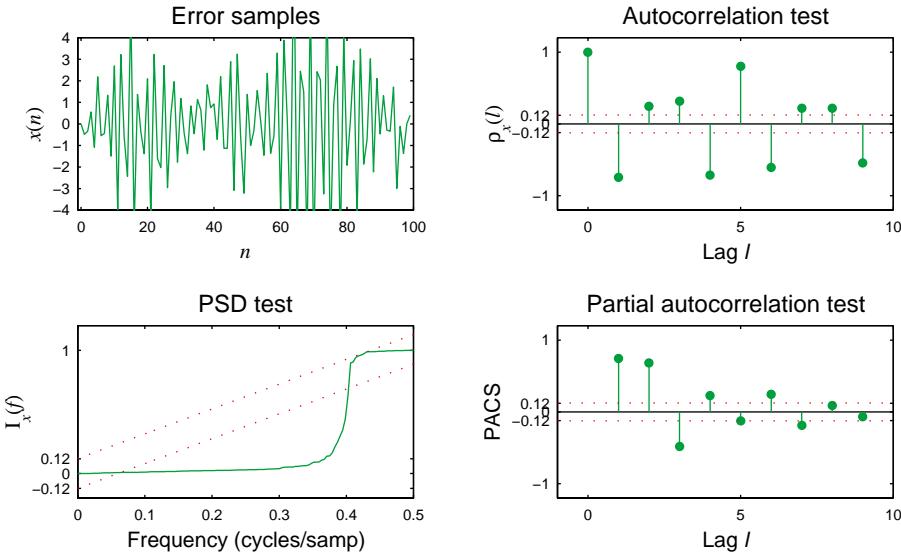


Figure 9.4a: Plots of error samples and its whiteness tests for $P = 1$

```

plot([-1,L],[-conf_lim,-conf_lim], 'r:');
plot([-1,L],[conf_lim,conf_lim], 'r:');
hold off; conf_lim = round(conf_lim*100)/100;
xlabel('Lag {\it l}', 'fontsize', label_fontsize);
ylabel('PACS', 'fontsize', label_fontsize);
title('Partial autocorrelation test', 'Fontsize', title_fontsize);
set(gca, 'xtick', [0:5:L], 'ytick', [-1,-conf_lim,0,conf_lim,1]);

```

The plots are shown in Figure 9.4a.

(b) Matlab Script for $P = 2$:

```

% (b1) Optimum 2nd-order linear predictor
P = 2;
[a,e,V,FPE]=arls(x,P);

Hf_2 = figure('units','SCRUN','position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',[0,0,5,3]);
set(Hf_2,'NumberTitle','off','Name','Pr0904b');

L = 10; l = 0:L-1;
K = N/2; f = [0:K]/K; k = 1:K+1;
x = e;
subplot('position',[0.1,0.65,0.35,0.3]);
plot(n(1:100),x(1:100),'g'); axis([-1,100,-4,4]);
xlabel('\it n', 'fontsize', label_fontsize, 'fontname', 'times');
ylabel('{\it x}(\{\it n\})', 'fontsize', label_fontsize, 'fontname', 'times');
title('Error samples', 'Fontsize', title_fontsize);
set(gca, 'xtick', [0:20:100], 'ytick', [-4:1:4]);

% Autocorrelation test
r = autoc(x,L+1); rho = r(1:L)/r(1);
conf_lim = 1.96/sqrt(N);

```

```

    subplot('position',[0.6,0.65,0.35,0.3]);
    plot([-1,L],[0,0],'w'); hold on;
    Hr = stem(l,rho,'filled','g');
    set(Hr,'markersize',3); axis([-1,L,-1.2,1.2]);
    plot([-1,L],[-conf_lim,-conf_lim],'r:');
    plot([-1,L],[conf_lim,conf_lim],'r:');
    hold off; conf_lim = round(conf_lim*100)/100;
    xlabel('Lag \itl','fontsize',label_fontsize);
    ylabel('rho_{\itx}(\{\itl\})','fontsize',label_fontsize,'fontname','times');
    title('Autocorrelation test','FontSize',title_fontsize);
    set(gca,'xtick',[0:5:L],'ytick',[-1,-conf_lim,0,conf_lim,1]);

    % PSD test
    Rx = psd(x,N,2,boxcar(length(x)), 'none');
    Ix = cumsum(Rx(1:K+1)); Ix = Ix/Ix(K+1);
    Ibl = -1.36/sqrt(K-1) + (k-1)/(K-1);
    Ibu = 1.36/sqrt(K-1) + (k-1)/(K-1);
    subplot('position',[0.1,0.15,0.35,0.3]);
    plot(f,Ix,'g',f,Ibl,'r:',f,Ibu,'r:'); axis([0,1,-0.2,1.2]);
    xlabel('Frequency (cycles/samp)','fontsize',label_fontsize);
    ylabel('I_{\itx}(\{\itf\})','fontsize',label_fontsize,'fontname','times');
    title('PSD test','FontSize',title_fontsize);
    Ibl(1) = round(Ibl(1)*100)/100;
    Ibu(1) = round(Ibu(1)*100)/100;
    set(gca,'xtick',[0:0.2:1],'ytick',[Ibl(1),0,Ibu(1),1]);
    set(gca,'xticklabel',[ '0';'0.1';'0.2';'0.3';'0.4';'0.5']);

    % PARCOR test
    Vx=r(1); r=r/Vx;
    [a,PACS,var]=durbin(r,L); PACS = [1;PACS(1:L-1)];
    conf_lim = 1.96/sqrt(N);
    subplot('position',[0.6,0.15,0.35,0.3]);
    plot([-1,L],[0,0],'w'); hold on;
    Hr = stem(l(2:L),PACS(2:L),'filled','g');
    set(Hr,'markersize',3); axis([-1,L,-1.2,1.2]);
    plot([-1,L],[-conf_lim,-conf_lim],'r:');
    plot([-1,L],[conf_lim,conf_lim],'r:');
    hold off; conf_lim = round(conf_lim*100)/100;
    xlabel('Lag {\itl}','fontsize',label_fontsize);
    ylabel('PACS','fontsize',label_fontsize);
    title('Partial autocorrelation test','FontSize',title_fontsize);
    set(gca,'xtick',[0:5:L],'ytick',[-1,-conf_lim,0,conf_lim,1]);

```

The plots are shown in Figure 9.4b.

Matlab Script for $P = 3$:

```

% (b2) Optimum 3rd-order linear predictor
P = 3;
[a,e,V,FPE]=arls(x,P);

```

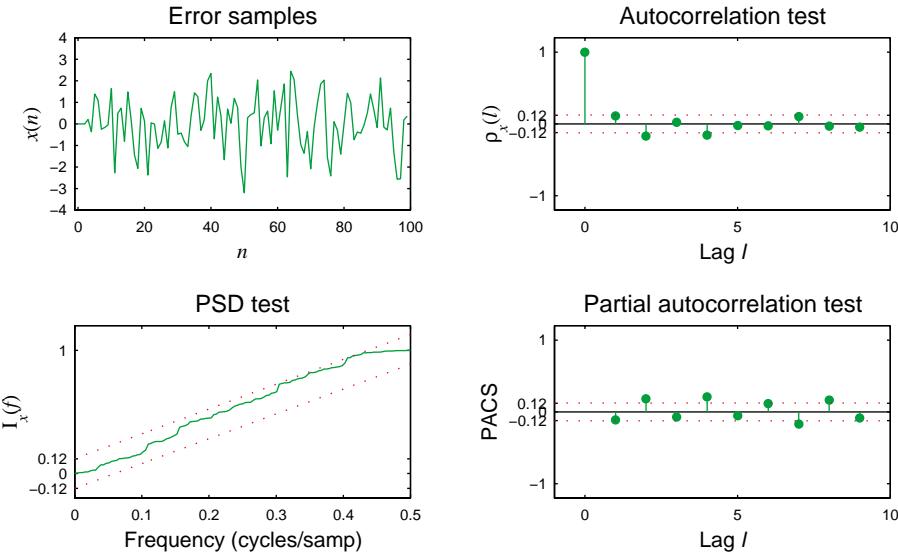


Figure 9.4b: Plots of error samples and its whiteness tests for $P = 2$

```

Hf_3 = figure('units','SCRUN','position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',[0,0,5,3]);
set(Hf_3,'NumberTitle','off','Name','Pr0904c');

L = 10; l = 0:L-1;
K = N/2; f = [0:K]/K; k = 1:K+1;
x = e;
subplot('position',[0.1,0.65,0.35,0.3]);
plot(n(1:100),x(1:100),'g'); axis([-1,100,-4,4]);
xlabel('\itn','fontsize',label_fontsize,'fontname','times');
ylabel('{\itx}{\itn}','fontsize',label_fontsize,'fontname','times');
title('Error samples','FontSize',title_fontsize);
set(gca,'xtick',[0:20:100],'ytick',[-4:1:4]);

% Autocorrelation test
r = autoc(x,L+1); rho = r(1:L)/r(1);
conf_lim = 1.96/sqrt(N);
subplot('position',[0.6,0.65,0.35,0.3]);
plot([-1,L],[0,0],'w'); hold on;
Hr = stem(l,rho,'filled','g');
set(Hr,'markersize',3); axis([-1,L,-1.2,1.2]);
plot([-1,L],[-conf_lim,-conf_lim],'r:');
plot([-1,L],[conf_lim,conf_lim],'r:');
hold off; conf_lim = round(conf_lim*100)/100;
xlabel('Lag \itl','FontSize',label_fontsize);
ylabel('\rho_{\itx}{\itl}','FontSize',label_fontsize,'fontname','times');
title('Autocorrelation test','FontSize',title_fontsize);
set(gca,'xtick',[0:5:L],'ytick',[-1,-conf_lim,0,conf_lim,1]);

% PSD test
Rx = psd(x,N,2,boxcar(length(x)), 'none');

```

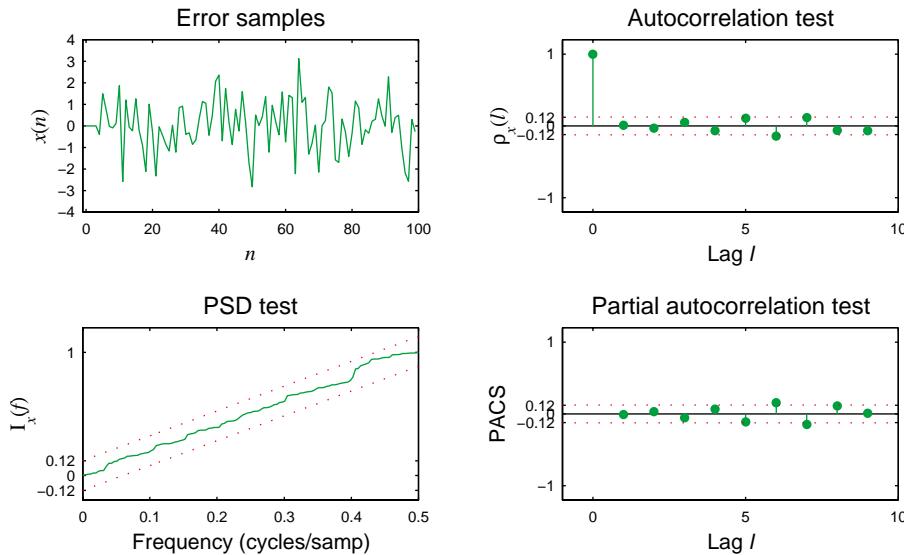


Figure 9.4c: Plots of error samples and its whiteness tests for $P = 2$

```

Ix = cumsum(Rx(1:K+1)); Ix = Ix/Ix(K+1);
Ibl = -1.36/sqrt(K-1) + (k-1)/(K-1);
Ibu = 1.36/sqrt(K-1) + (k-1)/(K-1);
subplot('position',[0.1,0.15,0.35,0.3]);
plot(f,Ix,'g',f,Ibl,'r:',f,Ibu,'r:'); axis([0,1,-0.2,1.2]);
xlabel('Frequency (cycles/samp)', 'fontsize', label_fontsize);
ylabel('I_{\it{x}}(\{\it{tf}\})', 'fontsize', label_fontsize, 'fontname', 'times');
title('PSD test', 'FontSize', title_fontsize);
Ibl(1) = round(Ibl(1)*100)/100;
Ibu(1) = round(Ibu(1)*100)/100;
set(gca, 'xtick', [0:0.2:1], 'ytick', [Ibl(1),0,Ibu(1),1]);
set(gca, 'xticklabel', [' 0 ' ; '0.1' ; '0.2' ; '0.3' ; '0.4' ; '0.5']);

% PARCOR test
Vx=r(1); r=r/Vx;
[a,PACS,var]=durbin(r,L); PACS = [1;PACS(1:L-1)];
conf_lim = 1.96/sqrt(N);
subplot('position',[0.6,0.15,0.35,0.3]);
plot([-1,L],[0,0], 'w'); hold on;
Hr = stem(l(2:L),PACS(2:L), 'filled', 'g');
set(Hr, 'markersize', 3); axis([-1,L,-1.2,1.2]);
plot([-1,L],[-conf_lim,-conf_lim], 'r:');
plot([-1,L],[conf_lim,conf_lim], 'r:');
hold off; conf_lim = round(conf_lim*100)/100;
xlabel('Lag {\it{l}}', 'fontsize', label_fontsize);
ylabel('PACS', 'fontsize', label_fontsize);
title('Partial autocorrelation test', 'FontSize', title_fontsize);
set(gca, 'xtick', [0:5:L], 'ytick', [-1,-conf_lim,0,conf_lim,1]);

```

The plots are shown in Figure 9.4c.

9.5 A MA(1) process is generated using the difference equation

$$x(n) = 0.5w(n) + 0.5w(n - 1)$$

where $w(n)$ WGN(0, 1).

(a) Matlab Script for the whiteness of $x(n)$:

```
% Generate x(n)
b = [0.5,0.5]; a = [1];
N = 256; n = 0:N-1; w = randn(N,1);
x = filter(b,a,w);

% (a) Whiteness of x(n)
Hf_1 = figure('units','SCRUN','position',SCRPOS,...
    'paperunits','PAPUN','paperposition',[0,0,5,3]);
set(Hf_1,'NumberTitle','off','Name','Pr0905a');

L = 10; l = 0:L-1;
K = N/2; f = [0:K]/K; k = 1:K+1;

subplot('position',[0.1,0.65,0.35,0.3]);
plot(n(1:100),x(1:100),'g'); axis([-1,100,-4,4]);
xlabel('\it{n}', 'fontsize',label_fontsize,'fontname','times');
ylabel('\it{x}(\it{n})', 'fontsize',label_fontsize,'fontname','times');
title('Signal x(n) Samples','Fontsize',title_fontsize);
set(gca,'xtick',[0:20:100], 'ytick',[-4:1:4]);

% Autocorrelation test
r = autoc(x,L+1); rho = r(1:L)/r(1);
conf_lim = 1.96/sqrt(N);
subplot('position',[0.6,0.65,0.35,0.3]);
plot([-1,L],[0,0],'w'); hold on;
Hr = stem(l,rho,'filled','g');
set(Hr,'markersize',3); axis([-1,L,-1.2,1.2]);
plot([-1,L],[-conf_lim,-conf_lim],'r:');
plot([-1,L],[conf_lim,conf_lim],'r:');
hold off; conf_lim = round(conf_lim*100)/100;
xlabel('Lag \it{l}', 'fontsize',label_fontsize);
ylabel('rho_{\it{l}}', 'fontsize',label_fontsize,'fontname','times');
title('Autocorrelation Test','Fontsize',title_fontsize);
set(gca,'xtick',[0:5:L], 'ytick',[-1,-conf_lim,0,conf_lim,1]);

% PSD test
Rx = psd(x,N,2,boxcar(length(x)), 'none');
Ix = cumsum(Rx(1:K+1)); Ix = Ix/Ix(K+1);
Ibl = -1.36/sqrt(K-1) + (k-1)/(K-1);
Ibu = 1.36/sqrt(K-1) + (k-1)/(K-1);
subplot('position',[0.1,0.15,0.35,0.3]);
plot(f,Ix,'g',f,Ibl,'r:',f,Ibu,'r:'); axis([0,1,-0.2,1.2]);
```

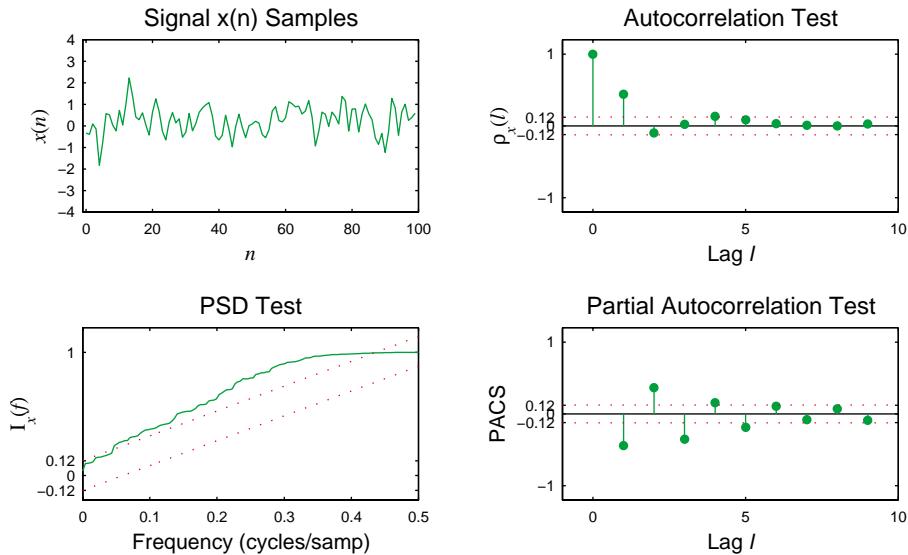


Figure 9.5a: Plots of signal samples and its whiteness tests for $x(n)$

```

xlabel('Frequency (cycles/samp)', 'fontsize', label_fontsize);
ylabel('I_{\it{x}}(\{ \it{f} \})', 'fontsize', label_fontsize, 'fontname', 'times');
title('PSD Test', 'FontSize', title_fontsize);
Ibl(1) = round(Ibl(1)*100)/100;
Ibu(1) = round(Ibu(1)*100)/100;
set(gca, 'xtick', [0:0.2:1], 'ytick', [Ibl(1), 0, Ibu(1), 1]);
set(gca, 'xticklabel', ['0'; '0.1'; '0.2'; '0.3'; '0.4'; '0.5']);

% PARCOR test
Vx=r(1); r=r/Vx;
[a,PACS,var]=durbin(r,L); PACS = [1;PACS(1:L-1)];
conf_lim = 1.96/sqrt(N);
subplot('position', [0.6, 0.15, 0.35, 0.3]);
plot([-1,L], [0,0], 'w'); hold on;
Hr = stem(l(2:L),PACS(2:L), 'filled', 'g');
set(Hr, 'markersize', 3); axis([-1,L,-1.2,1.2]);
plot([-1,L], [-conf_lim,-conf_lim], 'r:');
plot([-1,L], [conf_lim,conf_lim], 'r:');
hold off; conf_lim = round(conf_lim*100)/100;
xlabel('Lag {\it{l}}', 'fontsize', label_fontsize);
ylabel('PACS', 'fontsize', label_fontsize);
title('Partial Autocorrelation Test', 'FontSize', title_fontsize);
set(gca, 'xtick', [0:5:L], 'ytick', [-1,-conf_lim,0,conf_lim,1]);

```

The plots are shown in Figure 9.5a.

- (b) The signal $x(n)$ is processed through the AR(1) filter

$$H(z) = \frac{1}{1 + 0.95z^{-1}}$$

to generate $y(n)$. Matlab Script for the whiteness of $y(n)$:

```
% (b) Whiteness of y(n)
```

```

bH = 1; aH = [1,0.95];
y = filter(bH,aH,x);

Hf_2 = figure('units','SCRUN','position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',[0,0,5,3]);
set(Hf_2,'NumberTitle','off','Name','Pr0905b');

L = 10; l = 0:L-1;
K = N/2; f = [0:K]/K; k = 1:K+1;
x = y;
subplot('position',[0.1,0.65,0.35,0.3]);
plot(n(1:100),x(1:100),'g'); axis([-1,100,-4,4]);
xlabel('\itn','fontsize',label_fontsize,'fontname','times');
ylabel('{\itx}{\itn}','fontsize',label_fontsize,'fontname','times');
title('Signal y(n) samples','Fontsize',title_fontsize);
set(gca,'xtick',[0:20:100],'ytick',[-4:1:4]);

% Autocorrelation test
r = autoc(x,L+1); rho = r(1:L)/r(1);
conf_lim = 1.96/sqrt(N);
subplot('position',[0.6,0.65,0.35,0.3]);
plot([-1,L],[0,0],'w'); hold on;
Hr = stem(l,rho,'filled','g');
set(Hr,'markersize',3); axis([-1,L,-1.2,1.2]);
plot([-1,L],[-conf_lim,-conf_lim],'r:');
plot([-1,L],[conf_lim,conf_lim],'r:');
hold off; conf_lim = round(conf_lim*100)/100;
xlabel('Lag \itl','fontsize',label_fontsize);
ylabel('\rho_{\itx}{\itl}','fontsize',label_fontsize,'fontname','times');
title('Autocorrelation test','Fontsize',title_fontsize);
set(gca,'xtick',[0:5:L],'ytick',[-1,-conf_lim,0,conf_lim,1]);

% PSD test
Rx = psd(x,N,2,boxcar(length(x)), 'none');
Ix = cumsum(Rx(1:K+1)); Ix = Ix/Ix(K+1);
Ibl = -1.36/sqrt(K-1) + (k-1)/(K-1);
Ibu = 1.36/sqrt(K-1) + (k-1)/(K-1);
subplot('position',[0.1,0.15,0.35,0.3]);
plot(f,Ix,'g',f,Ibl,'r:',f,Ibu,'r:'); axis([0,1,-0.2,1.2]);
xlabel('Frequency (cycles/samp)','fontsize',label_fontsize);
ylabel('I_{\itx}{\itf}','fontsize',label_fontsize,'fontname','times');
title('PSD test','Fontsize',title_fontsize);
Ibl(1) = round(Ibl(1)*100)/100;
Ibu(1) = round(Ibu(1)*100)/100;
set(gca,'xtick',[0:0.2:1],'ytick',[Ibl(1),0,Ibu(1),1]);
set(gca,'xticklabel',[ ' 0 ' ; ' 0.1 ' ; ' 0.2 ' ; ' 0.3 ' ; ' 0.4 ' ; ' 0.5 ' ]);

% PARCOR test
Vx=r(1); r=r/Vx;

```

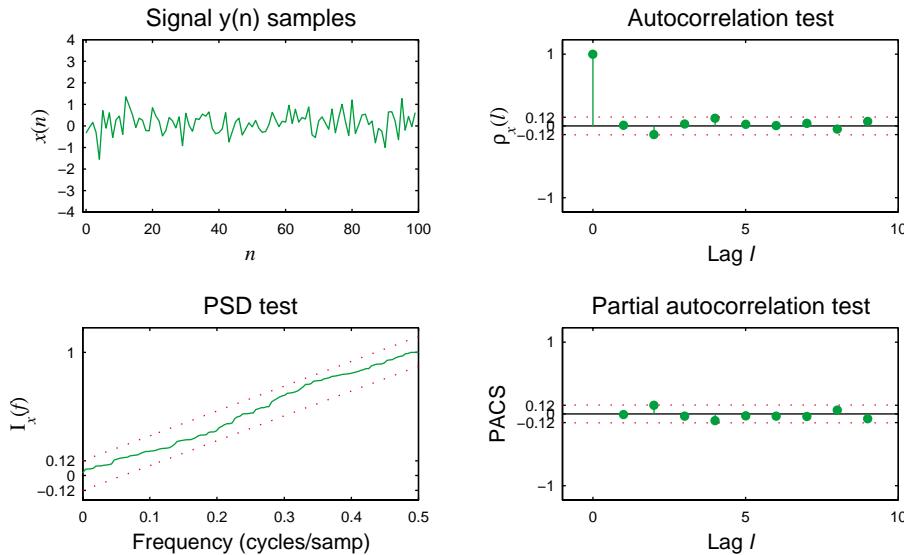


Figure 9.5b: Plots of signal samples and its whiteness tests for $y(n)$

```
[a,PACS,var]=durbin(r,L); PACS = [1;PACS(1:L-1)];
conf_lim = 1.96/sqrt(N);
subplot('position',[0.6,0.15,0.35,0.3]);
plot([-1,L],[0,0],'w'); hold on;
Hr = stem(l(2:L),PACS(2:L),'filled','g');
set(Hr,'markersize',3); axis([-1,L,-1.2,1.2]);
plot([-1,L],[-conf_lim,-conf_lim],'r:');
plot([-1,L],[conf_lim,conf_lim],'r:');
hold off; conf_lim = round(conf_lim*100)/100;
xlabel('Lag {\it l}', 'fontsize', label_fontsize);
ylabel('PACS', 'fontsize', label_fontsize);
title('Partial autocorrelation test', 'Fontsize', title_fontsize);
set(gca,'xtick',[0:5:L], 'ytick', [-1,-conf_lim,0,conf_lim,1]);
```

The plots are shown in Figure 9.5b.

9.6 The process $x(n)$ is given by

$$x(n) = Ae^{j(\omega_0 n + \theta)} + w(n)$$

where A is real positive constant, θ is uniformly distributed between $[0, 2\pi]$, ω_0 is a constant between $[0, \pi]$, and $w(n) \sim \text{WGN}(0, 1)$ that is uncorrelated with θ .

(a) The autocorrelation of $x(n)$ is given by

$$\begin{aligned} r_x(l) &= E[x(n)x^*(n-l)] \\ &= E[\{Ae^{j(\omega_0 n + \theta)} + w(n)\}\{Ae^{-j(\omega_0[n-l] + \theta)} + w^*(n-l)\}] \\ &= A^2 e^{j\omega_0 l} + r_w(l) = A^2 e^{j\omega_0 l} + \sigma_w^2 \delta(l) \end{aligned}$$

Thus the $(P + 1) \times (P + 1)$ autocorrelation matrix is given by

$$\begin{aligned} \mathbf{R}_x &= \begin{bmatrix} r_x(0) & r_x(1) & r_x(2) & \cdots & r_x(P) \\ r_x^*(1) & r_x(0) & r_x(1) & \cdots & r_x(P-1) \\ r_x^*(2) & r_x^*(1) & r_x(0) & \cdots & r_x(P-2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_x^*(P) & r_x^*(P-1) & r_x^*(P-2) & \cdots & r_x(0) \end{bmatrix} \\ &= A^2 \begin{bmatrix} 1 & e^{j\omega_0} & e^{j2\omega_0} & \cdots & e^{jP\omega_0} \\ e^{-j\omega_0} & 1 & e^{j\omega_0} & \cdots & e^{j(P-1)\omega_0} \\ e^{-j2\omega_0} & e^{-j\omega_0} & 1 & \cdots & e^{j(P-2)\omega_0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ e^{-jP\omega_0} & e^{-j(P-1)\omega_0} & e^{-j(P-2)\omega_0} & \cdots & 1 \end{bmatrix} + \sigma_w^2 \mathbf{I} \end{aligned}$$

The matrix in the last equation can be expressed as

$$\underbrace{\begin{bmatrix} 1 \\ e^{-j\omega_0} \\ e^{-j2\omega_0} \\ \vdots \\ e^{-jP\omega_0} \end{bmatrix}}_{\mathbf{e}} \underbrace{\begin{bmatrix} 1 & e^{j\omega_0} & e^{j2\omega_0} & \cdots & e^{jP\omega_0} \end{bmatrix}}_{\mathbf{e}^H}$$

Hence

$$\mathbf{R}_x = A^2 \mathbf{e} \mathbf{e}^H + \sigma_w^2 \mathbf{I}$$

- (b) To be completed
- (c) To be completed

9.7 An AR(2) process $y(n)$ is observed in noise $v(n)$ to obtain $x(n)$, that is,

$$x(n) = y(n) + v(n); \quad v(n) \sim \text{WGN}(0, \sigma_v^2)$$

where $v(n)$ is uncorrelated with $y(n)$ and

$$y(n) = 1.27y(n-1) - 0.81y(n-2) + w(n); \quad w(n) \sim \text{WGN}(0, 1)$$

- (a) Matlab script to compute and plot the true spectrum $R(e^{j\omega})$:

```
% (a) Plot of the true spectrum of x(n)
b = [1, -1.27, 0.81]; % AR(2) signal y(n) parameters
Rv = 10; sigv = sqrt(Rv); % Additive noise variance
omg = [0:500]*pi/500; % 501 frequency points over 0 to pi
H = freqz(b, a, omg); % AR(2) filter response
Ry = abs(H).*abs(H); % Spectrum of y(n)
Rx = Ry + Rv; % Spectrum of x(n)
Rx_db = 10*log10(Rx); % Spectrum in dB

Hf_1 = figure('units','SCRUN','position',SCRPOS,...%
'paperunits',PAPUN,'paperposition',[0,0,6,4]);
```

```

set(Hf_1,'NumberTitle','off','Name','Pr0907');

subplot(2,2,1);
plot(omg/pi,Rx_db,'g');
title('True Spectrum','fontsize',title_fontsize);
% xlabel('normalized frequency','fontsize',label_fontsize);
ylabel('R_x(e^{j\omega}) in dB','fontsize',label_fontsize);
axis([0,1,floor(min(Rx_db)/10)*10,ceil(max(Rx_db)/10)*10]);
text(0.6,15,'R_v = 10','fontsize',text_fontsize);

```

The plot is shown in Figure 9.7.

- (b) LS estimate of power spectrum using forward-backward linear predictor with $P = 2$ and $\sigma_v^2 = 1$:

```

% (b) Spectrum Estimation using LS (FB-predictor) Approach (Rv = 1)
N = 256; n = 0:N-1;
Rv = 1; sigv = sqrt(Rv);
P = 2;

% Loop over realizations
M = 10;
subplot(2,2,2);
for i = 1:M
    % Generate y(n) and x(n)
    w = randn(N,1);
    y = filter(b,a,w);
    v = sigv*randn(N,1);
    x = y + v;
    % Estimate model parameters
    [ahat,VarP] = armodcov(x,P);
    Hhat = freqz(b,[1;ahat]',omg);
    Rxhat = abs(Hhat).*abs(Hhat);
    Rxhat_db = 10*log10(Rxhat);
    plot(omg/pi,Rxhat_db,'g'); hold on;
end
title('Estimated Spectrum','fontsize',title_fontsize);
% xlabel('normalized frequency','fontsize',label_fontsize);
ylabel('R_x(e^{j\omega})','fontsize',label_fontsize);
axis([0,1,floor(min(Rxhat_db)/10)*10,ceil(max(Rxhat_db)/10)*10]);
text(0.6,2,['R_v = ',num2str(Rv)],'fontsize',text_fontsize);

```

The plot is shown in Figure 9.7.

- (c) LS estimate of power spectrum using forward-backward linear predictor with $P = 2$ and $\sigma_v^2 = 10$:

```

% (c) Spectrum Estimation using LS (FB-predictor) Approach (Rv = 10)
N = 256; n = 0:N-1;
Rv = 10; sigv = sqrt(Rv);
P = 2;

% Loop over realizations
M = 10;
subplot(2,2,3);

```

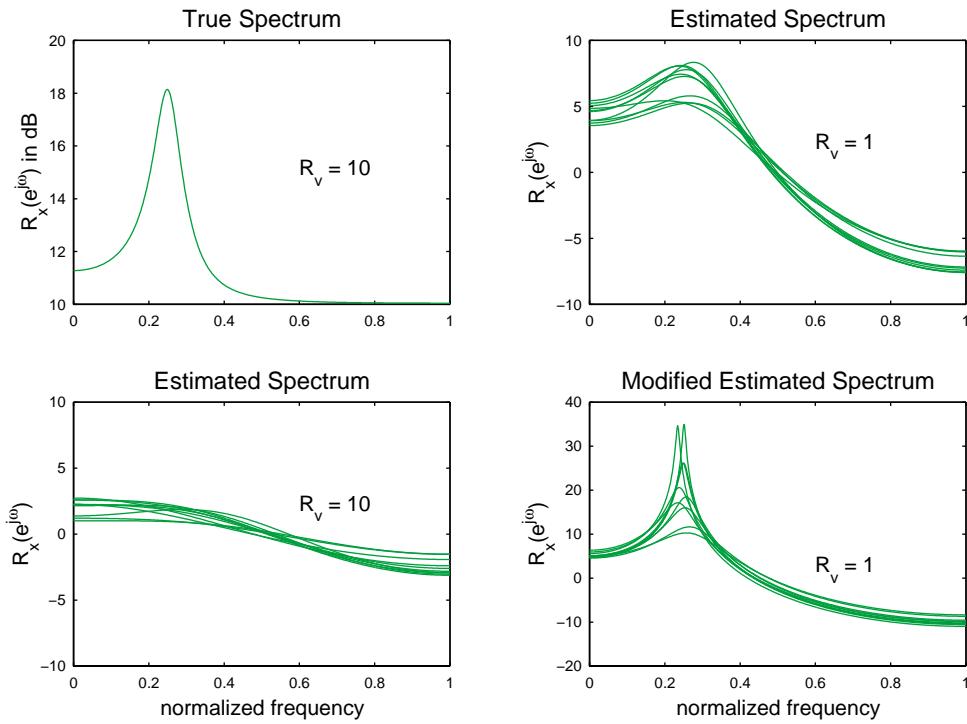


Figure 9.7: Plots of various spectra in P9.7

```
Rxhatdbmin = 0; Rxhatdbmax = 0;
for i = 1:M
    % Generate y(n) and x(n)
    w = randn(N,1);
    y = filter(b,a,w);
    v = sigv*randn(N,1);
    x = y + v;
    % Estimate model parameters
    [ahat,VarP] = armocov(x,P);
    Hhat = freqz(b,[1;ahat]',omg);
    Rxhat = abs(Hhat).*abs(Hhat);
    Rxhat_db = 10*log10(Rxhat);
    Rxhatdbmin = min([Rxhatdbmin,min(Rxhat_db)]);
    Rxhatdbmax = max([Rxhatdbmax,max(Rxhat_db)]);
    plot(omg/pi,Rxhat_db,'g'); hold on;
end
title('Estimated Spectrum','fontsize',title_fontsize);
xlabel('normalized frequency','fontsize',label_fontsize);
ylabel('R_x(e^{j\omega})','fontsize',label_fontsize);
axis([0,1,floor(Rxhatdbmin/10)*10,ceil(Rxhatdbmax/10)*10]);
text(0.6,2,['R_v = ',num2str(Rv)],'fontsize',text_fontsize);
```

The plot is shown in Figure 9.7.

- (d) The effect of subtracting a small amount of noise from $r_x(0)$ on the LS estimate of power spectrum using forward-backward linear predictor with $P = 2$ and $\sigma_v^2 = 1$:

```
% (d) Spectrum Estimation using LS (FB-predictor) Approach (Rv = 1)
```

```

N = 256; n = 0:N-1;
Rv = 1; sigv = sqrt(Rv);
P = 2;

% Loop over realizations
M = 10;
subplot(2,2,4);
Rxhatdbmin = 0; Rxhatdbmax = 0;
for i = 1:M
    % Generate y(n) and x(n)
    w = randn(N,1);
    y = filter(b,a,w);
    v = sigv*randn(N,1);
    x = y + v;
    % Estimate model parameters
    Rbar = lsmatrix(x,P+1);
    Rbar = Rbar - diag(N*ones(length(Rbar),1));
    Rfb = Rbar + flipud(fliplr(Rbar));
    [ahat,VarP] = olsigest(Rfb,1);
    Hhat = freqz(b,[1;ahat]',omg);
    Rxhat = abs(Hhat).*abs(Hhat);
    Rxhat_db = 10*log10(Rxhat);
    Rxhatdbmin = min([Rxhatdbmin,min(Rxhat_db)]);
    Rxhatdbmax = max([Rxhatdbmax,max(Rxhat_db)]);
    plot(omg/pi,Rxhat_db,'g'); hold on;
end
title('Modified Estimated Spectrum','fontsize',title_fontsize);
xlabel('normalized frequency','fontsize',label_fontsize);
ylabel('R_x(e^{j\omega})','fontsize',label_fontsize);
axis([0,1,floor(Rxhatdbmin/10)*10,ceil(Rxhatdbmax/10)*10]);
text(0.6,2,['R_v = ',num2str(Rv)],'fontsize',text_fontsize);

```

The plot is shown in Figure 9.7.

9.8 The first five estimated correlation lag values of a process $x(n)$ are

$$r_x(0) = 1, r_x(1) = 0.7, r_x(2) = 0.5, r_x(3) = 0.3, \text{ and } r_x(4) = 0$$

(a) The Blackman-Tukey Power spectrum estimates:

```

rx = [1;0.75;0.5;0.25;0]; L = length(rx);

% (a) Blackman-Tukey Estimates
Nfft = 1024;
RxBT = real(fft([rx;zeros(Nfft-2*L+1,1);flipud(rx(2:end))]));
RxBT = RxBT(1:Nfft/2+1);
omg = [0:Nfft/2]*2*pi/Nfft;

Hf_1 = figure('units',SCRUN,'position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',[0,0,6,4]);

```

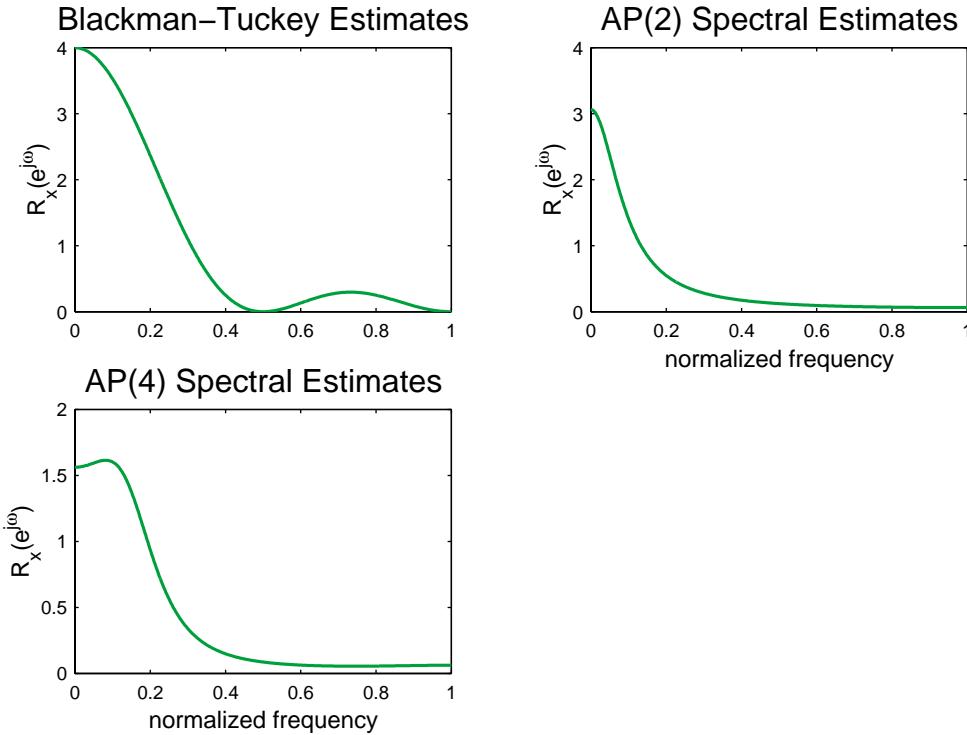


Figure 9.8: Plots of various spectra in P9.8

```

set(Hf_1,'NumberTitle','off','Name','Pr0908');

subplot(2,2,1);
plot(omg/pi,RxBT,'g','linewidth',1);
title('Blackman-Tuckey Estimates','fontsize',title_fontsize);
%xlabel('normalized frequency','fontsize',8);
ylabel('R_x(e^{j\omega})','fontsize',label_fontsize);
axis([0,1,floor(min(RxBT)/1.0)*1.0,ceil(max(RxBT)/1.0)*1.0]);

```

The plot is shown in Figure 9.8.

(b) Spectrum estimates using AP(2) model:

```

% (b) Spectrum Estimation using AP(2) model
P = 2;
Rbar = toeplitz(rx(1:P),rx(1:P));
[ahat,VarP] = olsigest(Rbar,1);
Hhat = freqz(VarP,[1;ahat]',omg);
RxAP = abs(Hhat).*abs(Hhat);
subplot(2,2,2); plot(omg/pi,RxAP,'g','linewidth',1); hold on;

title('AP(2) Spectral Estimates','fontsize',title_fontsize);
xlabel('normalized frequency','fontsize',label_fontsize);
ylabel('R_x(e^{j\omega})','fontsize',label_fontsize);
axis([0,1,0,ceil(max(RxAP)/1.0)*1.0]);

```

The plot is shown in Figure 9.8.

(c) Spectrum estimates using AP(4) model:

```
% (c) Spectrum Estimation using AP(4) model
P = 4;
Rbar = toeplitz(rx(1:P),rx(1:P));
[ahat,VarP] = olsigest(Rbar,1);
Hhat = freqz(VarP,[1;ahat]',omg);
RxAP = abs(Hhat).*abs(Hhat);
subplot(2,2,3); plot(omg/pi,RxAP,'g','linewidth',1); hold on;

title('AP(4) Spectral Estimates','fontsize',title_fontsize);
xlabel('normalized frequency','fontsize',label_fontsize);
ylabel('R_x(e^{j\omega})','fontsize',label_fontsize);
axis([0,1,0,ceil(max(RxAP)/1.0)*1.0]);
```

The plot is shown in Figure 9.8.

9.9 The narrowband process $x(n)$ is generated using the AP(4) model

$$H(z) = \frac{1}{1 + 0.98z^{-1} + 1.92z^{-2} + 0.94z^{-3} + 0.92z^{-4}}$$

driven by WGN(0, 0.001).

(a) Matlab script to compute and plot the true spectrum $R(e^{j\omega})$:

```
% (a) Plot of the true spectrum of x(n)
b = [1]; a = [1,0.98,1.92,0.94,0.92]; % AP(4) signal y(n) parameters
omg = [0:500]*pi/500; % 501 frequency points over 0 to pi
H = freqz(b,a,omg); % AP(4) filter response
Rx = abs(H).*abs(H); % Spectrum of x(n)
Rx = 10*log10(Rx); % Spectrum in dB

Hf_1 = figure('units',SCRUN,'position',SCRPOS+[0,0,0,0.2],...
    'paperunits',PAPUN,'paperposition',[0,0,6,6]);
set(Hf_1,'NumberTitle','off','Name','Pr0909');

subplot(3,2,1);
plot(omg/pi,Rx,'g'); title('True Spectrum','fontsize',title_fontsize);
% xlabel('normalized frequency','fontsize',label_fontsize);
ylabel('R_x(e^{j\omega})','fontsize',label_fontsize);
axis([0,1,floor(min(Rx)/10)*10,ceil(max(Rx)/10)*10]);
```

The plot is shown in Figure 9.9.

(b) LS estimate of power spectrum using forward linear predictor with $P = 4$:

```
% (b) Spectrum Estimation using LS (Forward-predictor) Approach (P = 4)
N = 256; n = 0:N-1;
P = 4;

% Loop over realizations
M = 10;
subplot(3,2,2);
for i = 1:M
```

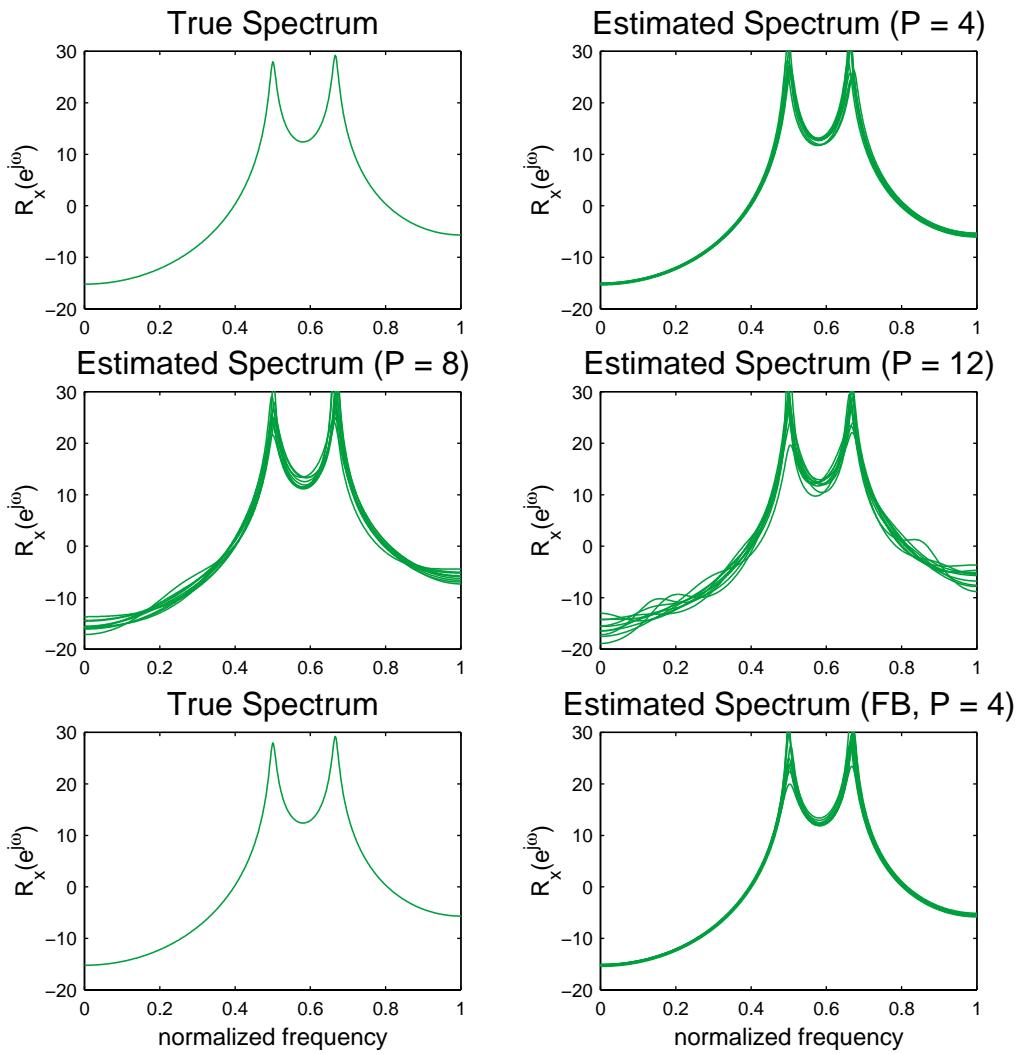


Figure 9.9: Plots of various spectra in P9.9

```
% Generate x(n)
w = randn(N,1);
x = filter(b,a,w);
% Estimate model parameters
Rbar = lsmatrix(x,P+1); [ahat,VarP] = olsigest(Rbar,1);
%[ahat,VarP] = armmodcov(x,P);
Hhat = freqz(b,[1;ahat]',omg);
Rxhat = abs(Hhat).*abs(Hhat); Rxhat = 10*log10(Rxhat);
plot(omg/pi,Rxhat,'g'); hold on;
end
title('Estimated Spectrum (P = 4)','fontsize',title_fontsize);
%xlabel('normalized frequency','fontsize',label_fontsize);
ylabel('R_x(e^{j\omega})','fontsize',label_fontsize);
axis([0,1,floor(min(Rx)/10)*10,ceil(max(Rx)/10)*10]);
```

The plot is shown in Figure 9.9.

- (c) LS estimate of power spectrum using forward linear predictor with $P = 8$ and $P = 12$:

```
% (c) Spectrum Estimation using LS (Forward-predictor) Approach (P = 8 and 10)
P = 8;

% Loop over realizations
M = 10;
subplot(3,2,3);
for i = 1:M
    % Generate x(n)
    w = randn(N,1);
    x = filter(b,a,w);
    % Estimate model parameters
    Rbar = lsmatrix(x,P+1); [ahat,VarP] = olsigest(Rbar,1);
    %[ahat,VarP] = armocov(x,P);
    Hhat = freqz(b,[1;ahat]',omg);
    Rxhat = abs(Hhat).*abs(Hhat); Rxhat = 10*log10(Rxhat);
    plot(omg/pi,Rxhat,'g'); hold on;
end
title('Estimated Spectrum (P = 8)','fontsize',title_fontsize);
% xlabel('normalized frequency','fontsize',label_fontsize);
ylabel('R_x(e^{j\omega})','fontsize',label_fontsize);
axis([0,1,floor(min(Rx)/10)*10,ceil(max(Rx)/10)*10]);

% Loop over realizations (P = 12)
P = 12; M = 10;
subplot(3,2,4);
for i = 1:M
    % Generate x(n)
    w = randn(N,1);
    x = filter(b,a,w);
    % Estimate model parameters
    Rbar = lsmatrix(x,P+1); [ahat,VarP] = olsigest(Rbar,1);
    %[ahat,VarP] = armocov(x,P);
    Hhat = freqz(b,[1;ahat]',omg);
    Rxhat = abs(Hhat).*abs(Hhat); Rxhat = 10*log10(Rxhat);
    plot(omg/pi,Rxhat,'g'); hold on;
end
title('Estimated Spectrum (P = 12)', 'fontsize', title_fontsize);
% xlabel('normalized frequency','fontsize',label_fontsize);
ylabel('R_x(e^{j\omega})','fontsize',label_fontsize);
axis([0,1,floor(min(Rx)/10)*10,ceil(max(Rx)/10)*10]);
```

The plots are shown in Figure 9.9.

- (d) LS estimate of power spectrum using forward-backward linear predictor with $P = 4$:

```
% (d) Spectrum Estimation using LS (FB-predictor) Approach (P = 4)
P = 4;

% Loop over realizations
M = 10;
```

```

subplot(3,2,5);
plot(omg/pi,Rx,'g'); title('True Spectrum','fontsize',title_fontsize);
xlabel('normalized frequency','fontsize',label_fontsize);
ylabel('R_x(e^{j\omega})','fontsize',label_fontsize);
axis([0,1,floor(min(Rx)/10)*10,ceil(max(Rx)/10)*10]);

subplot(3,2,6);
for i = 1:M
    % Generate x(n)
    w = randn(N,1);
    x = filter(b,a,w);
    % Estimate model parameters
    [ahat,VarP] = armocov(x,P);
    Hhat = freqz(b,[1;ahat]',omg);
    Rxhat = abs(Hhat).*abs(Hhat); Rxhat = 10*log10(Rxhat);
    plot(omg/pi,Rxhat,'g'); hold on;
end
title('Estimated Spectrum (FB, P = 4)','fontsize',title_fontsize);
xlabel('normalized frequency','fontsize',label_fontsize);
ylabel('R_x(e^{j\omega})','fontsize',label_fontsize);
axis([0,1,floor(min(Rx)/10)*10,ceil(max(Rx)/10)*10]);

```

The plot is shown in Figure 9.9.

9.10 An ARMA process $x(n)$ is generated using the PZ(4,2) model

$$H(z) = \frac{1 - z^{-2}}{1 + 0.41z^{-4}}$$

driven by WGN(0, 1).

(a) Matlab script to compute and plot the true spectrum $R(e^{j\omega})$:

```

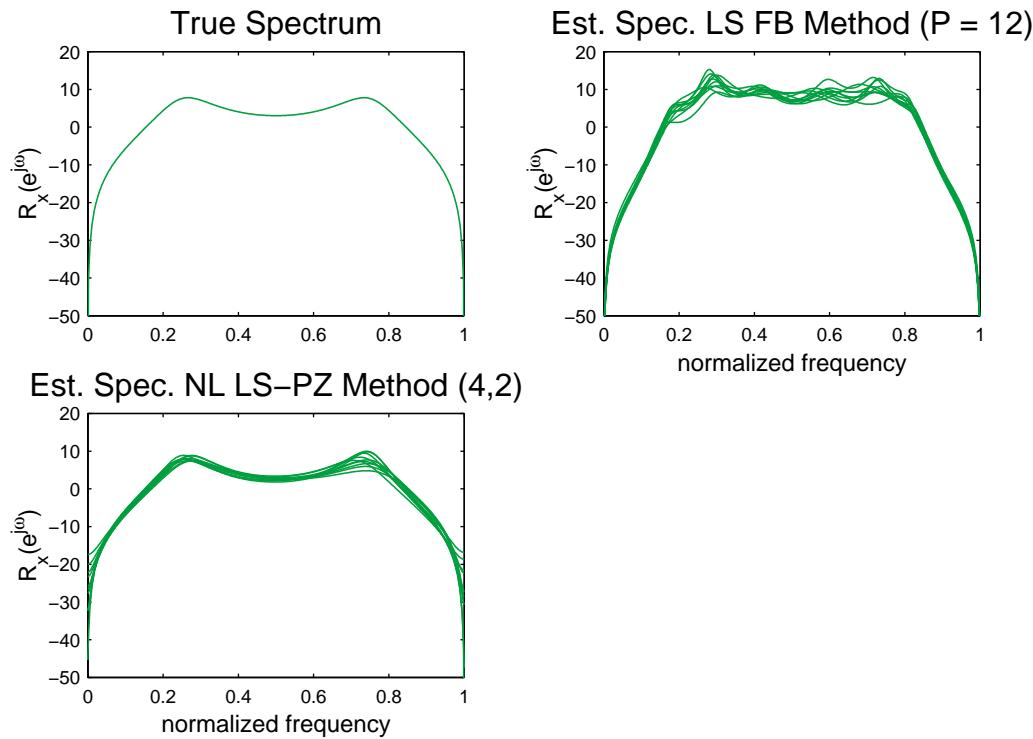
% (a) Plot of the true spectrum of x(n)
b = [1,0,-1]; a = [1,0,0,0,0.41];      % AP(4) signal y(n) parameters
omg = [0:500]*pi/500;                  % 501 frequency points over 0 to pi
H = freqz(b,a,omg);                   % AP(4) filter response
Rx = abs(H).*abs(H);                  % Spectrum of x(n)
Rx = 10*log10(Rx + eps);              % Spectrum in dB

Hf_1 = figure('units',SCRUN,'position',SCRPOS+[0,0,0,0],...
    'paperunits',PAPUN,'paperposition',[0,0,6,4]);
set(Hf_1,'NumberTitle','off','Name','Pr0910');

subplot(2,2,1);
plot(omg/pi,Rx,'g');
title('True Spectrum','fontsize',title_fontsize);
%xlabel('normalized frequency','fontsize',label_fontsize);
ylabel('R_x(e^{j\omega})','fontsize',label_fontsize);
axis([0,1,-50,20]);

```

The plot is shown in Figure 9.10.

**Figure 9.10:** Plots of various spectra in P9.10

- (b) LS estimate of power spectrum using forward-backward linear predictor with $P = 12$:

```
% (b) Spectrum Estimation using LS (Forward-predictor) Approach (P = 12)
N = 256; n = 0:N-1;
P = 12;

% Loop over realizations
M = 10;
subplot(2,2,2);
for i = 1:M
    % Generate x(n)
    w = randn(N,1);
    x = filter(b,a,w);
    % Estimate model parameters
    [ahat,VarP] = armmodcov(x,P);
    Hhat = freqz(b,[1;ahat]',omg);
    Rxhat = abs(Hhat).*abs(Hhat); Rxhat = 10*log10(Rxhat+eps);
    plot(omg/pi,Rxhat,'g'); hold on;
end
title('Est. Spec. LS FB Method (P = 12)', 'fontsize', title_fontsize);
xlabel('normalized frequency', 'fontsize', label_fontsize);
ylabel('R_x(e^{j\omega})', 'fontsize', label_fontsize);
axis([0,1,-50,20]);
```

The plot is shown in Figure 9.10.

- (c) Nonlinear LS estimate of power spectrum using pole-zero modeling algorithm with $P = 4$ and $Q = 2$:

```
% (c) Spectrum Estimation using Nonlinear LS-PZ algorithm of 9.3.3
P = 4; Q = 2;

% Loop over realizations
M = 10;
subplot(2,2,3);
for i = 1:M
    % Generate x(n)
    w = randn(N,1);
    x = filter(b,a,w);
    % Estimate model parameters
    [ahat,bhat,V,FPE,C] = armals(x,P,Q,100,1e-4);
    Hhat = freqz(bhat,ahat,omg);
    Rxhat = abs(Hhat).*abs(Hhat); Rxhat = 10*log10(Rxhat);
    plot(omg/pi,Rxhat,'g'); hold on;
end
title('Est. Spec. NL LS-PZ Method (4,2)', 'fontsize', title_fontsize);
xlabel('normalized frequency', 'fontsize', label_fontsize);
ylabel('R_x(e^{j\omega})', 'fontsize', label_fontsize);
axis([0,1,-50,20]);
```

The plots are shown in Figure 9.10.

9.11 A random process $x(n)$ is given by

$$x(n) = \cos\left(\frac{\pi n}{3} + \theta_1\right) + w(n) - w(n-2) + \cos\left(\frac{2\pi n}{3} + \theta_2\right)$$

where $w(n) \sim \text{WGN}(0, 1)$ and θ_1 and θ_2 are IID random variables uniformly distributed over $[0, 2\pi]$.

(a) Matlab script to compute and plot the true spectrum $R(e^{j\omega})$:

```
% (a) Plot of the true spectrum of x(n)
omg1 = pi/3; omg2 = 2*pi/3;
omg = [0:300]*pi/300; % 301 frequency points over 0 to pi
Rx = 2*(1-cos(2*omg)); % Spectrum of x(n)

Hf_1 = figure('units','SCRUN','position',SCRPOS+[0,0,0,0],...
    'paperunits','PAPUN','paperposition',[0,0,6,4]);
set(Hf_1,'NumberTitle','off','Name','Pr0911a');

subplot(2,2,1);
plot(omg/pi,Rx,'g');
title('True Spectrum', 'fontsize', title_fontsize);
%xlabel('normalized frequency', 'fontsize', label_fontsize);
ylabel('R_x(e^{j\omega})', 'fontsize', label_fontsize);
axis([0,1,0,5]); hold on;
plot([omg1/pi,omg1/pi],[0,4],'c');
fill([1/3-0.03,1/3+0.03,1/3,1/3-0.03],[4,4,4+0.3,4],'c');
plot([omg2/pi,omg2/pi],[0,4],'m');
fill([2/3-0.03,2/3+0.03,2/3,2/3-0.03],[4,4,4+0.3,4],'m');
```

```
set(gca,'xtick',[0,1/3,2/3,1]);
set(gca,'xticklabel',[ ' 0 ';' 1/3 ';' 2/3 ';' 1 '])
```

The plot is shown in Figure 9.11a.

- (b) LS estimate of power spectrum using forward-backward linear predictor with $P = 10, 20$, and 40 :

```
% Generate x(n)
N = 256; n = [0:N-1]';
w = randn(N,1); y = filter([1,0,-1],1,w);
x = y + cos(pi*n/3+rand(1)*2*pi) + cos(2*pi*n/3+rand(1)*2*pi);

% (b) Spectrum Estimation using LS (FB-predictor) Approach (P = 10,20,40)

% Estimate model parameters (P = 10)
P = 10;
[ahat,VarP] = armmodcov(x,P);
Hhat = freqz(1,[1;ahat]',omg);
Rxhat = abs(Hhat).*abs(Hhat);

subplot(2,2,2);
plot(omg/pi,Rxhat,'g'); hold on;
title('Est. Spec. LS FB Method (P = 10)','fontsize',title_fontsize);
% xlabel('normalized frequency','fontsize',label_fontsize);
ylabel('R_x(e^{j\omega})','fontsize',label_fontsize); %axis([0,1,-50,20]);
set(gca,'xtick',[0,1/3,2/3,1]);
set(gca,'xticklabel',[ ' 0 ';' 1/3 ';' 2/3 ';' 1 '])

% Estimate model parameters (P = 10)
P = 20;
[ahat,VarP] = armmodcov(x,P);
Hhat = freqz(1,[1;ahat]',omg);
Rxhat = abs(Hhat).*abs(Hhat);

subplot(2,2,3);
plot(omg/pi,Rxhat,'g'); hold on;
title('Est. Spec. LS FB Method (P = 20)', 'fontsize',title_fontsize);
xlabel('normalized frequency','fontsize',label_fontsize);
ylabel('R_x(e^{j\omega})','fontsize',label_fontsize); %axis([0,1,-50,20]);
set(gca,'xtick',[0,1/3,2/3,1]);
set(gca,'xticklabel',[ ' 0 ';' 1/3 ';' 2/3 ';' 1 '])

% Estimate model parameters (P = 10)
P = 40;
[ahat,VarP] = armmodcov(x,P);
Hhat = freqz(1,[1;ahat]',omg);
Rxhat = abs(Hhat).*abs(Hhat);

subplot(2,2,4);
plot(omg/pi,Rxhat,'g'); hold on;
title('Est. Spec. LS FB Method (P = 40)', 'fontsize',title_fontsize);
```

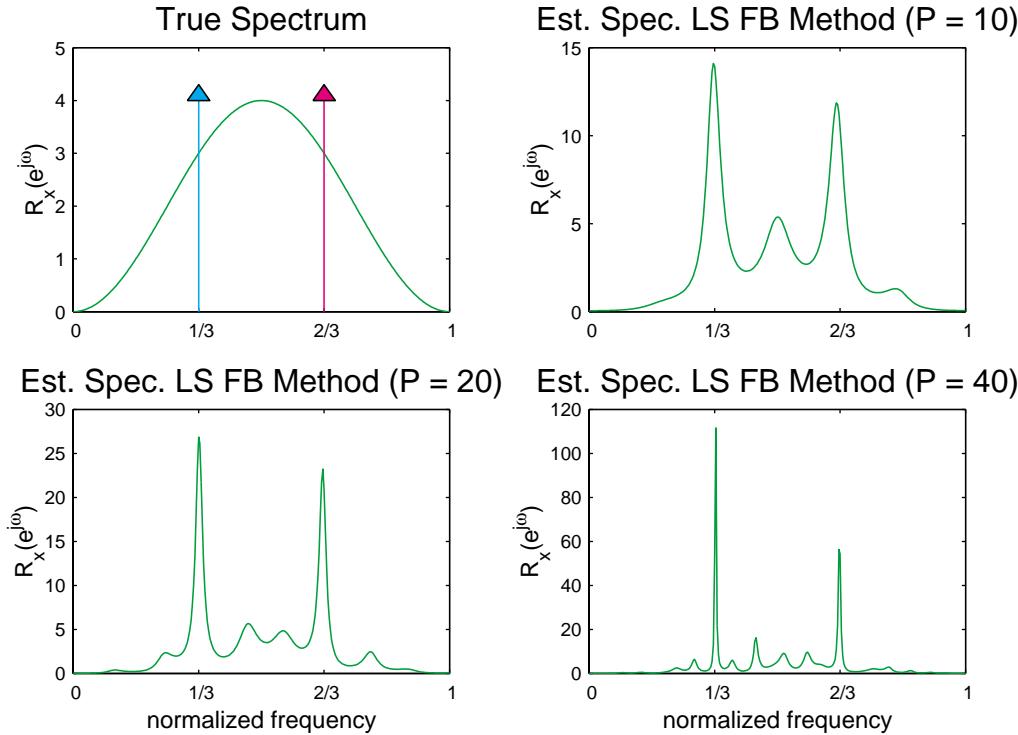


Figure 9.11a: Plots of various spectra in P9.11a and b

```

xlabel('normalized frequency', 'fontsize', label_fontsize);
ylabel('R_x(e^{j\omega})', 'fontsize', label_fontsize); %axis([0,1,-50,20]);
set(gca, 'xtick', [0,1/3,2/3,1]);
set(gca, 'xticklabel', [' 0 ';' 1/3 ';' 2/3 ';' 1 '])

exportfig(gcf, 'p0911a.eps', 'fontmode', 'scaled',...
    'linemode', 'scaled', 'color', 'cmyk');

```

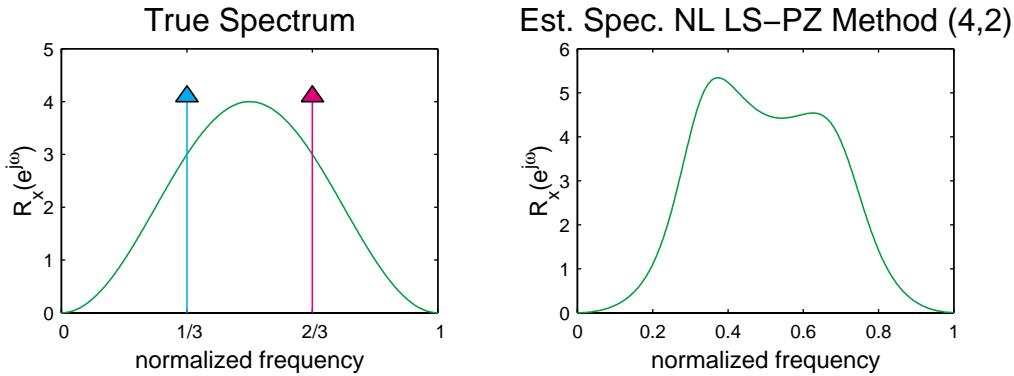
The plots are shown in Figure 9.11a.

- (c) Nonlinear LS estimate of power spectrum using pole-zero modeling algorithm with $P = 4$ and $Q = 2$:

```

% (c) Spectrum Estimation usinf NL PZ modeling algorithm of Sec. 9.3.3
Hf_2 = figure('units',SCRUN,'position',SCRPOS+[0,0,0,0],...
    'paperunits',PAPUN,'paperposition',[0,0,6,4]);
set(Hf_2, 'NumberTitle', 'off', 'Name', 'Pr0911b');
subplot(2,2,3);
plot(omg/pi,Rx,'g'); title('True Spectrum', 'fontsize', title_fontsize);
xlabel('normalized frequency', 'fontsize', label_fontsize);
ylabel('R_x(e^{j\omega})', 'fontsize', label_fontsize);
axis([0,1,0,5]); hold on;
plot([omg1/pi,omg1/pi],[0,4],'c');
fill([1/3-0.03,1/3+0.03,1/3,1/3-0.03],[4,4,4+0.3,4],'c');
plot([omg2/pi,omg2/pi],[0,4],'m');
fill([2/3-0.03,2/3+0.03,2/3,2/3-0.03],[4,4,4+0.3,4],'m');
set(gca, 'xtick', [0,1/3,2/3,1]);
set(gca, 'xticklabel', [' 0 ';' 1/3 ';' 2/3 ';' 1 '])

```

**Figure 9.11b:** Plots of various spectra in P9.11c

```
% Estimate model parameters
subplot(2,2,4);
P = 4; Q = 2;
[ahat,bhat,V,FPE,C] = armals(x,P,Q,1000,1e-5);
Hhat = freqz(bhat,ahat,omg);
Rxhat = abs(Hhat).*abs(Hhat);
plot(omg/pi,Rxhat,'g');

title('Est. Spec. NL LS-PZ Method (4,2)', 'fontsize', title_fontsize);
xlabel('normalized frequency', 'fontsize', label_fontsize);
ylabel('R_x(e^{j\omega})', 'fontsize', label_fontsize);
```

The plots are shown in Figure 9.11b.

- 9.12** For large values of N , the correlation matrix obtained using no windowing tends to a Toeplitz Hermitian matrix and as $N \rightarrow \infty$, the matrix in fact becomes a Toeplitz Hermitian matrix. Thus the model coefficients can be obtained by the Levinson-Durbin algorithm. Therefore, the modeling error variance estimates are given by (7.4.21) and (7.4.16). Using $P_m = \hat{\sigma}_m^2$, we obtain

$$\begin{aligned}\hat{\sigma}_m^2 &= \hat{\sigma}_{m-1}^2 + \beta_{m-1} k_{m-1}^* \\ &= \hat{\sigma}_{m-1}^2 - \hat{\sigma}_{m-1}^2 k_{m-1} k_{m-1}^* = \hat{\sigma}_{m-1}^2 (1 - |k_{m-1}|^2)\end{aligned}$$

- 9.13** To be completed

- 9.14** An ARMA process $x(n)$ is generated by the PZ model

$$x(n) = 0.3x(n-1) + 0.4x(n-2) + w(n) + 0.25w(n-2)$$

driven by $w(n) \sim \text{WGN}(0, 10)$. The PZ model parameters are estimated by using the equation error method of Section 9.3.1.

- (a) Estimates of model parameters:

```
% Generate input and output signal of the PZ(2,2) model
N = 2000;
b = [1,0,0.25]; a = [1,-0.3,-0.4];
varw = 10; w = sqrt(varw)*randn(N,1);
```

```

x = filter(b,a,w);

% (a) Estimates of PZ(2,2) model coefficients
P = 2; Q = 2;
[ahat,bhat,varwhat,FPE] = pzls(x,w,P,Q);

Model Parameters
bhat: 1.0000 -0.0254 0.2386
ahat: 1.0000 -0.2932 -0.4198

```

- (b) Estimate of error variance:

```
% (b) Input variance Estimate
```

```

True error variance: 10.0000
Estimated error variance: 10.3212

```

9.15 An ARMA process $x(n)$ is generated by the PZ(4,2) model

$$\begin{aligned} x(n) = & 1.8766x(n-1) - 2.6192x(n-2) + 1.6936x(n-3) - 0.8145x(n-4) \\ & + w(n) + 0.05w(n-1) - 0.855w(n-2) \end{aligned}$$

driven by $w(n) \sim \text{WGN}(0, 10)$.

- (a) Parameter and error variance estimation using nonlinear LS pole-zero modeling algorithm:

```

% Generate input and output signal of the PZ(4,2) model
N = 300;
b = [1,0.05,-0.855]; a = [1,-1.8766,2.6192,-1.6936,0.8145];
varw = 10; w = sqrt(varw)*randn(N,1);
x = filter(b,a,w);

% (a) Estimates of PZ(4,2) model coefficients
P = 4; Q = 2;
[ahat,bhat,varwhat,FPE,C] = armals(x,P,Q,100,0.001);

Model Parameters
bhat: 1.0000 0.0245 -0.8911
ahat: 1.0000 -1.9019 2.6539 -1.7316 0.8217

True error variance: 10.0000
Estimated error variance: 10.0175

```

- (b) Error variance estimation using AP(10) modeling algorithm:

```

% (b) Estimates using AP(10) model
P2 = 10;
%Rbar = lsmatrix(x,P2+1); [ahat2,VarP] = olsigest(Rbar,1);
[ahat2,e,VarP] = arls(x,P2);

Estimated error variance: 10.4861

```

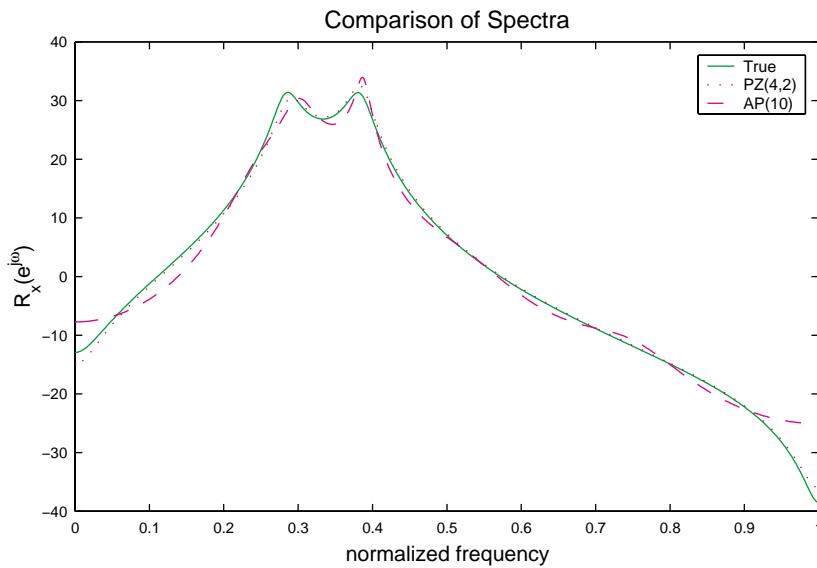


Figure 9.15: Comparison plots of various spectra in P9.15

(c) Plots:

```
% (c) Plots
Hf_1 = figure('units','SCRUN','position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',[0,0,5,3]);
set(Hf_1,'NumberTitle','off','Name','Pr0915');

% True Spectra
omg = [0:500]*pi/500;
H = freqz(b,a,omg);
Rx = abs(H).*abs(H);
Rx = 10*log10(Rx);
plot(omg/pi,Rx,'g');
title('Comparison of Spectra','fontsize',title_fontsize);
xlabel('normalized frequency','fontsize',label_fontsize);
ylabel('R_x(e^{j\omega})','fontsize',label_fontsize);
axis([0,1,floor(min(Rx)/10)*10,ceil(max(Rx)/10)*10]); hold on;

% PZ(4,2) model spectra
H = freqz(bhat,ahat,omg);
Rx = abs(H).*abs(H);
Rx = 10*log10(Rx);
plot(omg/pi,Rx,'r:');
plot(omg/pi,Rx,'r:');

% AP(10) model Spectra
H = freqz(1,ahat2,omg);
Rx = abs(H).*abs(H);
Rx = 10*log10(Rx);
plot(omg/pi,Rx,'m--');

legend('True','PZ(4,2)','AP(10)')
```

The comparison plots are shown in Figure 9.15.

9.16 To be completed

9.17 An ARMA process $x(n)$ is generated by the PZ(4,2) model

$$\begin{aligned}x(n) = & 1.8766x(n-1) - 2.6192x(n-2) + 1.6936x(n-3) - 0.8145x(n-4) \\& + w(n) + 0.05w(n-1) - 0.855w(n-2)\end{aligned}$$

driven by $w(n) \sim \text{WGN}(0, 10)$.

(a) AP(5) model and its spectrum

```
% Generate input and output signal of the PZ(4,2) model
N = 300;
b = [1,0.05,-0.855]; a = [1,-1.8766,2.6192,-1.6936,0.8145];
varw = 10; w = sqrt(varw)*randn(N,1);
x = filter(b,a,w);
Hf_1 = figure('units','SCRUN','position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',[0,0,5,3]);
set(Hf_1,'NumberTitle','off','Name','Pr0917');

% (a) Estimates using AP(5) model
P1 = 5;
[ahat1,e,VarP1] = arls(x,P1);

% AP(5) model spectra
H = freqz(1,ahat1,omg);
Rx = abs(H).*abs(H);
Rx = 10*log10(Rx);
plot(omg/pi,Rx,'r:','linewidth',1);
```

The plot is shown in Figure 9.17.

(b) AP(10) model and its spectrum

```
% (b) Estimates using AP(10) model
P2 = 10;
[ahat2,e,VarP2] = arls(x,P2);

% AP(10) model Spectra
H = freqz(1,ahat2,omg);
Rx = abs(H).*abs(H);
Rx = 10*log10(Rx);
plot(omg/pi,Rx,'m--','linewidth',1);
```

The plot is shown in Figure 9.17.

(c) AP(50) model and its spectrum

```
% (c) Estimates using AP(50) model
P3 = 50;
[ahat3,e,VarP3] = arls(x,P3);
```

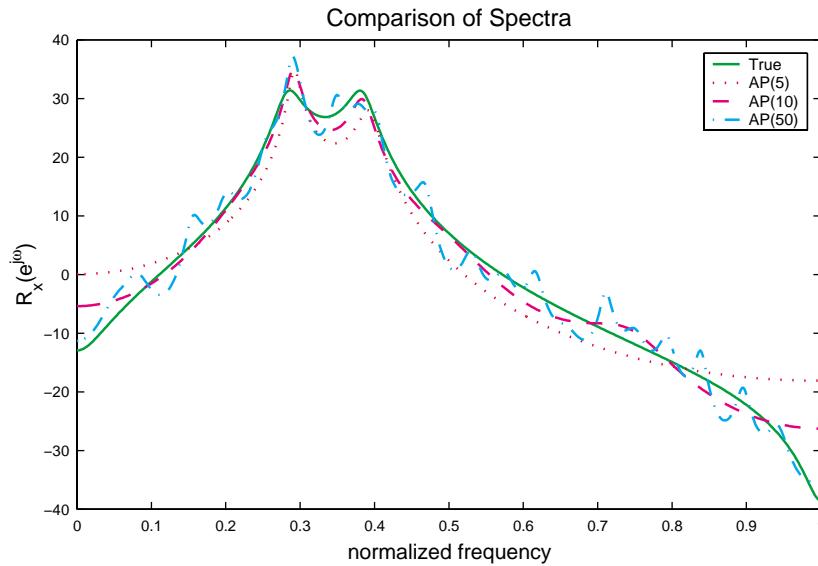


Figure 9.17: Comparison plots of various spectra in P9.17

```
% AP(50) model Spectra
H = freqz(1,ahat3,omg);
Rx = abs(H).*abs(H);
Rx = 10*log10(Rx);
plot(omg/pi,Rx,'c-.','linewidth',1);
```

The plot is shown in Figure 9.17.

(d) True spectrum

```
% (d) True Spectra
omg = [0:500]*pi/500;
H = freqz(b,a,omg);
Rx = abs(H).*abs(H);
Rx = 10*log10(Rx);
plot(omg/pi,Rx,'g','linewidth',1);
title('Comparison of Spectra','fontsize',title_fontsize);
xlabel('normalized frequency','fontsize',label_fontsize);
ylabel('R_x(e^{j\omega})','fontsize',label_fontsize);
axis([0,1,floor(min(Rx)/10)*10,ceil(max(Rx)/10)*10]); hold on;

legend('True','AP(5)','AP(10)','AP(50)')
```

The plot is shown in Figure 9.17.

9.18 Spectral estimation of a speech signal

(a) Periodogram of the speech signal:

```
% Read the speech file
x = load('testspee.dat'); Fs = 10000;
x = filter([1,-1],1,x);
N = length(x);
```

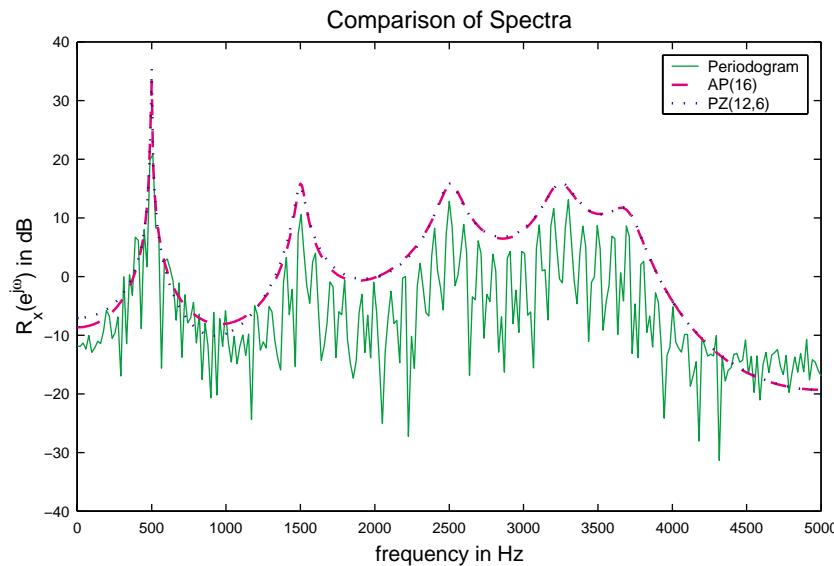


Figure 9.18: Comparison plots of various spectra in P9.18

```

%(a) Periodogram of the speech signal
win = boxcar(N); Nfft = 2^(ceil(log10(N)/log10(2)));
Rx = psd(x,Nfft,Fs,win,'none');

(b) Spectral estimate based on AP(16) model:
%(b) Estimates using AP(16) model
P2 = 16;
[ahat2,e,VarP] = arls(x,P2);

(c) Spectral estimate based on PZ(12,6) model:
% (c) Estimates using PZ(12,6) model
P = 12; Q = 6;
[ahat,bhat,varwhat,FPE,C] = armals(x,P,Q,100,0.001);

(d) Plots:
% (d) Plots
% Periodogram Spectra
freq = [0:Nfft/2]*Fs/Nfft;
Rx1 = 10*log10(Rx)-60;
plot(freq,Rx1,'g');
title('Comparison of Spectra','fontsize',title_fontsize);
xlabel('frequency in Hz','fontsize',label_fontsize);
ylabel('R_x(e^{j\omega}) in dB','fontsize',label_fontsize);
hold on;

% AP(16) model Spectra
omg = [0:500]*pi/500;
H = freqz(1,ahat2,omg);
Rx = abs(H).*abs(H);
Rx2 = 10*log10(Rx);

```

```

plot(omg/pi*Fs/2,Rx2,'m--','linewidth',1);

% PZ(12,6) model spectra
H = freqz(bhat,ahat,omg);
Rx = abs(H).*abs(H);
Rx3 = 10*log10(Rx);
plot(omg/pi*Fs/2,Rx3,'b:','linewidth',1);

Rxmin = min([min(Rx1),min(Rx2),min(Rx3)]);
Rxmax = max([max(Rx1),max(Rx2),max(Rx3)]);
axis([0,Fs/2,floor(Rxmin/10)*10,ceil(Rxmax/10)*10]); hold off;
legend('Periodogram','AP(16)','PZ(12,6)')

```

The plots are shown in Figure 9.18.

9.19 Spectral estimation using prewhitening and postcoloring method.

- (a) The Matlab function psd_lpprew:

```

function [Rx] = PSD_lpprew(x,P,L,Nfft);

Nfft = max(Nfft,2*L);
Nfft = 2^(ceil(log10(Nfft)/log10(2)));
x = x-mean(x);
noverlap = L/2;

% Fit an AP(P) model using forward-backward linear prediction
[ahat,VarP] = armodcov(x,P);
a = [1;ahat];

% Compute residuals
e = filter(a,1,x);
e = e - mean(e); var_e = (std(e))^2;
N = length(e);

%% Determine Welch PSD estimate of the residuals
Re = psd(e,Nfft,2,hamming(L),L/2,'none');

%% Perform postcoloring to obtain the PSD
w = [0:Nfft/2]*2*pi/Nfft;
A = freqz(1,a,w);
magAsq = (abs(A')).^2;
Rx = Re(1:Nfft/2+1).*magAsq;

```

- (b) Verification of the function psd_lpprew:

```

% Read the speech file
x = load('testspee.dat'); Fs = 10000;
x = filter([1,-0.95],1,x);
N = length(x);
% Periodogram of the speech signal

```

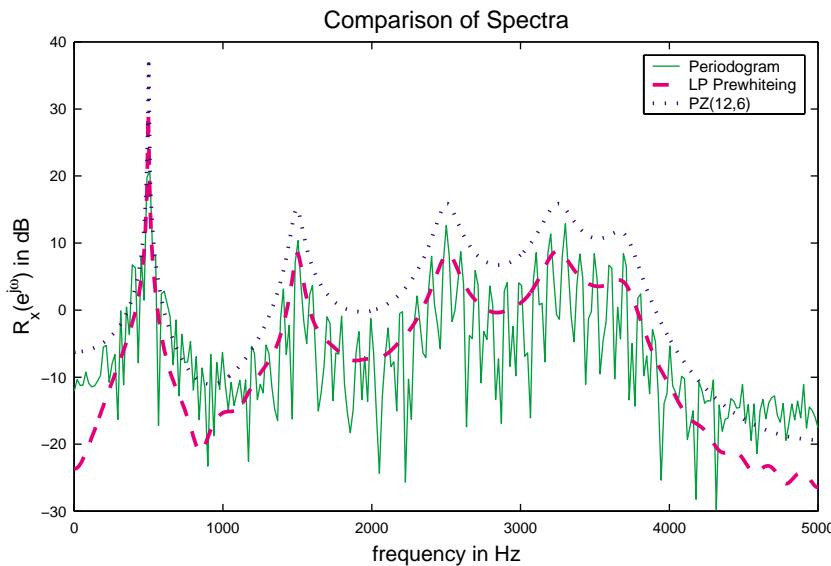


Figure 9.19: Comparison plots of various spectra in P9.19

```

win = boxcar(N); NfftP = 2^(ceil(log10(N)/log10(2)));
Rx1 = psd(x,NfftP,Fs,win,'none');

% (b) Verification of the psd_lpprew function
P = 16; L = 64; Nfft = 1024;
Rx2 = psd_lpprew(x,P,L,Nfft);

(c) Comparison using plots:
% (c) Estimates using PZ(12,6) model
P = 12; Q = 6;
[ahat,bhat,varwhat,FPE,C] = armals(x,P,Q,100,0.001);

% (d) Plots
% Periodogram Spectra
freq = [0:NfftP/2]*Fs/NfftP;
Rx1 = 10*log10(Rx1)-60;
plot(freq,Rx1,'g'); title('Comparison of Spectra','fontsize',10);
xlabel('frequency in Hz','fontsize',8);
ylabel('R_x(e^{j\omega}) in dB','fontsize',8); hold on;
set(gca,'xtick',[0:1000:Fs/2],'fontsize',6)

% LP_PREW Spectra
freq = [0:Nfft/2]*Fs/Nfft;
Rx2 = 10*log10(Rx2)-60;
plot(freq,Rx2,'m--','linewidth',1.5);

% PZ(12,6) model spectra
omg = [0:500]*pi/500;
H = freqz(bhat,ahat,omg);
Rx = abs(H).*abs(H);

```

```

Rx3 = 10*log10(Rx);
plot(omg/pi*Fs/2,Rx3,'b:','linewidth',1.5);

Rxmin = min([min(Rx1),min(Rx2),min(Rx3)]);
Rxmax = max([max(Rx1),max(Rx2),max(Rx3)]);
axis([0,Fs/2,floor(Rxmin/10)*10,ceil(Rxmax/10)*10]); hold off;
legend('Periodogram','LP Prewhiteing','PZ(12,6)')

```

The plots are shown in Figure 9.19.

- 9.20** Consider a white noise process with variance σ_w^2 . The correlation matrix of this white noise process is given by

$$\mathbf{R}_x = \sigma_w^2 \mathbf{I}.$$

The minimum variance distortionless response weight vector for the MVDR spectrum estimator is given by

$$\begin{aligned}
\mathbf{c}_k &= \frac{\sqrt{M} \mathbf{R}_x^{-1} \mathbf{v}(f_k)}{\mathbf{v}^H(f_k) \mathbf{R}_x^{-1} \mathbf{v}(f_k)} \\
&= \frac{\sqrt{M} \mathbf{v}(f_k)}{\mathbf{v}^H(f_k) \mathbf{v}(f_k)} \quad \text{independent of } \sigma_w^2 \\
&= \frac{1}{\sqrt{M}} \mathbf{v}(f_k)
\end{aligned}$$

since $\mathbf{v}^H(f_k) \mathbf{v}(f_k) = M$ from (9.5.6). Therefore, the MVDR spectral estimate of the white noise process is given by

$$\begin{aligned}
\hat{R}_M^{(mv)}(e^{j2\pi f_k}) &= E\{|y(n)|^2\} \\
&= E\left\{|\mathbf{c}_k^H \mathbf{x}(n)|^2\right\} \\
&= \frac{1}{M} \mathbf{v}^H(f_k) \sigma_w^2 \mathbf{I} \mathbf{v}(f_k) \\
&= \sigma_w^2 \frac{1}{M} M \\
&= \sigma_w^2
\end{aligned}$$

- 9.21** The first-order all-pole model is given by

$$x(n) = -a_1 x(n-1) + w(n)$$

where $w(n)$ is white Gaussian noise with variance σ_w^2 . From pg. 167, the autocorrelation function of the AP(1) model is

$$r_x(l) = \frac{\sigma_w^2}{1-a_1^2} (-a_1)^{|l|}$$

Using the minimum-variance spectral estimate from (9.5.11)

$$\hat{R}_M^{(mv)}(e^{j2\pi f}) = \frac{M}{\mathbf{v}^H(f) \mathbf{R}_x^{-1} \mathbf{v}(f)}$$

we see that we must solve for the inverse of the correlation matrix. The M th order correlation matrix of the AP(1) model is

$$\mathbf{R}_x = \frac{\sigma_w^2}{1-a_1^2} \begin{bmatrix} 1 & -a_1 & a_1^2 & \cdots & (-a_1)^{M-1} \\ -a_1 & 1 & -a_1 & & \vdots \\ a_1^2 & -a_1 & 1 & \ddots & \\ \vdots & \ddots & \ddots & \ddots & -a_1 \\ (-a_1)^{M-1} & \cdots & \cdots & -a_1 & 1 \end{bmatrix}$$

>From (9.5.16), the inverse of the autocorrelation matrix is

$$\mathbf{R}_x^{-1} = \mathbf{A}^H \bar{\mathbf{D}}^{-1} \mathbf{A}$$

where

$$\mathbf{A} = \begin{bmatrix} 1 & -a_1 & 0 & \cdots & 0 \\ 0 & 1 & -a_1 & \ddots & \vdots \\ \vdots & 0 & 1 & \ddots & 0 \\ & & \ddots & \ddots & -a_1 \\ 0 & \cdots & & 0 & 1 \end{bmatrix}$$

and

$$\begin{aligned} \bar{\mathbf{D}} &= \text{diag}\{P_M, P_{M-1}, \dots, P_1\} \\ &= \text{diag}\left\{\sigma_w^2, \sigma_w^2, \dots, \frac{\sigma_w^2}{1-a_1^2}\right\} \end{aligned}$$

Therefore,

$$\bar{\mathbf{D}}^{-1} = \frac{1}{\sigma_w^2} \text{diag}\{1, 1, \dots, 1-a_1^2\}$$

and the inverse of the correlation matrix is given by

$$\begin{aligned} \mathbf{R}_x^{-1} &= \mathbf{A}^H \bar{\mathbf{D}}^{-1} \mathbf{A} \\ &= \frac{1}{\sigma_w^2} \begin{bmatrix} 1 & -a_1 & 0 & \cdots & 0 \\ -a_1 & 1+a_1^2 & -a_1 & \ddots & \vdots \\ 0 & -a_1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 1+a_1^2 & -a_1 \\ 0 & \cdots & 0 & -a_1 & 1 \end{bmatrix} \end{aligned}$$

>From (9.5.6), the time-window frequency vector is

$$\mathbf{v}(f) = [1 \ e^{-j2\pi f} \ \dots \ e^{-j2\pi f(M-1)}]^T$$

then

$$\begin{aligned}
 \mathbf{v}^H(f) \mathbf{R}_x^{-1} \mathbf{v}(f) &= \frac{1}{\sigma_w^2} \mathbf{v}^H(f) \begin{bmatrix} 1 & -a_1 & 0 & \cdots & 0 \\ -a_1 & 1 + a_1^2 & -a_1 & \ddots & \vdots \\ 0 & -a_1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 1 + a_1^2 & -a_1 \\ 0 & \cdots & 0 & -a_1 & 1 \end{bmatrix} \mathbf{v}(f) \\
 &= \frac{1}{\sigma_w^2} [2 + (M-2)(1+a_1^2) - a_1(M-1)e^{j2\pi f} - a_1(M-1)e^{-j2\pi f}] \\
 &= \frac{1}{\sigma_w^2} [2 + (M-2)(1+a_1^2) - 2a_1(M-1)\cos 2\pi f]
 \end{aligned}$$

Substituting back in for the minimum-variance spectrum, we have

$$\widehat{R}_M^{(mv)}(e^{j2\pi f}) = \frac{M\sigma_w^2}{2 + (M-2)(1+a_1^2) - 2a_1(M-1)\cos 2\pi f}$$

Note that the actual spectrum of an AR(1) process is

$$R(e^{j2\pi f}) = \frac{1}{1 + a_1^2 - 2a_1 \cos 2\pi f}$$

so as $M \rightarrow \infty$ then $\widehat{R}_M^{(mv)}(e^{j2\pi f}) \rightarrow R(e^{j2\pi f})$.

9.22 >From (9.5.10), the minimum-variance weight vector is given by

$$\mathbf{c}_k = \frac{\sqrt{M} \mathbf{R}_x^{-1} \mathbf{v}(f_k)}{\mathbf{v}^H(f_k) \mathbf{R}_x^{-1} \mathbf{v}(f_k)}$$

Solving the constrained optimization problem

$$\min \mathbf{c}_k^H \mathbf{R}_x \mathbf{c}_k \quad \text{subject to} \quad \mathbf{c}_k^H \mathbf{v}(f_k) = \sqrt{M}$$

we use Lagrange multipliers from Appendix B.2. First, we define the two functions

$$f(\mathbf{c}_k) = \mathbf{c}_k^H \mathbf{R}_x \mathbf{c}_k \quad g(\mathbf{c}_k) = \mathbf{c}_k^H \mathbf{v}(f_k) - \sqrt{M}$$

The Lagrangian function is then

$$\begin{aligned}
 \mathcal{L}(\mathbf{c}_k, \lambda) &= f(\mathbf{c}_k) + \lambda g(\mathbf{c}_k) \\
 &= \mathbf{c}_k^H \mathbf{R}_x \mathbf{c}_k + \lambda [\mathbf{c}_k^H \mathbf{v}(f_k) - \sqrt{M}]
 \end{aligned}$$

where λ is the Lagrange multiplier. We then want to minimize the Lagrangian function with respect to both \mathbf{c}_k and λ . Taking the gradient of $\mathcal{L}(\mathbf{c}_k, \lambda)$ with respect to \mathbf{c}_k and setting it to zero, we have

$$\nabla_{\mathbf{c}_k} \mathcal{L}(\mathbf{c}_k, \lambda) = \mathbf{c}_k^H \mathbf{R}_x + \lambda \mathbf{v}^H(f_k) = 0$$

$$\mathbf{c}_k = -\lambda \mathbf{R}_x^{-1} \mathbf{v}(f_k) \tag{1}$$

Similarly, taking the gradient of $\mathcal{L}(\mathbf{c}_k, \lambda)$ with respect to λ and setting it to zero

$$\nabla_{\lambda} \mathcal{L}(\mathbf{c}_k, \lambda) = \mathbf{c}_k^H \mathbf{v}(f_k) - \sqrt{M} = 0 \tag{2}$$

Substituting for (1) into (2), yields

$$-\lambda^* \mathbf{v}^H(f_k) \mathbf{R}_x^{-1} \mathbf{v}(f_k) - \sqrt{M} = 0$$

and solving for λ

$$\lambda = \frac{-\sqrt{M}}{\mathbf{v}^H(f_k) \mathbf{R}_x^{-1} \mathbf{v}(f_k)} \quad (3)$$

Substituting (3) back into (1), we have the weight vector for the minimum-variance spectral estimator at the frequency f_k

$$\mathbf{c}_k = \frac{\sqrt{M} \mathbf{R}_x^{-1} \mathbf{v}(f_k)}{\mathbf{v}^H(f_k) \mathbf{R}_x^{-1} \mathbf{v}(f_k)}$$

9.23 Recall the relation between the minimum-variance and all-pole model spectrum estimators from (9.5.22)

$$\frac{1}{\widehat{R}_M^{(mv)}(e^{j2\pi f})} = \sum_{m=1}^M \frac{|\mathbf{v}_m^H(f) \mathbf{a}_m|^2}{M P_m} = \frac{1}{M} \sum_{m=1}^M \frac{1}{\widehat{R}_m^{(ap)}(e^{j2\pi f})}$$

Thus, the minimum-variance spectral estimate with a filter of length M is equal to an average of all-pole model spectral estimates from order $m = 1$ to $m = M$. Using this relationship, the minimum-variance spectral estimate with window length $M + 1$ is given by

$$\begin{aligned} \frac{1}{\widehat{R}_{M+1}^{(mv)}(e^{j2\pi f})} &= \frac{1}{M+1} \sum_{m=1}^{M+1} \frac{1}{\widehat{R}_m^{(ap)}(e^{j2\pi f})} \\ &= \frac{1}{M+1} \cdot \frac{1}{\widehat{R}_{M+1}^{(ap)}(e^{j2\pi f})} + \frac{1}{M+1} \sum_{m=1}^M \frac{1}{\widehat{R}_m^{(ap)}(e^{j2\pi f})} \\ &= \frac{1}{M+1} \cdot \frac{1}{\widehat{R}_{M+1}^{(ap)}(e^{j2\pi f})} + \frac{M}{M+1} \cdot \frac{1}{M} \sum_{m=1}^M \frac{1}{\widehat{R}_m^{(ap)}(e^{j2\pi f})} \\ &= \frac{1}{M+1} \cdot \frac{1}{\widehat{R}_{M+1}^{(ap)}(e^{j2\pi f})} + \frac{M}{M+1} \cdot \frac{1}{\widehat{R}_M^{(mv)}(e^{j2\pi f})} \end{aligned}$$

Therefore, the $(M + 1)$ th order minimum-variance spectrum is equal to the M th order minimum-variance spectrum refined by the $(M + 1)$ th order all-pole spectral estimate weighted by the $\frac{M}{M+1}$ and $\frac{1}{M}$, respectively.

9.24 For the Pisarenko harmonic decomposition, the pseudo-spectrum is given by

$$\bar{R}_{phd}(e^{j2\pi f}) = \frac{1}{|\mathbf{v}^H(f) \mathbf{q}_M|^2} = \frac{1}{|Q_M(e^{j2\pi f})|^2}$$

where \mathbf{q}_M is the eigenvector of the correlation matrix \mathbf{R}_x associated with the smallest eigenvalue, i.e., the noise eigenvector. Recall that the order is chosen to be $M = P + 1$ where P is the number of complex exponentials. The frequency estimates are then the $P = M - 1$ peaks in the pseudo-spectrum $\hat{f}_1, \hat{f}_2, \dots, \hat{f}_P$. Using the eigenvector/eigenvalue relation

$$\mathbf{R}_x \mathbf{q}_m = \lambda_m \mathbf{q}_m \quad \text{for } m = 1, 2, \dots, M$$

where recall that the eigenvectors have unit-norm ($\mathbf{q}_m^H \mathbf{q}_m = 1$). Multiplying both sides of the eigenvector/eigenvalue relation by \mathbf{q}_m^H

$$\mathbf{q}_m^H \mathbf{R}_x \mathbf{q}_m = \lambda_m \mathbf{q}_m^H \mathbf{q}_m = \lambda_m \quad (1)$$

Recall from the harmonic model, the correlation matrix from (9.6.7) is given by

$$\mathbf{R}_x = \sum_{p=1}^P A_p \mathbf{v}(f_p) \mathbf{v}^H(f_p) + \sigma_w^2 \mathbf{I} \quad (2)$$

where $A_p = |\alpha_p|^2$ is the power of the p th complex exponential. In the case of the Pisarenko harmonic decomposition, we have chosen $M = P+1$ using our a priori knowledge of the number of complex exponentials. In addition, the smallest eigenvalue corresponding to the noise is $\lambda_M = \sigma_w^2$. Substituting (2) into (1), we have

$$\begin{aligned} \sum_{p=1}^P A_p \mathbf{q}_m^H \mathbf{v}(f_p) \mathbf{v}(f_p)^H \mathbf{q}_m + \sigma_w^2 &= \lambda_m \\ \sum_{p=1}^P A_p |\mathbf{q}_m^H \mathbf{v}(f_p)|^2 &= \sum_{p=1}^P A_p |Q_m(e^{j2\pi f_p})|^2 = \lambda_m - \sigma_w^2 \end{aligned}$$

where $Q_m(e^{j2\pi f_p})$ is the Fourier transform of the eigenvector \mathbf{q}_m evaluated at the frequency f_p . This same equation can be written for all of the frequency estimates $\hat{f}_1, \hat{f}_2, \dots, \hat{f}_P$ from the Pisarenko harmonic decomposition obtained by picking the peaks in the pseudo-spectrum. Writing this set of equations in matrix form we have

$$\begin{bmatrix} |Q_1(e^{j2\pi \hat{f}_1})|^2 & |Q_1(e^{j2\pi \hat{f}_2})|^2 & \dots & |Q_1(e^{j2\pi \hat{f}_P})|^2 \\ |Q_2(e^{j2\pi \hat{f}_1})|^2 & |Q_2(e^{j2\pi \hat{f}_2})|^2 & \dots & |Q_2(e^{j2\pi \hat{f}_P})|^2 \\ \vdots & \vdots & & \vdots \\ |Q_P(e^{j2\pi \hat{f}_1})|^2 & |Q_P(e^{j2\pi \hat{f}_2})|^2 & \dots & |Q_P(e^{j2\pi \hat{f}_P})|^2 \end{bmatrix} \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_P \end{bmatrix} = \begin{bmatrix} \lambda_1 - \lambda_M \\ \lambda_2 - \lambda_M \\ \vdots \\ \lambda_P - \lambda_M \end{bmatrix}$$

This equation can then be solved for the powers of the complex exponentials A_1, A_2, \dots, A_P .

9.25 The minimum-norm pseudo-spectrum from (9.6.41) is given by

$$\bar{R}_{mn}(e^{j2\pi f}) = \frac{1}{|\mathbf{v}^H(f) \mathbf{u}_{mn}|^2}$$

where \mathbf{u}_{mn} is the minimum-norm vector from (9.6.40)

$$\mathbf{u}_{mn} = \frac{\mathbf{P}_w \boldsymbol{\delta}_1}{\boldsymbol{\delta}_1^H \mathbf{P}_w \boldsymbol{\delta}_1}$$

where $\mathbf{P}_w = \mathbf{Q}_w \mathbf{Q}_w^H$ is the noise subspace projection matrix, \mathbf{Q}_w is the matrix of noise eigenvectors, and $\boldsymbol{\delta}_1 = [1 \ 0 \ \dots \ 0]^T$. This guarantees that \mathbf{u}_{mn} is in the noise subspace. Since the complex exponentials are all in the signal subspace which is orthogonal to the noise subspace, they are thus orthogonal to \mathbf{u}_{mn} . Consider the denominator of the minimum-norm pseudo-spectrum

$$\begin{aligned} |\mathbf{v}(f)^H \mathbf{u}_{mn}|^2 &= \left| \sum_{k=1}^M \mathbf{u}_{mn}(k) \cdot e^{-j2\pi f(k-1)} \right|^2 \\ &= |U_{mn}(e^{j2\pi f})|^2 \end{aligned}$$

where $U_{mn}(e^{j2\pi f})$ is the Fourier transform of the minimum-norm vector \mathbf{u}_{mn} . Converting to the z -transform ($z = e^{-j2\pi f}$)

$$U_{mn}(z) = \sum_{k=0}^{M-1} \mathbf{u}_{mn}(k+1) z^{-k}$$

The z-transform of the denominator of the pseudo-spectrum can then be written as the following polynomial

$$\bar{P}_{mn}(z) = |U_{mn}(z)|^2 = U_{mn}(z) U_{mn}^* \left(\frac{1}{z^*} \right)$$

This $(2M-1)$ th order polynomial has $(M-1)$ pairs of roots with one inside and one outside the unit circle. The peaks in the pseudo-spectrum correspond to the roots of this polynomial. Since we assume that the complex exponentials are undamped, their roots should theoretically lie on the unit circle. Therefore, looking at the $M-1$ roots of this polynomial that lie inside the unit circle, we associate the P closest ones to the unit circle with the P complex exponentials. The phases of these roots are then the root minimum-norm frequency estimates. A MATLAB example of the use of the root minimum-norm method is given below:

```
% Routine to show example of root minimum-norm frequency
% estimates
clear

% Number of sinusoids
P = 2; % number of complex exponentials
M = 8; % number of samples in time-window
Nsamples = 128; % number of time samples to generate

% Define signal parameters
f_s = [0.1 0.2]'; % frequencies of complex exponentials
SNR_db = [5]*ones(P,1); % signal-to-noise ratios (dB)

% Generate signal of complex exponentials in noise
x = zeros(Nsamples,1);
phase = rand(P) - 0.5;
for p = 1:P
    alpha(p) = (10^(SNR_db(p)/20))*exp(j*2*pi*phase(p));
    x = x + alpha(p)*exp(j*2*pi*f_s(p)*[0:(Nsamples-1)]');
end

% Add unit variance noise
x = x + (randn(Nsamples,1) + i*randn(Nsamples,1))/sqrt(2);

% estimate correlation matrix
% Generate data matrix
N = length(x) - M + 1;
X = zeros(N,M);
for n = 1:M
    X(:,n) = x((1:N)+(M-n));
end
R = (1/N)*X'*X;

% Compute eigendecomposition and order by descending eigenvalues
[Q0,D] = eig(R);
[lambda,index] = sort(abs(diag(D)));
lambda = lambda(M:-1:1);
```

```

Q=Q0(:,index(M:-1:1));

% Compute noise subspace projection matrix
% and minimum-norm vector
delta1 = zeros(M,1);
delta1(1) = 1;
Pw = Q(:,(P+1):M)*Q(:,(P+1):M)'; % noise subspace projection matrix
u_mn = (Pw*delta1)/(delta1'*Pw*delta1); % minimum-norm vector

% Compute minimum-norm polynomial and root it
Pbar = conv(u_mn,conj(u_mn(M:-1:1)));
r_Pbar = roots(Pbar); % roots of polynomial

% Find the roots closest to the unit circle
[i_min] = find(abs(r_Pbar) < 1);
r_Pbar_min = r_Pbar(i_min); % roots closest to unit circle
freq = angle(r_Pbar_min)/(2*pi);
[r_Vn_order,index] = sort(abs((abs(r_Pbar_min)-1)));
fest = freq(index(1:P)); % frequency estimates

```

9.26 The minimum-variance spectral estimate from (9.5.11) is given by

$$\hat{R}_M^{(mv)}(e^{j2\pi f}) = \frac{M}{\mathbf{v}^H(f) \mathbf{R}_x^{-1} \mathbf{v}(f)} \quad (1)$$

The correlation matrix has the following eigendecomposition

$$\mathbf{R}_x = \sum_{m=1}^M \lambda_m \mathbf{q}_m \mathbf{q}_m^H \quad (2)$$

and the inverse of the correlation matrix is given by

$$\mathbf{R}_x^{-1} = \sum_{m=1}^M \frac{1}{\lambda_m} \mathbf{q}_m \mathbf{q}_m^H \quad (3)$$

Substituting (3) into (1) we have

$$\begin{aligned}
\hat{R}_M^{(mv)}(e^{j2\pi f}) &= \frac{M}{\sum_{m=1}^M \frac{1}{\lambda_m} \mathbf{v}^H(f) \mathbf{q}_m \mathbf{q}_m^H \mathbf{v}(f)} \\
&= \frac{M}{\sum_{m=1}^M \frac{1}{\lambda_m} |\mathbf{v}^H(f) \mathbf{q}_m|^2} \\
&= \frac{M}{\sum_{m=1}^M \frac{1}{\lambda_m} |Q_m(e^{j2\pi f})|^2}
\end{aligned}$$

where $Q_m(e^{j2\pi f})$ is the Fourier transform of the m th eigenvector \mathbf{q}_m . Recall from (9.6.30) and (9.6.31) the pseudo-spectra of the MUSIC and eigenvector methods, respectively

$$\bar{R}_{\text{music}}(e^{j2\pi f}) = \frac{1}{\sum_{m=P+1}^M |Q_m(e^{j2\pi f})|^2}$$

$$\bar{R}_{\text{ev}}(e^{j2\pi f}) = \frac{1}{\sum_{m=P+1}^M \left| \frac{1}{\lambda_m} Q_m(e^{j2\pi f}) \right|^2}$$

If we assume that the estimated noise eigenvalues are all equal to $\lambda_m = \sigma_w^2$ for $m = P + 1, \dots, M$, then

$$\bar{R}_{\text{music}}(e^{j2\pi f}) = \frac{1}{\sigma_w^4} \bar{R}_{\text{ev}}(e^{j2\pi f}) \quad (4)$$

The minimum-variance spectral estimate is related to the eigenvector method pseudo-spectrum as

$$\hat{R}_M^{(mv)}(e^{j2\pi f}) = \frac{M}{\bar{R}_{\text{ev}}(e^{j2\pi f}) + \sum_{m=1}^P \left| \frac{1}{\lambda_m} Q_m(e^{j2\pi f}) \right|^2}$$

The relation for the MUSIC pseudo-spectrum follows from (4).

9.27 Recall the pseudo-spectrum from the MUSIC frequency estimation method from (9.6.30)

$$\bar{R}_{\text{music}}(e^{j2\pi f}) = \frac{1}{\sum_{m=P+1}^M |\mathbf{v}^H(f) \mathbf{q}_m|^2} = \frac{1}{\sum_{m=P+1}^M |Q_m(e^{j2\pi f})|^2}$$

and the minimum-variance spectrum from (9.5.11)

$$\hat{R}_M^{(mv)}(e^{j2\pi f}) = \frac{M}{\mathbf{v}^H(f) \mathbf{R}_x^{-1} \mathbf{v}(f)} \quad (1)$$

The MUSIC pseudo-spectrum can be rewritten as

$$\bar{R}_{\text{music}}(e^{j2\pi f}) = \frac{1}{\mathbf{v}^H(f) \mathbf{P}_w \mathbf{v}(f)} \quad (2)$$

where the noise subspace projection matrix \mathbf{P}_w is given by

$$\mathbf{P}_w = \mathbf{Q}_w \mathbf{Q}_w^H = \mathbf{I} - \mathbf{Q}_s \mathbf{Q}_s^H \quad (3)$$

where \mathbf{Q}_w and \mathbf{Q}_s are matrices whose columns are the noise and signal eigenvectors of the correlation matrix, respectively. From (9.6.7), the correlation matrix of complex exponentials in noise is

$$\mathbf{R}_x = \mathbf{V} \mathbf{A} \mathbf{V}^H + \sigma_w^2 \mathbf{I}$$

where \mathbf{V} is the matrix of frequency vectors of the P complex exponentials

$$\mathbf{V} = [\mathbf{v}(f_1) \ \mathbf{v}(f_2) \ \dots \ \mathbf{v}(f_P)]$$

and \mathbf{A} is the diagonal matrix of their respective powers

$$\mathbf{A} = \text{diag} \{ |\alpha_1|^2, |\alpha_2|^2, \dots, |\alpha_P|^2 \} \quad (4)$$

Using the matrix inversion lemma from Appendix A, the inverse of the correlation matrix of the harmonic model from (4) is

$$\mathbf{R}_x^{-1} = \frac{1}{\sigma_w^2} \left[\mathbf{I} - \mathbf{V} (\sigma_w^2 \mathbf{A}^{-1} + \mathbf{V}^H \mathbf{V})^{-1} \mathbf{V}^H \right] \quad (5)$$

For the case of infinite signal-to-noise ratio, the matrix $\mathbf{A}^{-1} = \mathbf{0}$. Therefore, the inverse of the correlation matrix in (5) becomes

$$\begin{aligned} \mathbf{R}_x^{-1} &= \frac{1}{\sigma_w^2} \left[\mathbf{I} - \mathbf{V} (\mathbf{V}^H \mathbf{V})^{-1} \mathbf{V}^H \right] \\ &= \frac{1}{\sigma_w^2} (\mathbf{I} - \mathbf{Q}_s \mathbf{Q}_s^H) \\ &= \frac{1}{\sigma_w^2} \mathbf{Q}_w \mathbf{Q}_w^H \\ &= \frac{1}{\sigma_w^2} \mathbf{P}_w \end{aligned}$$

making use of the projection matrix relation from (9.6.16) for the subspace spanned by the signal frequency vectors in \mathbf{V} . Therefore, for the case of infinite signal-to-noise ratio, the inverse of the correlation matrix becomes the noise subspace projection matrix from (3) normalized by the noise power σ_w^2 . Using this result in the equations for the minimum-variance spectrum in (1)

$$\begin{aligned} \hat{R}_M^{(mv)} (e^{j2\pi f}) &= \frac{M}{\mathbf{v}^H(f) \mathbf{R}_x^{-1} \mathbf{v}(f)} \\ &= \frac{M\sigma_w^2}{\mathbf{v}^H(f) \mathbf{P}_w \mathbf{v}(f)} \\ &= M\sigma_w^2 \bar{R}_{music} (e^{j2\pi f}) \end{aligned} \quad (6)$$

which is the MUSIC pseudo-spectrum from (2) weighted by $M\sigma_w^2$. Thus, the MUSIC pseudo-spectrum can be viewed as a minimum-variance spectrum with infinite signal-to-noise ratio. Since the resolution of the complex exponentials is related to their signal-to-noise ratios, the MUSIC pseudo-spectrum exhibits superior resolution capability over the minimum-variance spectrum and has led to the MUSIC frequency estimation method commonly being referred to as a *superresolution* technique.

9.28 If it has not been done so in your edition, this problem should be restated as:

Find the relationship between the minimum-norm pseudo-spectrum and the all-pole model spectrum in the case of infinite signal-to-noise ratio. What are the implications of this relationship?

First, let us examine the M th order all-pole model spectrum

$$\hat{R}_M^{(ap)} (e^{j2\pi f}) = \frac{P_M}{|\mathbf{v}^H(f) \mathbf{a}_M|} \quad (1)$$

where

$$\mathbf{a}_M = [1 \ a_1^{(M)} \ a_2^{(M)} \ \dots \ a_{M-1}^{(M)}]$$

is the vector of the M th order all-pole model coefficients which is found by solving the following set of linear equations

$$\mathbf{R}_x \mathbf{a}_M = \begin{bmatrix} P_M \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Therefore, we can also write \mathbf{a}_M as

$$\mathbf{a}_M = P_M \mathbf{R}_x^{-1} \boldsymbol{\delta}_1$$

where $\boldsymbol{\delta}_1 = [1 \ 0 \ \cdots \ 0]^T$, i.e., the column vector with a value of unity in its first element and otherwise all zeros. Since

$$\mathbf{a}_M(1) = \boldsymbol{\delta}_1^H \mathbf{a}_M = \boldsymbol{\delta}_1^H \mathbf{R}_x^{-1} \boldsymbol{\delta}_1 P_M = 1$$

then

$$P_M = \frac{1}{\boldsymbol{\delta}_1^H \mathbf{R}_x^{-1} \boldsymbol{\delta}_1} \quad (2)$$

and

$$\mathbf{a}_M = \frac{\mathbf{R}_x^{-1} \boldsymbol{\delta}_1}{\boldsymbol{\delta}_1^H \mathbf{R}_x^{-1} \boldsymbol{\delta}_1} \quad (3)$$

Substituting (2) and (3) into the all-pole model spectrum in (1), we have

$$\begin{aligned} \hat{R}_M^{(ap)}(e^{j2\pi f}) &= \frac{P_M}{|\mathbf{v}^H(f) \mathbf{a}_M|} \\ &= \frac{\boldsymbol{\delta}_1^H \mathbf{R}_x^{-1} \boldsymbol{\delta}_1}{|\mathbf{v}^H(f) \mathbf{R}_x^{-1} \boldsymbol{\delta}_1|^2} \end{aligned} \quad (4)$$

>From (9.6.7), the correlation matrix of P complex exponentials in white noise is

$$\mathbf{R}_x = \mathbf{V} \mathbf{A} \mathbf{V}^H + \sigma_w^2 \mathbf{I} \quad (5)$$

where \mathbf{V} is the matrix of P frequency time-window vectors

$$\mathbf{V} = [\mathbf{v}(f_1) \ \mathbf{v}(f_2) \ \cdots \ \mathbf{v}(f_P)]$$

and \mathbf{A} is the matrix of signal powers

$$\mathbf{A} = \text{diag}\{|\alpha_1|^2, |\alpha_2|^2, \dots, |\alpha_P|^2\}$$

The correlation matrix can also be written in terms of the signal and noise eigenvector and eigenvalues from (9.6.14)

$$\mathbf{R}_x = \mathbf{Q}_s \Lambda_s \mathbf{Q}_s^H + \sigma_w^2 \mathbf{Q}_w \mathbf{Q}_w^H$$

where the matrices \mathbf{Q}_s and \mathbf{Q}_w are made up of the signal and noise eigenvectors respectively

$$\mathbf{Q}_s = [\mathbf{q}_1 \ \mathbf{q}_2 \ \cdots \ \mathbf{q}_P] \quad \mathbf{Q}_w = [\mathbf{q}_{P+1} \ \mathbf{q}_{P+2} \ \cdots \ \mathbf{q}_M]$$

and

$$\Lambda_s = \text{diag} \{ \lambda_1, \dots, \lambda_P \}$$

Also recall the signal and noise subspace projection matrices from (9.6.17)

$$\mathbf{P}_s = \mathbf{Q}_s \mathbf{Q}_s^H \quad \mathbf{P}_w = \mathbf{Q}_w \mathbf{Q}_w^H$$

where from (9.6.16)

$$\mathbf{P}_s = \mathbf{Q}_s \mathbf{Q}_s^H = \mathbf{V} (\mathbf{V}^H \mathbf{V})^{-1} \mathbf{V}^H$$

Note also that the noise subspace projection matrix is given by

$$\mathbf{P}_w = \mathbf{I} - \mathbf{P}_s = \mathbf{I} - \mathbf{V} (\mathbf{V}^H \mathbf{V})^{-1} \mathbf{V}^H$$

Using the matrix inversion lemma from Appendix A on the correlation matrix from (5)

$$\mathbf{R}_x^{-1} = \frac{1}{\sigma_w^2} \left[\mathbf{I} - \mathbf{V} (\sigma_w^2 \mathbf{A}^{-1} + \mathbf{V}^H \mathbf{V})^{-1} \mathbf{V}^H \right] \quad (6)$$

In the case of infinite signal-to-noise ratio, $\mathbf{A}^{-1} = \mathbf{0}$ and (6) becomes

$$\mathbf{R}_x^{-1} = \frac{1}{\sigma_w^2} \left[\mathbf{I} - \mathbf{V} (\mathbf{V}^H \mathbf{V})^{-1} \mathbf{V}^H \right] = \frac{1}{\sigma_w^2} \mathbf{P}_w \quad (7)$$

Substituting (7) into the all-pole model spectrum from (4)

$$\begin{aligned} \hat{R}_M^{(ap)} (e^{j2\pi f}) &= \frac{\frac{1}{\sigma_w^2} \delta_1^H \mathbf{P}_w \delta_1}{\frac{1}{\sigma_w^4} |\mathbf{v}^H(f) \mathbf{P}_w \delta_1|^2} \\ &= \frac{\sigma_w^2 \mathbf{P}_w (1, 1)}{|\mathbf{v}^H(f) \mathbf{P}_w \delta_1|^2} \end{aligned} \quad (8)$$

The minimum-norm pseudo-spectrum from (9.6.41) is given by

$$\bar{R}_{mn} (e^{j2\pi f}) = \frac{1}{|\mathbf{v}^H(f) \mathbf{u}_{mn}|^2} \quad (9)$$

where the minimum-norm vector is given by

$$\mathbf{u}_{mn} = \frac{\mathbf{P}_w \delta_1}{\delta_1^H \mathbf{P}_w \delta_1} \quad (10)$$

Substituting (10) into (9), the minimum-norm pseudo-spectrum is

$$\bar{R}_{mn} (e^{j2\pi f}) = \frac{(\delta_1^H \mathbf{P}_w \delta_1)^2}{|\mathbf{v}^H(f) \mathbf{P}_w \delta_1|^2} = \frac{\mathbf{P}_w (1, 1)^2}{|\mathbf{v}^H(f) \mathbf{P}_w \delta_1|^2} \quad (11)$$

Comparing (8) and (11), we see that

$$\bar{R}_{mn} (e^{j2\pi f}) = \frac{\mathbf{P}_w (1, 1)}{\sigma_w^2} \hat{R}_M^{(ap)} (e^{j2\pi f})$$

Therefore, the minimum-norm pseudo-spectrum becomes a multiple of the all-pole model spectrum with infinite signal-to-noise ratio for all of the complex exponentials. This relation explains the superior resolution that the minimum-norm method can achieve over the all-pole model spectrum.

9.29 Recall both the MUSIC and minimum-norm pseudo-spectra from (9.6.30) and (9.6.41)

$$\bar{R}_{music}(e^{j2\pi f}) = \frac{1}{\sum_{m=P+1}^M |\mathbf{v}^H(f) \mathbf{q}_m|^2}$$

$$\bar{R}_{mn}(e^{j2\pi f}) = \frac{1}{|\mathbf{v}^H(f) \mathbf{u}_{mn}|^2}$$

where the minimum-norm vector is

$$\mathbf{u}_{mn} = \frac{\mathbf{P}_w \boldsymbol{\delta}_1}{\boldsymbol{\delta}_1^H \mathbf{P}_w \boldsymbol{\delta}_1}$$

Substituting for the minimum-norm vector into the minimum-norm pseudo-spectrum

$$\bar{R}_{mn}(e^{j2\pi f}) = \frac{\mathbf{P}_w(1, 1)^2}{|\mathbf{v}^H(f) \mathbf{P}_w \boldsymbol{\delta}_1|^2}$$

Looking at the reciprocal of the minimum-norm pseudo-spectra

$$\begin{aligned} \frac{1}{\bar{R}_{mn}(e^{j2\pi f})} &= |\mathbf{v}^H(f) \mathbf{P}_w \boldsymbol{\delta}_1|^2 \\ &= \sum_{m=P+1}^M |\mathbf{v}^H(f) \mathbf{q}_m \mathbf{q}_m^H \boldsymbol{\delta}_1|^2 \\ &= \sum_{m=P+1}^M |\mathbf{q}_m(1) \mathbf{v}^H(f) \mathbf{q}_m|^2 \end{aligned} \quad (1)$$

where $\mathbf{q}_m(1)$ is simply the first element of the m th eigenvector \mathbf{q}_m . Therefore, the minimum-norm pseudo-spectrum has the same form as the MUSIC pseudo-spectrum where the contribution from each eigenvector simply has an alternate weighting, namely the first element of the corresponding eigenvector.

Adaptive Filters

10.1 Consider the process $x(n)$ generated using the AR(3) model

$$x(n) = -0.729x(n-3) + w(n)$$

where $w(n) \sim \text{WGN}(0, 1)$. We want to design a linear predictor of $x(n)$ using the SDA algorithm. Let

$$\hat{y}(n) = \hat{x}(n) = c_{o,1}x(n-1) + c_{o,2}x(n-2) + c_{o,3}x(n-3)$$

(a) Determine the 3×3 autocorrelation matrix \mathbf{R} of $x(n)$, and compute its eigenvalues $\{\lambda_i\}_{i=1}^3$.

The AR(3) model system function is given by

$$H(z) = \frac{1}{1 + 0.729z^{-3}}$$

Hence the autocorrelation sequence of the process $x(n)$ is

$$r_x(l) = \mathcal{Z}^{-1}[H(z)H(z^{-1})] = \mathcal{Z}^{-1}\left[\left(\frac{1}{1 + 0.729z^{-3}}\right)\left(\frac{1}{1 + 0.729z^3}\right)\right]$$

The partial fraction expansion of $\left(\frac{1}{1 + 0.729z^{-3}}\right)\left(\frac{1}{1 + 0.729z^3}\right)$ is given by

$$\begin{aligned} \left(\frac{1}{1 + 0.729z^{-3}}\right)\left(\frac{1}{1 + 0.729z^3}\right) &= \frac{1}{0.729z^{-3} + 1.5314 + 0.729z^3} \\ &= \frac{0.7114}{1 + 0.9z^{-1}} - \frac{0.7114}{1 + \frac{10}{9}z^{-1}} \\ &\quad + 1.4228 \frac{1 - 0.45z^{-1}}{1 - 0.9z^{-1} + 0.81z^{-2}} \\ &\quad - 1.4228 \frac{1 - (5/9)z^{-1}}{1 - \frac{10}{9}z^{-1} + (\frac{10}{9})^2 z^{-2}} \end{aligned}$$

Thus after inverse z -transformation

$$r_x(l) = 0.7114(-0.9)^{|l|} + 1.4228(0.9)^{|l|} \cos\left(\frac{\pi l}{3}\right) \quad ((1))$$

Using (1), the 3×3 autocorrelation matrix \mathbf{R} is given by

$$\mathbf{R} = \begin{bmatrix} r_x(0) & r_x(1) & r_x(2) \\ r_x(1) & r_x(0) & r_x(1) \\ r_x(2) & r_x(1) & r_x(0) \end{bmatrix} = \begin{bmatrix} 2.1342 & 0 & 0 \\ 0 & 2.1342 & 0 \\ 0 & 0 & 2.1342 \end{bmatrix}$$

The eigenvalues of \mathbf{R} are given by

$$\lambda_1 = 2.1342, \quad \lambda_2 = 2.1342, \quad \text{and} \quad \lambda_3 = 2.1342$$

- (b) Determine the 3×1 cross-correlation vector \mathbf{d} .

For an AR(3) process, the cross-correlation vector \mathbf{d} is given by

$$\mathbf{d} = [r_x(1) \quad r_x(2) \quad r_x(3)]^T = [1.2808 \quad 0.0000405 \quad -0.51868]^T$$

Clearly

$$\begin{bmatrix} 2.1342 & 0 & 0 \\ 0 & 2.1342 & 0 \\ 0 & 0 & 2.1342 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ -1.5558 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0.729 \end{bmatrix}$$

as expected.

- (c) Choose the step size μ so that the resulting response is overdamped. Now implement the SDA

$$\mathbf{c}_k = [c_{k,1} \ c_{k,2} \ c_{k,3}]^T = \mathbf{c}_{k-1} + 2\mu(\mathbf{d} - \mathbf{R}\mathbf{c}_{k-1})$$

and plot the trajectories of $\{c_{k,i}\}_{i=1}^3$ as a function of k .

Since $\lambda_{\max} = \lambda_{\min} = \lambda = 2.1342$, $\chi(\mathbf{R}) = 1$ and the step-size $\mu < \frac{1}{\lambda_{\max}} = 0.46856$. We choose $\mu = 0.1$ for the overdamped case. The Matlab script is shown below and the plot is shown in Figure 10.1c.

```
% AR(3) model parameters
bx = 1; ax = [1,0,0,0.729];

% (a) 3x3 Autocorrelation matrix of x(n) and its eigenvalues
brx = [0,0,0,1]; % Num of Rx(z)
arx = conv(ax,fliplr(ax)); % Den of Rx(z)
[Res,pole,K] = residuez(brx,arx); % PFE of Rx(z)

% Computation of correlation values
l = 0:3;
rx = Res(4)*pole(4).^l + Res(5)*pole(5).^l + Res(6)*pole(6).^l;
rx = real(rx);
R = toeplitz(rx(1:3));

% Computation of eigenvalues
lambda = eig(R); lambda = lambda';

% (b) 3x1 Crosscorrelation vector d
d = rx(2:4)';

% (c) Implementation of SDA: Overdamped case
mu = 0.1; % Step-size
N = 30; % Number of iterations
c = zeros(3,1,N);
c(:,1,1) = 2*mu*d; % coefficient vector at k = 1
for k = 2:N
    c(:,1,k) = c(:,1,k-1) + 2*mu*(d-R*c(:,1,k-1));
end
k = 0:N;
c1 = squeeze(c(1,1,:)); c1 = [0;c1];
```

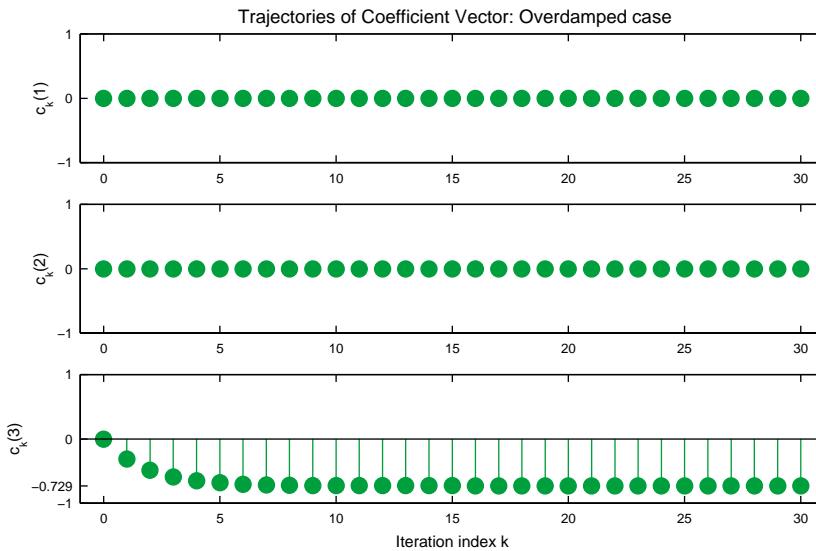


Figure 10.1c: Plots of trajectories for the overdamped case

```

c2 = squeeze(c(2,1,:)); c2 = [0;c2];
c3 = squeeze(c(3,1,:)); c3 = [0;c3];

Hf_1 = figure('units',SCRUN,'position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',[0,0,5,3]);
subplot(3,1,1); stem(k,c1,'g','filled'); axis([-1,N+1,-1,1]);
set(gca,'ytick',[-1:1:1]);
ylabel('c_k(1)', 'fontsize',label_fontsize);
title('Trajectories of Coefficient Vector: Overdamped case',...
    'fontsize',title_fontsize);
subplot(3,1,2); stem(k,c2,'g','filled'); axis([-1,N+1,-1,1]);
set(gca,'ytick',[-1:1:1]);
ylabel('c_k(2)', 'fontsize',label_fontsize);
subplot(3,1,3); stem(k,c3,'g','filled'); axis([-1,N+1,-1,1]);
hold on; plot([-1,N+1],[0,0],'w');
set(gca,'ytick',[-1,-0.729,0,1]);
xlabel('Iteration index k', 'fontsize',label_fontsize);
ylabel('c_k(3)', 'fontsize',label_fontsize);

```

- (d) Repeat part (c) by choosing μ so that the response is underdamped.

We choose $\mu = 0.4$ for the underdamped case. The Matlab script is shown below and the plot is shown in Figure 10.1d.

```

% (d) Implementation of SDA: Overdamped case
mu = 0.4; % Step-size
N = 30; % Number of iterations
c = zeros(3,1,N);
c(:,1,1) = 2*mu*d; % coefficient vector at k = 1
for k = 2:N
    c(:,1,k) = c(:,1,k-1) + 2*mu*(d-R*c(:,1,k-1));
end

```

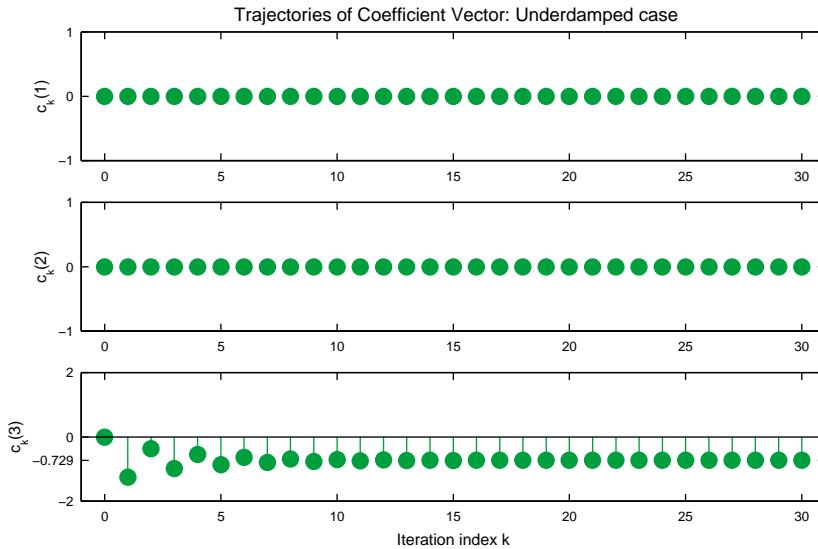


Figure 10.1c: Plots of trajectories for the underdamped case

```

k = 0:N;
c1 = squeeze(c(1,1,:)); c1 = [0;c1];
c2 = squeeze(c(2,1,:)); c2 = [0;c2];
c3 = squeeze(c(3,1,:)); c3 = [0;c3];

Hf_2 = figure('units','SCRUN','position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',[0,0,5,3]);
subplot(3,1,1); stem(k,c1,'g','filled'); axis([-1,N+1,-1,1]);
set(gca,'ytick',[-1:1:1]);
ylabel('c_k(1)', 'fontsize',label_fontsize);
title('Trajectories of Coefficient Vector: Underdamped case',...
    'fontsize',title_fontsize);
subplot(3,1,2); stem(k,c2,'g','filled'); axis([-1,N+1,-1,1]);
set(gca,'ytick',[-1:1:1]);
ylabel('c_k(2)', 'fontsize',label_fontsize);
subplot(3,1,3); stem(k,c3,'g','filled'); axis([-1,N+1,-2,2]);
hold on; plot([-1,N+1],[0,0],'w');
set(gca,'ytick',[-2,-0.729,0,2]);
xlabel('Iteration index k', 'fontsize',label_fontsize);
ylabel('c_k(3)', 'fontsize',label_fontsize);

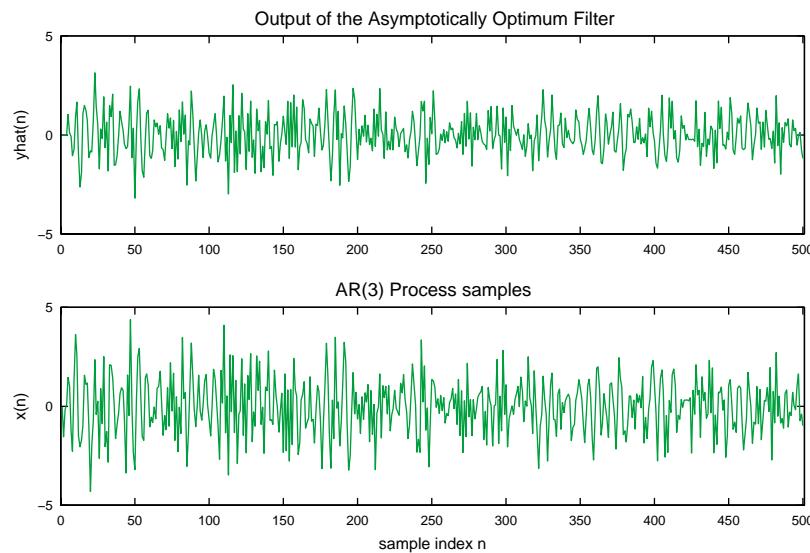
```

- 10.2** In the SDA algorithm, the index k is an iteration index and not a time index. However, we can treat it as a time index and use the instantaneous filter coefficient vector \mathbf{c}_k to filter data at $n = k$. This will result in an *asymptotically optimum* filter whose coefficients will converge to the optimum one. Consider the process $x(n)$ given in Problem 10.1.

- (a) Generate 500 samples of $x(n)$ and implement the asymptotically optimum filter. Plot the signal $\hat{y}(n)$.

The design and implementation is shown in the following Matlab script and the plot is shown in Figure 10.2a.

```
% Generate 500 samples of AR(3) process
```

**Figure 10.2a:** Asymptotically optimum filter response

```

bx = 1; ax = [1,0,0,0.729]; var_w = 1; % Model parameters
N = 500; w = randn(N,1)*sqrt(var_w); % white noise process
x = filter(bx,ax,w); % AR(3) process

% (a) Design and processing using AO filter
% Computation of correlation values
brx = [0,0,0,1]; % Num of Rx(z)
arx = conv(ax,fliplr(ax)); % Den of Rx(z)
[Res,pole,K] = residuez(brx,arx); % PFE of Rx(z)
l = 0:3;
rx = Res(4)*pole(4).^l + Res(5)*pole(5).^l + Res(6)*pole(6).^l;
R = toeplitz(rx(1:3)); % 3x3 Autocorrelation matrix
d = rx(2:4)'; % 3x1 Crosscorrelation vector d
% Design of the AO filter sequence
mu = 0.1; % Step-size
c = zeros(3,1,N);
c(:,1,1) = 2*mu*d; % coefficient vector at k = 1
for k = 2:N
    c(:,1,k) = c(:,1,k-1) + 2*mu*(d-R*c(:,1,k-1));
end
n = 1:N;
c1 = squeeze(c(1,1,:));
c2 = squeeze(c(2,1,:));
c3 = squeeze(c(3,1,:));
% Processing using the AO filter sequence
yhat = zeros(N,1);
for k = 4:N
    yhat(k) = c1(k)*x(k-1) + c2(k)*x(k-2) + c3(k)*x(k-3);
end

Hf_1 = figure('units','SCRUN','position',SCRPOS,...

```

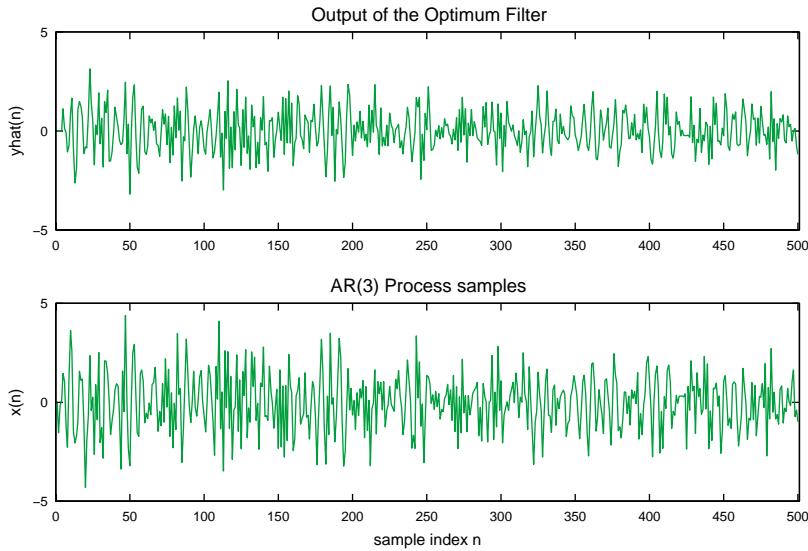


Figure 10.2b: Optimum filter response

```
'paperunits',PAPUN,'paperposition',[0,0,5,3]);
subplot(2,1,1);
plot(n,yhat,'g'); axis([0,N+1,-5,5]);
set(gca,'ytick',[-5:5:5]);
ylabel('yhat(n)', 'fontsize',label_fontsize);
title('Output of the Asymptotically Optimum Filter',...
      'fontsize',title_fontsize);
subplot(2,1,2);
plot(n,x,'g'); axis([0,N+1,-5,5]);
set(gca,'ytick',[-5:5:5]);
ylabel('x(n)', 'fontsize',label_fontsize);
xlabel('sample index n', 'fontsize',label_fontsize);
title('AR(3) Process samples','fontsize',title_fontsize);
```

- (b) Implement the optimum filter c_o on the same sequence, and plot the resulting $\hat{y}(n)$.

The design and implementation is shown in the following Matlab script and the plot is shown in Figure 10.2b.

```
% (b) Design and processing using Optimum filter
c_o = R\d;
yhat = filter([0;c_o],1,x);

Hf_2 = figure('units',SCRUN,'position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',[0,0,5,3]);
subplot(2,1,1);
plot(n,yhat,'g'); axis([0,N+1,-5,5]);
set(gca,'ytick',[-5:5:5]);
ylabel('yhat(n)', 'fontsize',label_fontsize);
title('Output of the Optimum Filter',...
      'fontsize',title_fontsize);
subplot(2,1,2);
```

```

plot(n,x,'g');
axis([0,N+1,-5,5]);
set(gca,'ytick',[-5:5:5]);
ylabel('x(n)', 'fontsize',label_fontsize);
xlabel('sample index n', 'fontsize',label_fontsize);
title('AR(3) Process samples', 'fontsize',title_fontsize);

```

- (c) Comment on the above two plots.

The plots are identical after the initial transients.

10.3 Consider the AR(2) process $x(n)$ given in Example 10.3.1. We want to implement the Newton-type algorithm for faster convergence using

$$\mathbf{c}_k = \mathbf{c}_{k-1} - \mu \mathbf{R}^{-1} \nabla P(\mathbf{c}_{k-1})$$

- (a) Using $a_1 = -1.5955$ and $a_2 = 0.95$, implement the above method for $\mu = 0.1$ and $\mathbf{c}_0 = \mathbf{0}$. Plot the locus of $c_{k,1}$ versus $c_{k,2}$.

From (10.3.29) and (10.3.35) and choosing σ_w^2 so that $\sigma_x^2 = 1$, we have

$$r(0) = 1, r(1) = \frac{-a_1}{1+a_2} r(0) = 0.81821, \text{ and } r(2) = \left(\frac{a_1^2}{1+a_2} - a_2 \right) r(0) = 0.35545$$

Hence

$$\mathbf{R} = \begin{bmatrix} 1 & 0.81821 \\ 0.81821 & 1 \end{bmatrix}, \quad d = \begin{bmatrix} 0.81821 \\ 0.35545 \end{bmatrix}$$

The Newton-type algorithm is implemented in the Matlab script below and the plot is shown in Figure 10.3.

```

% Initialization
c0 = zeros(2,1);

% (a) a1 = -1.5955, a2 = 0.95, and step-size mu = 0.1
a1 = -1.5955; a2 = 0.95; mu = 0.1;
r0 = 1; % var_w selected so that var_x = 1
r1 = -(a1/(1+a2))*r0;
r2 = (-a2+a1^2/(1+a2))*r0;
R = toeplitz([r0,r1]); Rinv = R\eye(2);
d = [r1;r2];
c = zeros(2,100);
delP = -d;
c(:,1) = -mu*Rinv*delP;
for k = 2:100
    delP = R*c(:,k-1) - d;
    c(:,k) = c(:,k-1) - mu*Rinv*delP;
end
c = [c0,c];

subplot(2,2,1);
plot(c(1,:),c(2,:),'md',c(1,:),c(2,:),'g-'); axis([0,-2*a1,-2*a2,0]);
%xlabel('c_{k,1}', 'fontsize',label_fontsize);
%ylabel('c_{k,2}', 'fontsize',label_fontsize);

```

```

title('Trajectory of c_{k,1} vs c_{k,2}, \mu = 0.1',...
    'fontsize',title_fontsize);
set(gca,'xtick',[0,-a1],'ytick',[-a2,0]);grid;
text(-a1+0.05,-a2+0.05,'c_o');
text(-a1+0.3,-a2+0.7,'a_1 = -1.5955');
text(-a1+0.3,-a2+0.5,'a_2 = 0.95');

```

- (b) Repeat part (a), using $a_1 = -0.195$ and $a_2 = 0.95$.

In this case

$$r(0) = 1, r(1) = \frac{-a_1}{1+a_2}r(0) = 0.1, \text{ and } r(2) = \left(\frac{a_1^2}{1+a_2} - a_2 \right) r(0) = -0.9305$$

and

$$\mathbf{R} = \begin{bmatrix} 1 & 0.1 \\ 0.1 & 1 \end{bmatrix}, \quad \mathbf{d} = \begin{bmatrix} 0.1 \\ -0.9305 \end{bmatrix}$$

The Newton-type algorithm is implemented in the Matlab script below and the plot is shown in Figure 10.3.

```

% (b) a1 = -0.195, a2 = 0.95, and step-size mu = 0.1
a1 = -0.195; a2 = 0.95; mu = 0.1;
r0 = 1; % var_w selected so that var_x = 1
r1 = -(a1/(1+a2))*r0-eps,
r2 = (-a2+a1^2/(1+a2))*r0-eps,
R = toeplitz([r0,r1]); Rinv = R\eye(2);
d = [r1;r2];
c = zeros(2,100);
delP = -d;
c(:,1) = -mu*Rinv*delP;
for k = 2:100
    delP = R*c(:,k-1) - d;
    c(:,k) = c(:,k-1) - mu*Rinv*delP;
end
c = [c0,c];

subplot(2,2,2);
plot(c(1,:),c(2,:),'md',c(1,:),c(2,:),'g-'); axis([0,-2*a1,-2*a2,0]);
% xlabel('c_{k,1}', 'fontsize', label_fontsize);
ylabel('c_{k,2}', 'fontsize', label_fontsize);
title('Trajectory of c_{k,1} vs c_{k,2}, \mu = 0.1',...
    'fontsize', title_fontsize);
set(gca,'xtick',[0,-a1],'ytick',[-a2,0]);grid;
text(-a1+0.005,-a2+0.05,'c_o');
text(-a1+0.03,-a2+0.7,'a_1 = -0.195');
text(-a1+0.03,-a2+0.5,'a_2 = 0.95');

```

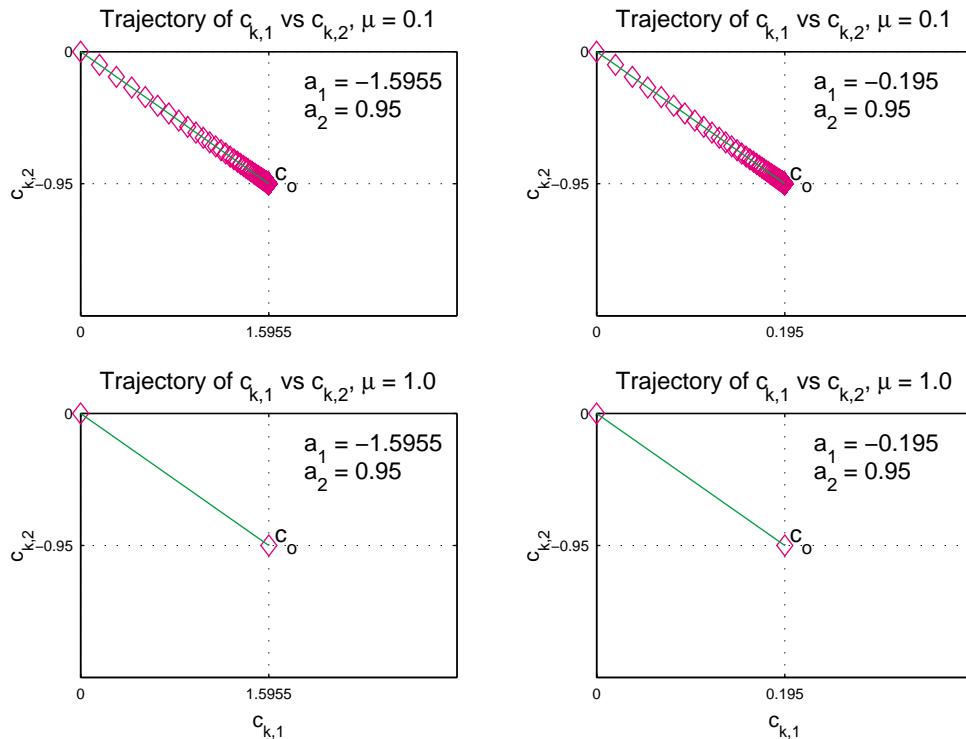
- (c) Repeat parts (a) and (b), using the optimum step size for μ that results in the fastest convergence.

The optimum step-size is $\mu = 1$. The Matlab script is shown below and the plot is shown in Figure 10.3.

```

% (c1) a1 = -1.5955, a2 = 0.95, and step-size mu = 1
a1 = -1.5955; a2 = 0.95; mu = 1;

```

**Figure 10.3:** Plots of trajectories in Problem 10.3

```

r0 = 1; % var_w selected so that var_x = 1
r1 = -(a1/(1+a2))*r0;
r2 = (-a2+a1^2/(1+a2))*r0;
R = toeplitz([r0,r1]); Rinv = R\eye(2);
d = [r1;r2];
c = zeros(2,100);
delP = -d;
c(:,1) = -mu*Rinv*delP;
for k = 2:100
    delP = R*c(:,k-1) - d;
    c(:,k) = c(:,k-1) - mu*Rinv*delP;
end
c = [c0,c];

subplot(2,2,3);
plot(c(1,:),c(2,:),'md',c(1,:),c(2,:),'g-'); axis([0,-2*a1,-2*a2,0]);
xlabel('c_{k,1}', 'fontsize', label_fontsize);
ylabel('c_{k,2}', 'fontsize', label_fontsize);
title('Trajectory of c_{k,1} vs c_{k,2}, \mu = 1.0',...
    'fontsize', title_fontsize);
set(gca,'xtick',[0,-a1], 'ytick',[-a2,0]); grid;
text(-a1+0.05,-a2+0.05,'c_o');
text(-a1+0.3,-a2+0.7,'a_1 = -1.5955');
text(-a1+0.3,-a2+0.5,'a_2 = 0.95');

```

```
% (c2) a1 = -0.195, a2 = 0.95, and step-size mu = 1
a1 = -0.195; a2 = 0.95; mu = 1;
r0 = 1; % var_w selected so that var_x = 1
r1 = -(a1/(1+a2))*r0;
r2 = (-a2+a1^2/(1+a2))*r0;
R = toeplitz([r0,r1]); Rinv = R\eye(2);
d = [r1;r2];
c = zeros(2,100);
delP = -d;
c(:,1) = -mu*Rinv*delP;
for k = 2:100
    delP = R*c(:,k-1) - d;
    c(:,k) = c(:,k-1) - mu*Rinv*delP;
end
c = [c0,c];

subplot(2,2,4);
plot(c(1,:),c(2,:),'md',c(1,:),c(2,:),'g-'); axis([0,-2*a1,-2*a2,0]);
xlabel('c_{k,1}', 'fontsize', label_fontsize);
ylabel('c_{k,2}', 'fontsize', label_fontsize);
title('Trajectory of c_{k,1} vs c_{k,2}, \mu = 1.0', ...
    'fontsize', title_fontsize);
set(gca, 'xtick', [0, -a1], 'ytick', [-a2, 0]); grid;
text(-a1+0.005, -a2+0.05, 'c_o');
text(-a1+.03, -a2+.7, 'a_1 = -0.195');
text(-a1+.03, -a2+.5, 'a_2 = 0.95');
```

10.4 Consider the adaptive linear prediction of an AR(2) process $x(n)$ using the LMS algorithm in which

$$x(n) = 0.95x(n-1) - 0.9x(n-2) + w(n)$$

where $w(n) \sim \text{WGN}(0, \sigma_w^2)$. The adaptive predictor is a second-order one given by $\mathbf{a}(n) = [a_1(n) \ a_2(n)]^T$.

- (a) Implement the LMS algorithm given in Table 10.3 as a Matlab function $[c, e] = \text{lplms}(x, y, \mu, M, c0)$ which computes filter coefficients in c and the corresponding error in e , given signal x , desired signal y , step size μ , filter order M , and the initial coefficient vector $c0$.

```
function [c,e] = lplms(x,y,mu,M,c0)
% Computes filter coefficients in c and the corresponding error in
% e, given signal x, desired signal y, step size mu, filter order M, and
% the initial coefficient vector c0.
%
% [c,e] = lplms(x,y,mu,M,c0)
%
N = length(x);           % Length of x(n)
x = reshape(x,N,1);      % x as a column vector
y = x(2:end);            % Generate y(n)
N = length(y);            % Number of iterations
c = zeros(M,N);           % Initialization of c
```

```

X = zeros(M,N); % Data matrix
xx = [zeros(M-1,1);x];
for i = 1:M
    X(i,:) = xx(M-i+1:end-i)';
end
e = zeros(1,N);

%--LMS Algorithm for Linear Prediction - real-valued case
% Algorithm initialization
yhat = c0'*X(:,1);
e(1) = y(1) - yhat;
c(:,1) = 2*mu*X(:,1)*e(1);;
% Iterations
for n = 2:N
    yhat = c(:,n-1)'*X(:,n);
    e(n) = y(n) - yhat;
    c(:,n) = c(:,n-1) + 2*mu*X(:,n)*e(n);
end
c = [c0,c];
e = [x(1),e];

```

- (b) Generate 500 samples of $x(n)$, and obtain linear predictor coefficients using the above function. Use step size μ so that the algorithm converges in the mean. Plot predictor coefficients as a function of time along with the true coefficients.

The Matlab script file is shown below and plots are shown in Figure 10.4 (a) and (b).

```

clc; close all
set(0,'defaultaxesfontsize',default_fontsize);
Lx = [-1,3]; % Maximum abs limit for the x-axis of contour plots
Ly = [-3,1]; % Maximum abs limit for the y-axis of contour plots
N = 501; % Signal size (also number of iterations)
Niter = 1000; % Ensemble size
mua = 0.04; % Large Step-size
num = 0.01; % Small Step-size

% Generate AR(2) process
a1 = -0.950; a2 = 0.9; varx = 1; r0 = varx;
%a1a = a1; a2a = a2;
varw = ((1-a2)*((1+a2)^2-a1^2)/(1+a2))*varx;
r1 = -a1/(1+a2)*r0; r2 = (-a2+a1^2/(1+a2))*r0;
lam1 = (1-a1/(1+a2))*varx; lam2 = (1+a1/(1+a2))*varx;
XR = lam1/lam2;d = [r1;r2]; R = toeplitz([r0,r1,r2]);
% Ensemble averaging
e2 = zeros(N,1); e = e2; c1m = e; c2m = e; c1 = e; c2 = e;
em2 = zeros(N,1); em = em2; cm1m = em; cm2m = em; cm1 = em; cm2 = em;
for i=1:Niter
    w = sqrt(varw)*randn(N,1);
    x = filter(1,[1 a1 a2],w);

    c1(1) = 0; c1(2) = 0; c2(1) = 0; c2(2) = 0;

```

```

cm1(1) = 0; cm1(2) = 0; cm2(1) = 0; cm2(2) = 0;
% LMS Algorithm
for n=3:N
    e(n) = x(n)-c1(n-1)*x(n-1)-c2(n-1)*x(n-2);
    c1(n) = c1(n-1)+2*mua*e(n)*x(n-1);
    c2(n) = c2(n-1)+2*mua*e(n)*x(n-2);
    em(n) = x(n)-cm1(n-1)*x(n-1)-cm2(n-1)*x(n-2);
    cm1(n) = cm1(n-1)+2*mum*em(n)*x(n-1);
    cm2(n) = cm2(n-1)+2*mum*em(n)*x(n-2);
end
c1m = c1m+c1; c2m = c2m+c2; e2 = e2+e.^2;
cm1m = cm1m+cm1; cm2m = cm2m+cm2; em2 = em2+em.^2;
end
c1ma = c1m/Niter; c2ma = c2m/Niter; e2a = e2/Niter;
cm1ma = cm1m/Niter; cm2ma = cm2m/Niter; em2a = em2/Niter;

% contour plots
a1p=(Lx(1):0.1:Lx(2))';
a2p=(Ly(1):0.1:Ly(2))';
L1=length(a1p);
L2=length(a2p);

for i=1:L1
    for j=1:L2
        a=[1 -a1p(i) -a2p(j)]';
        P(i,j)=a'*R*a;
    end
end

%Plots
Hf_1 = figure('units','SCRUN','position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',[0,0,6,4.8],...
    'NumberTitle','off','Name','Pr1004');

subplot('position',[0.075,0.55,0.4,0.4]);
contour(a1p,a2p,P',(1:2:20)/10); hold on;
plot(c1,c2,'r',c1ma,c2ma,'g.');
%plot(c1ma,c2ma,'r','linewidth',1.5);
%plot(Lx,[0,0],'w:',[0,0],Ly,'w:',Lx,[-a1,-a1],'w:',[-a2,-a2],[0,0],'w:');
axis([Lx,Ly]);axis('square');
title('(a) Averaged Trajectory','fontsize',10);
xlabel('c_{1}', 'fontsize',8); ylabel('c_{2}', 'fontsize',8);
set(gca,'xtick',[Lx(1),0,-a1,Lx(2)],'ytick',[Ly(1),-a2,0,Ly(2)],...
    'fontsize',6); grid;
hold off;

subplot('position',[0.575,0.55,0.4,0.4]); n = 0:N-1;
plot([-1,N],[-a2,-a2],'w:',[-1,N],[-a1,-a1],'w:'); hold on;
plot(n,c1,'c',n,c2,'g','linewidth',0.5);

```

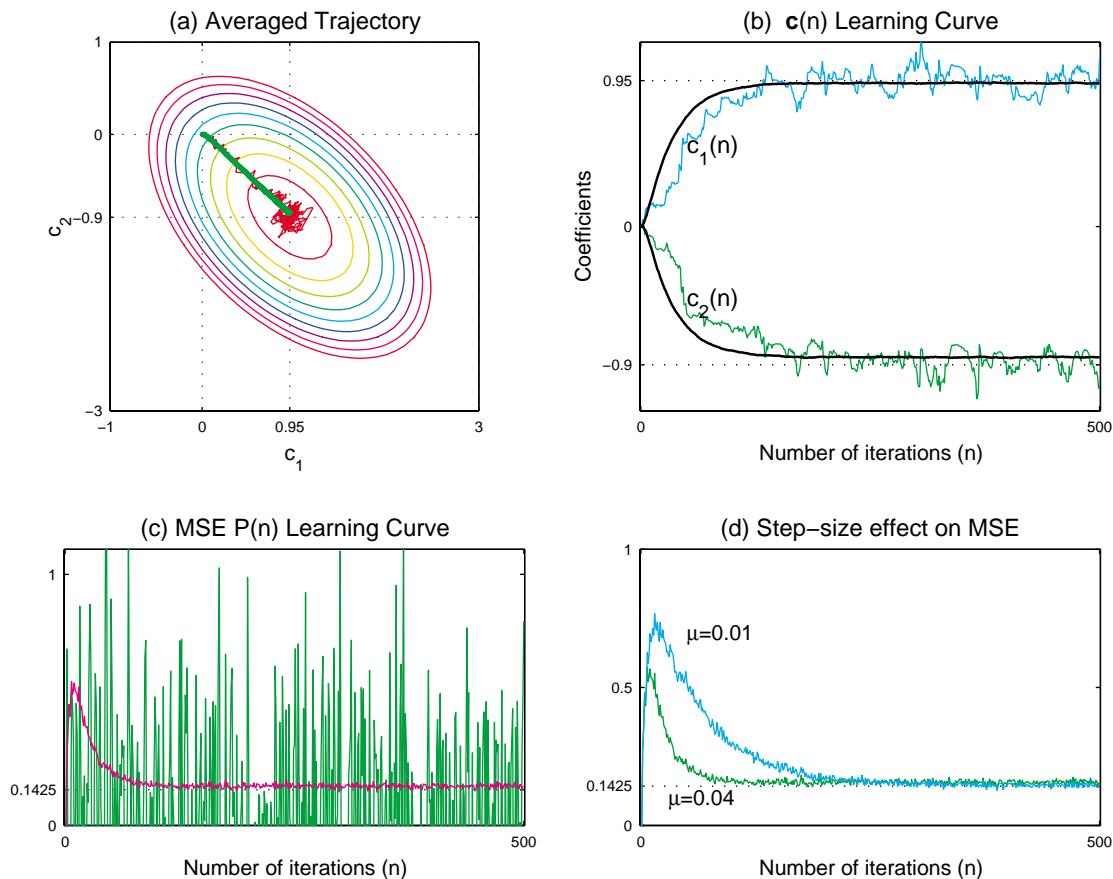


Figure 10.4: Plots of trajectories and learning curves in Problem 10.4

```

plot(n,c1ma,'w',n,c2ma,'w','linewidth',1);
axis([-1,N,-1.2,1.2]);
xlabel('Number of iterations (n)', 'fontsize',8);
ylabel('Coefficients', 'fontsize',8);
title(''(b) {\bf c}(n) Learning Curve', 'fontsize',10);
set(gca,'xtick',[0,N-1],'ytick',[-a2,0,-a1], 'fontsize',6);
text(50, 0.5,'c_1(n)', 'fontsize',10);
text(50,-0.5,'c_2(n)', 'fontsize',10);
hold off;

```

- (c) Repeat the above simulation 1000 times to obtain the learning curve, which is obtained by averaging the squared error $|e(n)|^2$. Plot this curve and compare its steady-state value with the theoretical MSE.

The Matlab script file is shown below and plots are shown in Figure 10.4 (c) and (d).

```

subplot('position',[0.075,0.1,0.4,0.3]);
plot([-1,N],[varw,varw],'w:',n,e,'g',n,e2a,'m'); axis([-1,N,0,1.1]);
xlabel('Number of iterations (n)', 'fontsize',8);
title(''(c) MSE P(n) Learning Curve', 'fontsize',10);
set(gca,'xtick',[0,N-1],'ytick',[0,varw,1], 'fontsize',6);

subplot('position',[0.575,0.1,0.4,0.3]);
plot([-1,N],[varw,varw],'w:',n,e2a,'g',n,em2a,'c'); axis([-1,N,0,1]);

```

```

xlabel('Number of iterations (n)', 'fontsize', 8);
title('(d) Step-size effect on MSE', 'fontsize', 10);
set(gca, 'xtick', [0, N-1], 'ytick', [0, varw, 0.5, 1], 'fontsize', 6);
text(30, 0.1, '\mu=0.04', 'fontsize', 8);
text(50, 0.7, '\mu=0.01', 'fontsize', 8);

```

10.5 Consider the adaptive echo canceler given in Figure 10.25. The FIR filter $c_o(n)$ is given by

$$c_o(n) = (0.9)^n \quad 0 \leq n \leq 2$$

In this simulation, ignore the far-end signal $u(n)$. The data signal $x(n)$ is a zero-mean, unit-variance white Gaussian process, and $y(n)$ is its echo.

We will first need the Matlab function `firlms` to implement the LMS algorithm which is given below.

```

function [c,e,yhat] = firlms(x,y,mu,M,c0)
% Computes filter coefficients in c and the corresponding error in
% e, given signal x, desired signal y, step size mu, filter order M, and
% the initial coefficient vector c0.
%
% [c,e] = firlms(x,y,mu,M,c0)
%

N = length(x);           % Length of x(n) & number of iterations
x = reshape(x,N,1);      % x as a column vector
c = zeros(M,N);          % Initialization of c
X = zeros(M,N);          % Data matrix
xx = [zeros(M-1,1);x]; % Prepend M-1 zeros
for i = 1:M
    X(i,:) = xx(M-i+1:end-i+1)';
end
e = zeros(1,N); yhat = zeros(N,1);

%--LMS Algorithm for FIR filter - real-valued case
% Algorithm initialization
yhat(1) = c0'*X(:,1);
e(1) = y(1) - yhat(1);
c(:,1) = c0 + 2*mu*X(:,1)*e(1);;
% Iterations
for n = 2:N
    yhat(n) = c(:,n-1)'*X(:,n);
    e(n) = y(n) - yhat(n);
    c(:,n) = c(:,n-1) + 2*mu*X(:,n)*e(n);
end

```

- (a) Generate 1000 samples of $x(n)$ and determine $y(n)$. Use these signals to obtain a fourth-order LMS echo canceler in which the step size μ is chosen to satisfy (10.4.40) and $\mathbf{c}(0) = \mathbf{0}$. Obtain the final echo canceler coefficients and compare them with the true ones.

The Matlab script is given below.

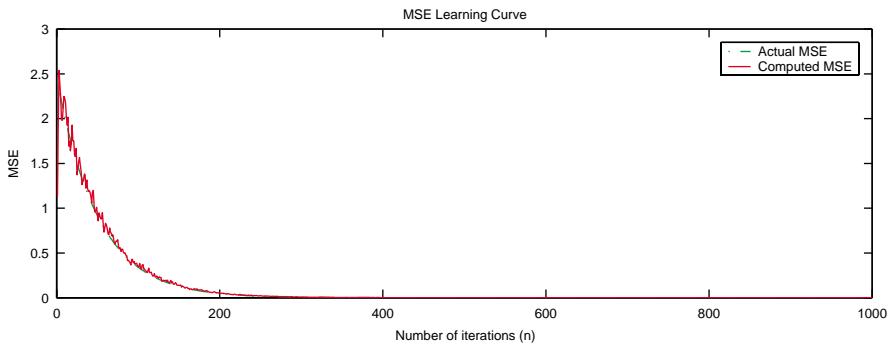


Figure 10.5: Plots of the MSE learning curve in Problem 10.5

```

clc; close all
set(0,'defaultaxesfontsize',default_fontsize);
co = (0.9).^ [0:2]; % Echo impulse response

% (a) Echo Canceller design
N = 1000; x = randn(N,1);
y = filter(co,1,x);
M = 4; % LMS filter order
mu = 0.005; c0 = zeros(M,1);
[c,e] = firnlms(x,y,mu,M,c0);

% Final Echo-canceller coefficients
cf = c(:,N);
*** Echo Canceller coefficients ***
1.0000    0.9000    0.8100    0.0000

*** Echo System coefficients ***
1.0000    0.9000    0.8100

```

- (b) Repeat the above simulation 500 times, and obtain the learning curve. Plot this curve along with the actual MSE and comment on the plot.

The Matlab script is given below and the learning curve is shown in Figure 10.5b.

```

% (b) Monte-Carlo analysis and the learning curve
Nsim = 500; N = 1000; M = 4; mu = 0.005; c0 = zeros(M,1);

eavg = zeros(1,N,Nsim);
cavg = zeros(M,N,Nsim);

for k = 1:Nsim
    x = randn(N,1); y = filter(co,1,x);
    [c,e] = firnlms(x,y,mu,M,c0);
    cavg(:,:,k) = c;
    eavg(:,:,k) = e;
end
cavg = mean(cavg,3);
eavg = eavg.^2; eavg = mean(eavg,3);

```

```
% Theoretical analysis
n = 1:N; alpha = (1-4*mu+4*mu*mu*M);
ethe = var_y*alpha.^n;

% Plots
Hf_1 = figure('units','SCRUN','position',SCRPOS,...,
    'paperunits',PAPUN,'paperposition',[0,0,5,2],...
    'NumberTitle','off','Name','Pr1005');
subplot('position',[0.1,0.15,0.85,0.7]);
plot(n,ethe,'g-.',n,eavg,'r');axis([0,N,0,3]);
xlabel('Number of iterations (n)', 'fontsize',label_fontsize);
title('MSE Learning Curve', 'fontsize',title_fontsize);
ylabel('MSE', 'fontsize',label_fontsize);
set(gca,'xtick',[0:200:N],'ytick',[0:0.5:3]);
legend('Actual MSE','Computed MSE',1);
```

- (c) Repeat parts (a) and (b), using a third-order echo canceler.

Modify above Matlab scripts to obtain the necessary results.

- (d) Repeat parts (a) and (b), using one-half the value of μ used in the first part.

Modify above Matlab scripts to obtain the necessary results.

10.6 The normalized LMS (NLMS) algorithm is given in (10.4.67), in which the effective step size is time-varying and is given by $\tilde{\mu}/\|\mathbf{x}(n)\|^2$, where $0 < \tilde{\mu} < 1$.

- (a) Modify the function `firlms` to implement the NLMS algorithm and obtain the function `[c, e] = nfirlms(x, y, mu, M, c0)`.

```
function [c,e] = nfirlms(x,y,mu,M,c0)
% Normalized LMS algorithm for FIR filter applications
% Computes filter coefficients in c and the corresponding error in
% e, given signal x, desired signal y, step size mu, filter order M, and
% the initial coefficient vector c0.
%
% [c,e] = nfirlms(x,y,mu,M,c0)
%

N = length(x); % Length of x(n) & number of iterations
x = reshape(x,N,1); % x as a column vector
c = zeros(M,N); % Initialization of c
X = zeros(M,N); % Data matrix
xx = [zeros(M-1,1);x]; % Prepend M-1 zeros
for i = 1:M
    X(i,:) = xx(M-i+1:end-i+1)';
end
e = zeros(1,N);

% Computation of EM(n)
```

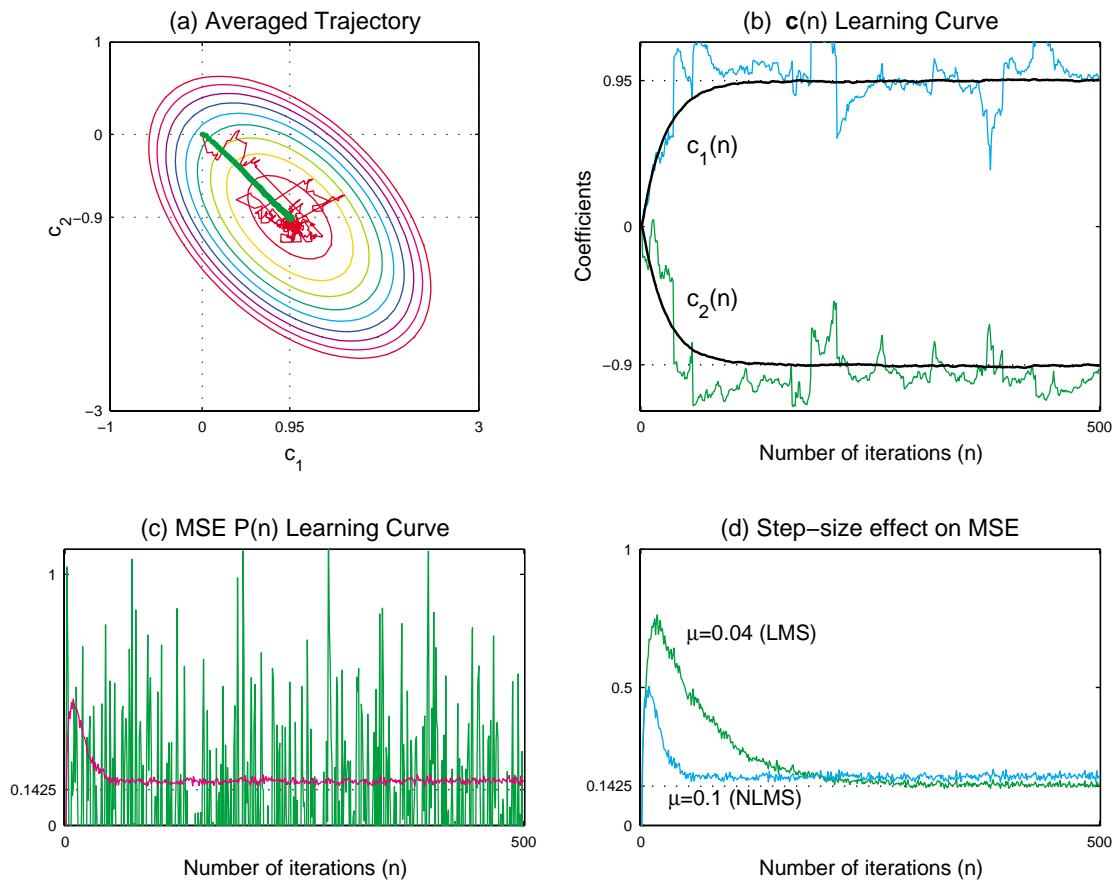


Figure 10.6b: Plots of the NLMS learning curve and its comparison with LMS in Problem 10.6b

```
w = abs(x).*abs(x);
b = [1,zeros(1,M-1),-1]; a = [1,-1];
EM = filter(b,a,w);

%--NLMS Algorithm for FIR filter - real-valued case
% Algorithm initialization
yhat = c0'*X(:,1);
e(1) = y(1) - yhat;
c(:,1) = c0 + mu*X(:,1)*e(1)/EM(1);;
% Iterations
for n = 2:N
    yhat = c(:,n-1)'*X(:,n);
    e(n) = y(n) - yhat;
    c(:,n) = c(:,n-1) + mu*X(:,n)*e(n)/EM(n);
end
e = e(M+1:end);
c = c(:,M+1:end);
```

- (b) Choose $\tilde{\mu} = 0.1$ and repeat Problem 10.4. Compare your results in terms of convergence speed.

The Matlab script is shown below and the plots in Figure 10.6b. From the plots we observe that the NLMS has the faster convergence.

```

% (b) NLMS for LP Applications
% AR(2) process:
%   x(n) + a1*x(n-1) + a2*x(n-2) = w(n)
%
% Linear Predictor
%   y(n) = x(n); yhat = c1*x(n-1) + c2*x(n-2);

clc; close all
set(0,'defaultaxesfontsize',default_fontsize);
Lx = [-1,3];      % Maximum abs limit for the x-axis of contour plots
Ly = [-3,1];      % Maximum abs limit for the y-axis of contour plots
N = 501;          % Signal size (also number of iterations)
Niter = 1000;     % Ensemble size
mua = 0.01;       % Step-size for LMS
mum = 0.1;        % Step-size for NLMS

% Generate AR(2) process
a1 = -0.950; a2 = 0.9; varx = 1; r0 = varx;
%a1a = a1; a2a = a2;
varw = ((1-a2)*((1+a2)^2-a1^2)/(1+a2))*varx;
r1 = -a1/(1+a2)*r0; r2 = (-a2+a1^2/(1+a2))*r0;
lam1 = (1-a1/(1+a2))*varx; lam2 = (1+a1/(1+a2))*varx;
XR = lam1/lam2;d = [r1;r2]; R = toeplitz([r0,r1,r2]);
% Ensemble averaging
e2 = zeros(N,1); e = e2; c1m = e; c2m = e; c1 = e; c2 = e;
em2 = zeros(N,1); em = em2; cm1m = em; cm2m = em; cm1 = em; cm2 = em;
for i=1:Niter
w = sqrt(varw)*randn(N,1);
x = filter(1,[1 a1 a2],w);
EM = filter([1,0,-1],[1,-1],abs(x).*abs(x))+0.001;

c1(1) = 0; c1(2) = 0; c2(1) = 0; c2(2) = 0;
cm1(1) = 0; cm1(2) = 0; cm2(1) = 0; cm2(2) = 0;
% LMS Algorithm
for n=3:N
e(n) = x(n)-c1(n-1)*x(n-1)-c2(n-1)*x(n-2);
c1(n) = c1(n-1)+2*mua*e(n)*x(n-1);
c2(n) = c2(n-1)+2*mua*e(n)*x(n-2);
em(n) = x(n)-cm1(n-1)*x(n-1)-cm2(n-1)*x(n-2);
cm1(n) = cm1(n-1)+mum*em(n)*x(n-1)/EM(n-1);
cm2(n) = cm2(n-1)+mum*em(n)*x(n-2)/EM(n-1);
end
c1m = c1m+c1; c2m = c2m+c2; e2 = e2+e.^2;
cm1m = cm1m+cm1; cm2m = cm2m+cm2; em2 = em2+em.^2;
end

c1ma = c1m/Niter; c2ma = c2m/Niter; e2a = e2/Niter;
cm1ma = cm1m/Niter; cm2ma = cm2m/Niter; em2a = em2/Niter;

```

```
% contour plots

a1p=(Lx(1):0.1:Lx(2))';
a2p=(Ly(1):0.1:Ly(2))';
L1=length(a1p);
L2=length(a2p);

for i=1:L1
    for j=1:L2
        a=[1 -a1p(i) -a2p(j)]';
        P(i,j)=a'*R*a;
    end
end

%Plots
Hf_1 = figure('units','SCRUN','position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',[0,0,6,4.8],...
    'NumberTitle','off','Name','Pr1006a');

subplot('position',[0.075,0.55,0.4,0.4]);
contour(a1p,a2p,P',(1:2:20)/10); hold on;
plot(cm1,cm2,'r',cm1ma,cm2ma,'g.');
%plot(c1ma,c2ma,'r','linewidth',1.5);
%plot(Lx,[0,0],'w:',[0,0],Ly,'w:',Lx,[-a1,-a1],'w:',[-a2,-a2],[0,0],'w:');
axis([Lx,Ly]);axis('square');
title(' (a) Averaged Trajectory','fontsize',10);
xlabel('c_{1}', 'fontsize',8); ylabel('c_{2}', 'fontsize',8);
set(gca,'xtick',[Lx(1),0,-a1,Lx(2)],'ytick',[Ly(1),-a2,0,Ly(2)],...
    'fontsize',6); grid;
hold off;

subplot('position',[0.575,0.55,0.4,0.4]); n = 0:N-1;
plot([-1,N],[-a2,-a2],'w:',[-1,N],[-a1,-a1],'w:'); hold on;
plot(n,cm1,'c',n,cm2,'g','linewidth',0.5);
plot(n,cm1ma,'w',n,cm2ma,'w','linewidth',1);
axis([-1,N,-1.2,1.2]);
xlabel('Number of iterations (n)', 'fontsize',8);
ylabel('Coefficients', 'fontsize',8);
title(' (b) {\bf c}(n) Learning Curve','fontsize',10);
set(gca,'xtick',[0,N-1],'ytick',[-a2,0,-a1], 'fontsize',6);
text(50, 0.5,'c_1(n)', 'fontsize',10);
text(50,-0.5,'c_2(n)', 'fontsize',10);
hold off;

subplot('position',[0.075,0.1,0.4,0.3]);
plot([-1,N],[varw,varw],'w:',n,em,'g',n,em2a,'m'); axis([-1,N,0,1.1]);
xlabel('Number of iterations (n)', 'fontsize',8);
title(' (c) MSE P(n) Learning Curve','fontsize',10);
set(gca,'xtick',[0,N-1],'ytick',[0,varw,1], 'fontsize',6);
```

```

subplot('position',[0.575,0.1,0.4,0.3]);
plot([-1,N],[varw,varw],'w:',n,e2a,'g',n,em2a,'c'); axis([-1,N,0,1]);
xlabel('Number of iterations (n)', 'fontsize',8);
title(' (d) Step-size effect on MSE', 'fontsize',10);
set(gca,'xtick',[0,N-1], 'ytick',[0,varw,0.5,1], 'fontsize',6);
text(30,0.1,' $\mu=0.1$  (NLMS)', 'fontsize',8);
text(50,0.7,' $\mu=0.04$  (LMS)', 'fontsize',8);

```

- (c) Choose $\tilde{\mu} = 0.1$ and repeat Problem 10.5(a) and (b). Compare your results in terms of convergence speed.

The Matlab script is shown below and the plots in Figure 10.6c. From the plots we again observe that the NLMS has the faster convergence.

```

% (c) NLMS for Echo Cancellation
% Data signal x(n): zero-mean, unit-variance white Gaussian process
% Echo System: co(n) = 0.9^n, 0 <= n <= 2
% Echo signal: y(n) = conv(x(n),co(n))

co = (0.9).^[0:2]; % Echo impulse response
var_y = sum(co.^2); % Variance of y(n)

% (ca) Echo Canceller design
M = 4; % LMS filter order
N = 400; x = randn(N,1); % Generate x(n)
y = filter(co,1,x); % Generate y(n)
mu = 0.1; c0 = zeros(M,1); % Initialization
[c,e] = nfirlms(x,y,mu,M,c0); % LMS algorithm

% Final Echo-canceller coefficients
cf = c(:,end);
*** Echo Canceller coefficients ***
1.0000    0.9000    0.8100   -0.0000

*** Echo System coefficients ***
1.0000    0.9000    0.8100

% (cb) Monte-Carlo analysis and the learning curve
Nsim = 500;

eavg = zeros(1,N,Nsim);
cavg = zeros(M,N,Nsim);

for k = 1:Nsim
    x = randn(N+M,1); y = filter(co,1,x);
    [c,e] = nfirlms(x,y,mu,M,c0);
    cavg(:,:,k) = c;
    eavg(:,:,k) = e;
end
cavg = mean(cavg,3);

```

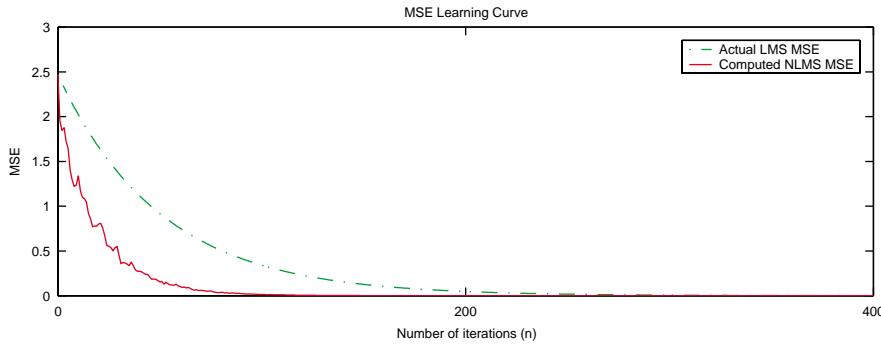


Figure 10.6c: Plots of the NLMS learning curve and its comparison with LMS in Problem 10.6c

```

eavg = eavg.^2; eavg = mean(eavg,3);
eavg = [var_y,eavg];

% Theoretical analysis
muo = 0.005;
n = 0:N; alpha = (1-4*muo+4*muo*muo*M);
ethe = var_y*alpha.^n;

% Plots
Hf_2 = figure('units',SCRUN,'position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',[0,0,5,2],...
    'NumberTitle','off','Name','Pr1006c');
subplot('position',[0.1,0.15,0.85,0.7]);
plot(n,ethe,'g-.' ,n,eavg,'r'); axis([0,N,0,3]);
xlabel('Number of iterations (n)', 'fontsize',label_fontsize);
title('MSE Learning Curve', 'fontsize',title_fontsize);
ylabel('MSE', 'fontsize',label_fontsize);
set(gca,'xtick',[0:200:N],'ytick',[0:0.5:3]);
legend('Actual LMS MSE','Computed NLMS MSE',1)

```

10.7 Another variation of the LMS algorithm is called the *sign-error* LMS algorithm, in which the coefficient update equation is given by

$$\mathbf{c}(n) = \mathbf{c}(n-1) + 2\mu \operatorname{sgn}[e(n)] \mathbf{x}(n), \text{ where}$$

$$\operatorname{sgn}[e(n)] = \begin{cases} 1 & \operatorname{Re}[e(n)] > 0 \\ 0 & \operatorname{Re}[e(n)] = 0 \\ -1 & \operatorname{Re}[e(n)] < 0 \end{cases}$$

- (a) Modify the function `firlms` to implement the NLMS algorithm and obtain the function `[c, e] = se-firlms(x, y, mu, M, c0)`.

```

function [c,e] = sefirlms(x,y,mu,M,c0)
% Sign-Error LMS Algorithm
% Computes filter coefficients in c and the corresponding error in

```

```
% e, given signal x, desired signal y, step size mu, filter order M, and
% the initial coefficient vector c0.
%
% [c,e] = sefirnlms(x,y,mu,M,c0)
%
N = length(x); % Length of x(n) & number of iterations
x = reshape(x,N,1); % x as a column vector
c = zeros(M,N); % Initialization of c
X = zeros(M,N); % Data matrix
xx = [zeros(M-1,1);x]; % Prepend M-1 zeros
for i = 1:M
    X(i,:) = xx(M-i+1:end-i+1)';
end
e = zeros(1,N);

%--LMS Algorithm for FIR filter - real-valued case
% Algorithm initialization
yhat = c0'*X(:,1);
e(1) = y(1) - yhat;
c(:,1) = c0 + 2*mu*X(:,1)*e(1);;
% Iterations
for n = 2:N
    yhat = c(:,n-1)'*X(:,n);
    e(n) = y(n) - yhat;
    c(:,n) = c(:,n-1) + 2*mu*X(:,n)*sign(e(n));
end
e = e(M+1:end);
c = c(:,M+1:end);
```

- (b) Repeat Problem 10.4 and compare your results in terms of convergence speed.

The Matlab script is shown below and the plots in Figure 10.7b. From the plots we observe that the SELMS has just a little faster convergence.

```
% (b) SELMS for LP Applications
% AR(2) process:
%   x(n) + a1*x(n-1) + a2*x(n-2) = w(n)
%
% Linear Predictor
%   y(n) = x(n); yhat = c1*x(n-1) + c2*x(n-2);

clc; close all
set(0,'defaultaxesfontsize',default_fontsize);
Lx = [-1,3]; % Maximum abs limit for the x-axis of contour plots
Ly = [-3,1]; % Maximum abs limit for the y-axis of contour plots
N = 501; % Signal size (also number of iterations)
Niter = 1000; % Ensemble size
mua = 0.01; % Step-size for LMS
num = 0.01; % Step-size for NLMS
```

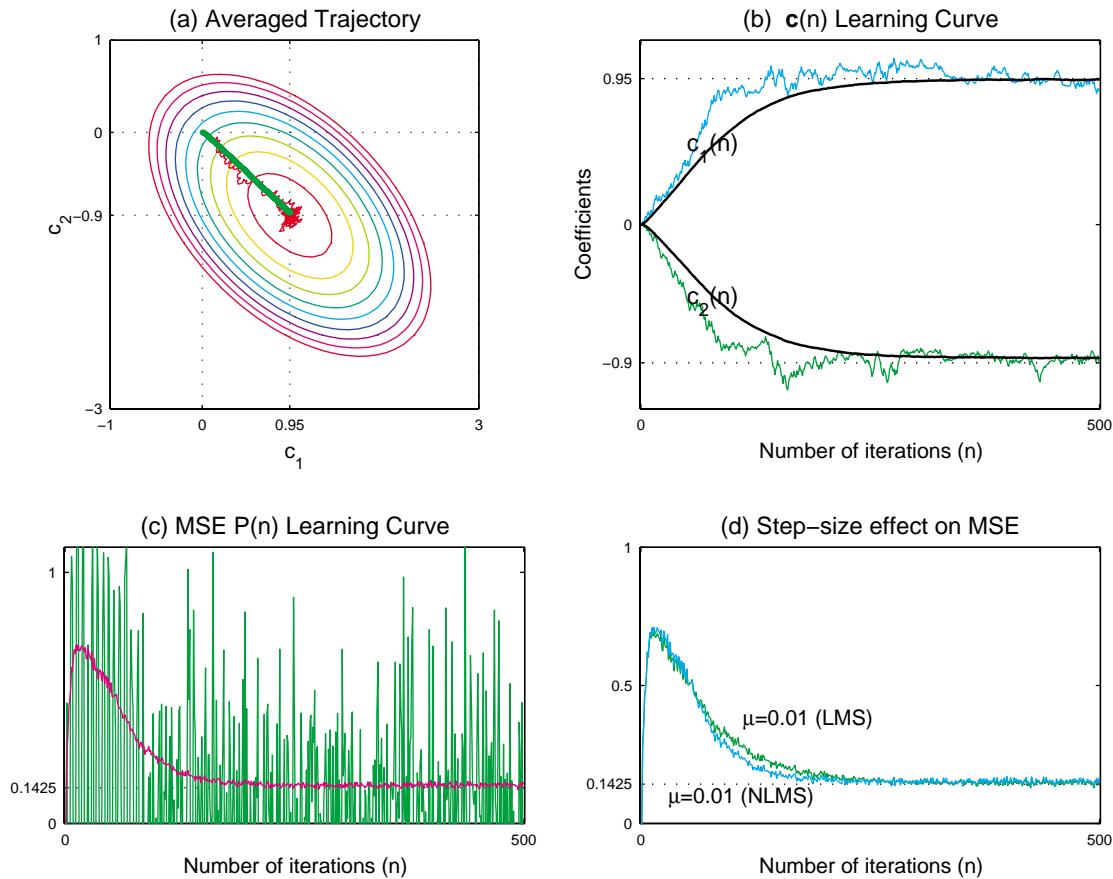


Figure 10.7b: Plots of the SELMS learning curve and its comparison with LMS in Problem 10.7b

```
% Generate AR(2) process
a1 = -0.950; a2 = 0.9; varx = 1; r0 = varx;
%a1a = a1; a2a = a2;
varw = ((1-a2)*((1+a2)^2-a1^2)/(1+a2))*varx;
r1 = -a1/(1+a2)*r0; r2 = (-a2+a1^2/(1+a2))*r0;
lam1 = (1-a1/(1+a2))*varx; lam2 = (1+a1/(1+a2))*varx;
XR = lam1/lam2;d = [r1;r2]; R = toeplitz([r0,r1,r2]);
% Ensemble averaging
e2 = zeros(N,1); e = e2; c1m = e; c2m = e; c1 = e; c2 = e;
em2 = zeros(N,1); em = em2; cm1m = em; cm2m = em; cm1 = em; cm2 = em;
for i=1:Niter
    w = sqrt(varw)*randn(N,1);
    x = filter(1,[1 a1 a2],w);

    c1(1) = 0; c1(2) = 0; c2(1) = 0; c2(2) = 0;
    cm1(1) = 0; cm1(2) = 0; cm2(1) = 0; cm2(2) = 0;
    % LMS Algorithm
    for n=3:N
        e(n) = x(n)-c1(n-1)*x(n-1)-c2(n-1)*x(n-2);
        c1(n) = c1(n-1)+2*mua*(e(n))*x(n-1);
        c2(n) = c2(n-1)+2*mua*(e(n))*x(n-2);
    end
    c1m = c1m + c1;
    cm1m = cm1m + cm1;
    c2m = c2m + c2;
    cm2m = cm2m + cm2;
end
c1m = c1m/N;
cm1m = cm1m/N;
c2m = c2m/N;
cm2m = cm2m/N;
```

```

em(n) = x(n)-cm1(n-1)*x(n-1)-cm2(n-1)*x(n-2);
cm1(n) = cm1(n-1)+2*mum*sign(em(n))*x(n-1);
cm2(n) = cm2(n-1)+2*mum*sign(em(n))*x(n-2);
end
c1m = c1m+c1; c2m = c2m+c2; e2 = e2+e.^2;
cm1m = cm1m+cm1; cm2m = cm2m+cm2; em2 = em2+em.^2;
end

c1ma = c1m/Niter; c2ma = c2m/Niter; e2a = e2/Niter;
cm1ma = cm1m/Niter; cm2ma = cm2m/Niter; em2a = em2/Niter;

% contour plots

a1p=(Lx(1):0.1:Lx(2))';
a2p=(Ly(1):0.1:Ly(2))';
L1=length(a1p);
L2=length(a2p);

for i=1:L1
    for j=1:L2
        a=[1 -a1p(i) -a2p(j)]';
        P(i,j)=a'*R*a;
    end
end

```

(c) Repeat Problem 10.5(a) and (b) and compare your results in terms of convergence speed.

The Matlab script is shown below and the plots in Figure 10.7c. From the plots we again observe that the NLMS has the faster convergence but a higher steady-state error.

```

% (ca) Echo Canceller design
M = 4; % LMS filter order
N = 400; x = randn(N,1); % Generate x(n)
y = filter(co,1,x); % Generate y(n)
mu = 0.1; c0 = zeros(M,1); % Initialization
[c,e] = nfirlms(x,y,mu,M,c0); % LMS algorithm

% Final Echo-canceller coefficients
cf = c(:,end);
*** Echo Canceller coefficients ***
0.9999  0.9000  0.8100  0.0000

*** Echo System coefficients ***
1.0000  0.9000  0.8100

% (cb) Monte-Carlo analysis and the learning curve
Nsim = 500;

eavg = zeros(1,N,Nsim);
cavg = zeros(M,N,Nsim);

```

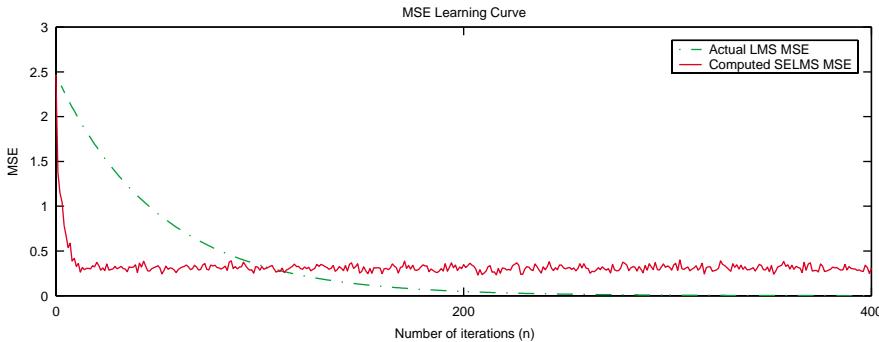


Figure 10.7c: Plots of the SELMS learning curve and its comparison with LMS in Problem 10.7c

```

for k = 1:Nsim
    x = randn(N+M,1);y = filter(co,1,x);
    [c,e] = sefirmlms(x,y,mu,M,c0);
    cavg(:,:,k) = c;
    eavg(:,:,k) = e;
end
cavg = mean(cavg,3);
eavg = eavg.^2; eavg = mean(eavg,3);
eavg = [var_y,eavg];

% Theoretical analysis
mu0 = 0.005;
n = 0:N; alpha = (1-4*mu0+4*mu0*mu0*M);
ethe = var_y*alpha.^n;

% Plots
Hf_2 = figure('units','SCRUN','position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',[0,0,5,2],...
    'NumberTitle','off','Name','Pr1006c');
subplot('position',[0.1,0.15,0.85,0.7]);
plot(n,ethe,'g-.',n,eavg,'r'); axis([0,N,0,3]);
xlabel('Number of iterations (n)', 'fontsize',label_fontsize);
title('MSE Learning Curve', 'fontsize',title_fontsize);
ylabel('MSE', 'fontsize',label_fontsize);
set(gca,'xtick',[0:200:N],'ytick',[0:0.5:3]);
legend('Actual LMS MSE','Computed SELMS MSE',1)

```

10.8 Consider an AR(1) process $x(n) = ax(n - 1) + w(n)$, where $w(n) \sim \text{WGN}(0, \sigma_w^2)$. We wish to design a one-step first-order linear predictor using the LMS algorithm

$$\hat{x}(n) = \hat{a}(n - 1)x(n - 1) \quad (1)$$

$$e(n) = x(n) - \hat{x}(n) \quad (2)$$

$$\hat{a}(n) = \hat{a}(n - 1) + 2\mu e(n)x(n - 1) \quad (3)$$

where μ is the adaptation step size.

- (a) Determine the autocorrelation $r_x(l)$, the optimum first-order linear predictor, and the corresponding MMSE.

The z -domain SDF, $R_x(z)$, is given by

$$\begin{aligned} R_x(z) &= H(z) H(z) R_w(z) \text{ where } H(z) = \frac{1}{1 - az^{-1}} \text{ and } R_w(z) = \sigma_w^2 \\ &= \sigma_w^2 \left(\frac{1}{1 - az^{-1}} \right) \left(\frac{1}{1 - az} \right) = \left(\frac{\sigma_w^2}{1 - a^2} \right) \left\{ \frac{1}{1 - az^{-1}} - \frac{1}{1 - a^{-1}z^{-1}} \right\}; |a| < |z| < |a|^{-1} \end{aligned}$$

Thus after inverse z -transformation

$$r_x(l) = \frac{\sigma_w^2}{1 - a^2} a^{|l|}, \quad \forall l$$

Now the optimum first-order linear predictor is given by

$$\hat{a} = \frac{r_x(1)}{r_x(0)} = a$$

and the MMSE is given by

$$P_e = r_x(0) - \hat{a} r_x(1) = \frac{\sigma_w^2}{1 - a^2} - \frac{\sigma_w^2}{1 - a^2} a^2 = \sigma_w^2$$

- (b) Using the independence assumption, first determine and then solve the difference equation for $E\{\hat{a}(n)\}$.

From (3), (1), and (2), we have

$$\begin{aligned} \hat{a}(n) &= \hat{a}(n-1) + 2\mu \{x(n) - \hat{a}(n-1)x(n-1)\} x(n-1) \\ &= \hat{a}(n-1) + 2\mu x(n)x(n-1) - 2\mu \hat{a}(n-1)x^2(n-1) \end{aligned}$$

Taking expectation of both sides using the independence assumption that $\hat{a}(n-1)$ and $x(n-1)$ are mutually independent (since $\hat{a}(n-1)$ depends only on $x(m)$, $m < n-1$), we obtain

$$\begin{aligned} E[\hat{a}(n)] &= E[\hat{a}(n-1)] + 2\mu E[x(n)x(n-1)] - 2\mu E[\hat{a}(n-1)]E[x^2(n-1)] \\ &= [1 - 2\mu r_x(0)] E[\hat{a}(n-1)] + 2\mu r_x(1) \end{aligned} \quad (4)$$

which is first-order difference equation for $E[\hat{a}(n)]$ driven by $2\mu r_x(1)$ for $n \geq 1$ with $E[\hat{a}(0)] = 0$. This equation can be solved using unilateral z -transform. Substituting

$$\alpha = 1 - 2\mu r_x(0), \beta = 2\mu r_x(1), \text{ and } y(n-1) \triangleq E[\hat{a}(n)] \quad (5)$$

in (4), we have

$$y(n) = \alpha y(n-1) + \beta u(n), \quad n \geq 0; \quad y(-1) = 0 \quad (6)$$

Taking unilateral z -transform of (6), we obtain

$$Y(z) = \alpha Y(z) + \beta \frac{1}{1 - z^{-1}}$$

or

$$Y(z) = \frac{\beta}{(1 - \alpha z^{-1})(1 - z^{-1})} = \frac{\beta}{1 - \alpha} \left\{ \frac{1}{1 - z^{-1}} - \frac{\alpha}{1 - \alpha z^{-1}} \right\}$$

or

$$y(n) = \frac{\beta}{1-\alpha} \{1 - \alpha^{n+1}\} u(n)$$

Using (5), we obtain the solution

$$E[\hat{a}(n)] = \frac{r_x(1)}{r_x(0)} \{1 - [1 - 2\mu r_x(0)]^n\} u(n-1) \quad (7)$$

Note that as $n \rightarrow \infty$ we have that $E[\hat{a}(n)] \rightarrow \frac{r_x(1)}{r_x(0)} = a$ (as expected in the absense of noise) if $|1 - 2\mu r_x(0)| < 1$.

- (c) For $a = \pm 0.95$, $\mu = 0.025$, $\sigma_x^2 = 1$, and $0 \leq n < N = 500$, determine the ensemble average of $E\{\hat{a}(n)\}$ using 200 independent runs and compare with the theoretical curve obtained in part (b).

The Matlab script is shown below and the plots in Figure 10.8c.

```
% AR(1) process parameters and LMS parameters
a = 0.95; N = 500; n = 0:N; var_x = 1; var_w = var_x*(1 - a*a);
mu = 0.025; Nsim = 200;

% Monte-Carlo analysis and the learning curve
Nsim = 500;

eavg = zeros(1,N,Nsim);
cavg = zeros(M,N,Nsim);

for k = 1:Nsim
    w = var_w*randn(N+1,1); x = filter(1,[1,-a],w);
    y = x(2:end); x = x(1:N);
    [c,e] = firnlms(x,y,mu,M,c0);
    cavg(:,:,k) = c;
    eavg(:,:,k) = e;
end
cavg = mean(cavg,3); cavg = [0,cavg];
eavg = eavg.^2; eavg = mean(eavg,3);
eavg = [1,eavg];

% Theoretical analysis
alpha = 1 - 2*mu*var_x;
cthe = a*(1-alpha.^n);

% Plots
Hf_1 = figure('units','SCRUN','position',SCRPOS,...
    'paperunits',PAPUN,'paperposition',[0,0,5,2],...
    'NumberTitle','off','Name','Pr1008c');
subplot('position',[0.1,0.15,0.85,0.7]);
plot(n,cthe,'g-.',n,cavg,'r'); axis([0,N,0,1]);
xlabel('Number of iterations (n)', 'fontsize',label_fontsize);
title('Linear-predictor Learning Curve', 'fontsize',title_fontsize);
ylabel('ahat(n)', 'fontsize',label_fontsize);
set(gca,'xtick',[0:200:N],'ytick',[0:0.2:1]);
legend('Theoretical','Averaged',4)
```

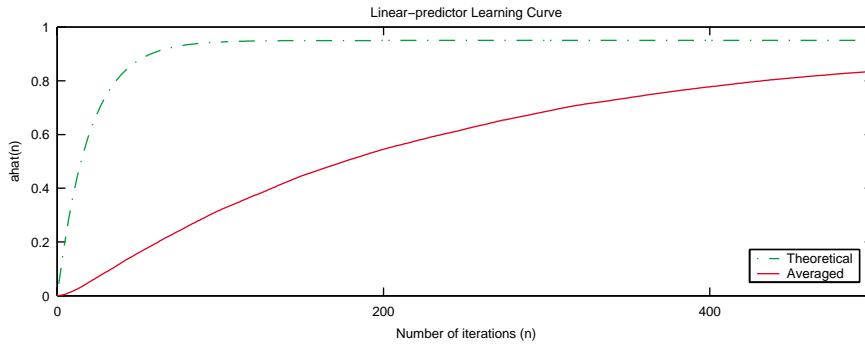


Figure 10.8c: Plots of learning curves in Problem 10.8c

- (d) Using the independence assumption, first determine and then solve the difference equation for $P(n) = E\{e^2(n)\}$.

An analysis similar to that in (b) can be used in this part.

- (e) Repeat part (c) for $P(n)$ and comment upon the results.

- 10.9** Using the a-posteriori error $\varepsilon(n) = y(n) - \mathbf{c}^H(n)\mathbf{x}(n)$, derive the coefficient updating formulas for the a-posteriori error LMS algorithm.

The a-posteriori form of the LMS algorithm

$$\mathbf{c}(n) = \mathbf{c}(n-1) + 2\mu\mathbf{x}(n)\varepsilon^*(n)$$

requires the a-posteriori error

$$\begin{aligned}\varepsilon(n) &= y(n) - \mathbf{c}^H(n)\mathbf{x}(n) \\ &= y(n) - [\mathbf{c}(n-1) + 2\mu\mathbf{x}(n)\varepsilon^*(n)]^H \mathbf{x}(n) \\ &= [y(n) - \mathbf{c}^H(n-1)\mathbf{x}(n)] - 2\mu e(n)\mathbf{x}^H(n)\mathbf{x}(n) \\ &= e(n) - 2\mu e(n)\mathbf{x}^H(n)\mathbf{x}(n)\end{aligned}$$

or

$$\varepsilon(n) = [\mathbf{I} - 2\mu\mathbf{x}^H(n)\mathbf{x}(n)] e(n)$$

Therefore, the a posteriori LMS algorithm is

$$\begin{aligned}e(n) &= y(n) - \mathbf{c}^H(n-1)\mathbf{x}(n) \\ \varepsilon(n) &= [\mathbf{I} - 2\mu\mathbf{x}^H(n)\mathbf{x}(n)] e(n) \\ \mathbf{c}(n) &= \mathbf{c}(n-1) + 2\mu\mathbf{x}(n)\varepsilon^*(n)\end{aligned}$$

- 10.10** Solve the interference cancellation problem described in Example 6.4.1, using the LMS algorithm, and compare its performance to that of the optimum canceler.

To be completed.

- 10.11** Repeat the convergence analysis of the LMS algorithm for the complex case, using formula (10.4.27) instead of (10.4.28).

This a tedious repetition of the work in Section 10.4.2 and should be given only as a project.

10.12 Consider the total transient excess MSE, defined by

$$P_{\text{tr}}^{(\text{total})} = \sum_{n=0}^{\infty} P_{\text{tr}}(n)$$

in Section 10.4.3.

- (a) Show that $P_{\text{tr}}^{(\text{total})}$ can be written as $P_{\text{tr}}^{(\text{total})} = \boldsymbol{\lambda}^T (\mathbf{I} - \mathbf{B})^{-1} \Delta \boldsymbol{\theta}(0)$, where $\Delta \boldsymbol{\theta}(0)$ is the initial (i.e., at $n = 0$) deviation of the filter coefficients from their optimum setting.

From (10.4.62) and (10.4.50) we have

$$\begin{aligned} P_{\text{tr}}^{(\text{total})} &= \sum_{n=0}^{\infty} P_{\text{tr}}(n) = \sum_{n=0}^{\infty} [P(n) - P(\infty)] \\ &= \sum_{n=0}^{\infty} \boldsymbol{\lambda}^T [\boldsymbol{\theta}(n) - \boldsymbol{\theta}(\infty)] = \sum_{n=0}^{\infty} \boldsymbol{\lambda}^T \mathbf{B}^n [\boldsymbol{\theta}(0) - \boldsymbol{\theta}(\infty)] \\ &= \boldsymbol{\lambda}^T \left(\sum_{n=0}^{\infty} \mathbf{B}^n \right) \Delta \boldsymbol{\theta}(0) = \boldsymbol{\lambda}^T (\mathbf{I} - \mathbf{B})^{-1} \Delta \boldsymbol{\theta}(0) \end{aligned}$$

where we have used (10.4.41) and $\sum_{n=0}^{\infty} \mathbf{B}^n = (\mathbf{I} - \mathbf{B})^{-1}$, which holds when the eigenvalues of \mathbf{B} have magnitude less than one.

- (b) Starting with the formula in step (a), show that

$$P_{\text{tr}}^{(\text{total})} = \frac{1}{4\mu} \frac{\sum_{i=1}^M \frac{\Delta \theta_i(0)}{1-2\mu\lambda_i}}{1 - \sum_{i=1}^M \frac{\mu\lambda_i}{1-2\mu\lambda_i}}$$

(b) From (10.4.33) we obtain

$$\mathbf{B} = \Lambda(\boldsymbol{\rho}) + 4\mu^2 \boldsymbol{\lambda} \boldsymbol{\lambda}^T$$

and

$$\mathbf{I} - \mathbf{B} = [\mathbf{I} - \Lambda(\boldsymbol{\rho})] - 4\mu^2 \boldsymbol{\lambda} \boldsymbol{\lambda}^T$$

Using the matrix inversion lemma (A.4) we obtain

$$(\mathbf{I} - \mathbf{B})^{-1} = [\mathbf{I} - \Lambda(\boldsymbol{\rho})]^{-1} + \frac{4\mu^2}{1 - 4\mu^2 \boldsymbol{\lambda}^T [\mathbf{I} - \Lambda(\boldsymbol{\rho})]^{-1} \boldsymbol{\lambda}} [\mathbf{I} - \Lambda(\boldsymbol{\rho})]^{-1} \boldsymbol{\lambda} \boldsymbol{\lambda}^T [\mathbf{I} - \Lambda(\boldsymbol{\rho})]^{-1}$$

Hence,

$$\begin{aligned} P_{\text{tr}}^{(\text{total})} &= \boldsymbol{\lambda}^T (\mathbf{I} - \mathbf{B})^{-1} \Delta \boldsymbol{\theta}(0) \\ &= \beta + \frac{4\mu^2 \alpha \beta}{1 - 4\mu^2 \alpha} = \frac{\beta}{1 - 4\mu^2 \alpha} \end{aligned}$$

where

$$\alpha = \boldsymbol{\lambda}^T [\mathbf{I} - \Lambda(\boldsymbol{\rho})]^{-1} \boldsymbol{\lambda} = \sum_{i=1}^M \frac{\lambda_i^2}{1 - \rho_i}$$

$$\beta = \lambda^T [\mathbf{I} - \Lambda(\rho)]^{-1} \Delta\theta(0) = \sum_{i=1}^M \frac{\lambda_i \Delta\theta_i(0)}{1 - \rho_i}$$

From (10.4.36) can easily see that

$$\frac{\lambda_i}{1 - \rho_i} = \frac{1}{4\mu(1 - 2\mu\lambda_i)}$$

Thus

$$\alpha = \frac{1}{4\mu} \sum_{i=1}^M \frac{\lambda_i}{1 - 2\mu\lambda_i}$$

$$\beta = \frac{1}{4\mu} \sum_{i=1}^M \frac{\Delta\theta_i(0)}{1 - 2\mu\lambda_i}$$

and

$$P_{tr}^{total} = \frac{1}{4\mu} \frac{\sum_{i=1}^M \frac{\Delta\theta_i(0)}{1 - 2\mu\lambda_i}}{1 - \sum_{i=1}^M \frac{\mu\lambda_i}{1 - 2\mu\lambda_i}}$$

(c) Show that if $\mu\lambda_k \ll 1$, then

$$P_{tr}^{(total)} \simeq \frac{1}{4\mu} \frac{\sum_{i=1}^M \Delta\theta_i(0)}{1 - \mu \text{tr}(\mathbf{R})} \simeq \frac{1}{4\mu} \sum_{i=1}^M \Delta\theta_i(0)$$

which is formula (10.4.62), discussed in Section 10.4.3.

When $\mu\lambda_i \ll 1$ we have $1 - \mu\lambda_i \simeq 1$ and $1 - \mu \text{tr} \mathbf{R} \simeq 1$. Hence

$$P_{tr}^{total} \simeq \frac{1}{4\mu} \frac{\sum_{i=1}^M \Delta\theta_i(0)}{1 - \mu \text{tr} \mathbf{R}} \simeq \frac{1}{4\mu} \sum_{i=1}^M \Delta\theta_i(0)$$

10.13 The frequency sampling structure for the implementation of an FIR filter $H(z) = \sum_{n=0}^{M-1} h(n) \cdot z^{-n}$ is specified by the following relation

$$H(z) = \frac{1 - z^{-M}}{M} \sum_{k=0}^{M-1} \frac{H(e^{j2\pi k/M})}{1 - e^{j2\pi k/M} z^{-1}} \triangleq H_1(z) H_2(z)$$

where $H_1(z)$ is a comb filter with M zeros equally spaced on the unit circle and $H_2(z)$ is a filter bank of resonators. Note that $\tilde{H}(k) \triangleq H(e^{j2\pi k/M})$, the DFT of $\{h(n)\}_0^{M-1}$, is the coefficients of the filter. Derive an LMS-type algorithm to update these coefficients, and sketch the resulting adaptive filter structure.

If we define

$$Y_k(z) \triangleq \frac{1}{1 - e^{j2\pi k/M} z^{-1}} \frac{1 - z^M}{M} X(x)$$

we can obtain the adaptive filter structure shown in Figure 10.13. An adaptive LMS filter is obtained using the following algorithm

$$\begin{aligned}\hat{y}(n) &= \sum_{k=0}^{M-1} c_k(n-1) y_k(n) \\ e(n) &= y(n) - \hat{y}(n) \\ c_k(n) &= c_k(n-1) + 2\mu e(n) \hat{y}_k(n)\end{aligned}$$

To understand the nature of this structure, consider the frequency response of the k th parallel path of the filter (resonator)

$$H_k(z) = \frac{1 - z^M}{M(1 - e^{j2\pi k/M}z^{-1})}$$

or for $z = e^{2\pi f}$

$$H_k(e^{2\pi f}) = \frac{1 - e^{-2\pi f M}}{M(1 - e^{j2\pi k/M}e^{-j2\pi f})} = \frac{e^{-\pi f M} (e^{\pi f M} - e^{-\pi f M})}{M e^{j\pi(\frac{k}{M}-f)} \left[e^{-j\pi(\frac{k}{M}-f)} - e^{j\pi(\frac{k}{M}-f)} \right]}$$

which leads to

$$|H_k(e^{2\pi f})| = \frac{1}{M} \left| \frac{\sin \pi M f}{\sin \pi (\frac{k}{M} - f)} \right| = \begin{cases} 1, & f = \frac{k}{M} \\ 0, & f \neq \frac{k}{M} \end{cases}$$

Therefore, when

$$y(n) = \sum_{k=0}^{M-1} A_k \cos \frac{2\pi k}{M} n$$

each coefficient of the adaptive filter can be updated without any interference from the other coefficients.

- 10.14** There are applications in which the use of a non-MSE criterion may be more appropriate. To this end, suppose that we wish to design and study the behavior of an “LMS-like” algorithm that minimizes the cost function $P^{(k)} = E\{e^{2k}(n)\}$, $k = 1, 2, 3, \dots$, using the model defined in Figure 10.19.

Since the cost function is

$$P^{(k)} = E\{e^{2k}(n)\}, \quad k = 1, 2, \dots$$

the error $e(n)$ must be real-valued. Hence,

$$e(n) = y(n) - \mathbf{c}^T(n-1) \mathbf{x}(n)$$

- (a) Use the instantaneous gradient vector to derive the coefficient updating formula for this LMS-like algorithm.

An LMS-type algorithm is obtained using the instantaneous gradient

$$\frac{\partial P^{(k)}}{\partial \mathbf{c}} = 2k e^{2k-1}(n) \frac{\partial e(n)}{\partial \mathbf{c}} = -2k e^{2k-1}(n) \mathbf{x}(n)$$

where we have used $e(n) = y(n) - \mathbf{c}^T \mathbf{x}(n)$. Hence

$$\mathbf{c}(n) = \mathbf{c}(n-1) + 2k \mu e^{2k-1}(n) \mathbf{x}(n)$$

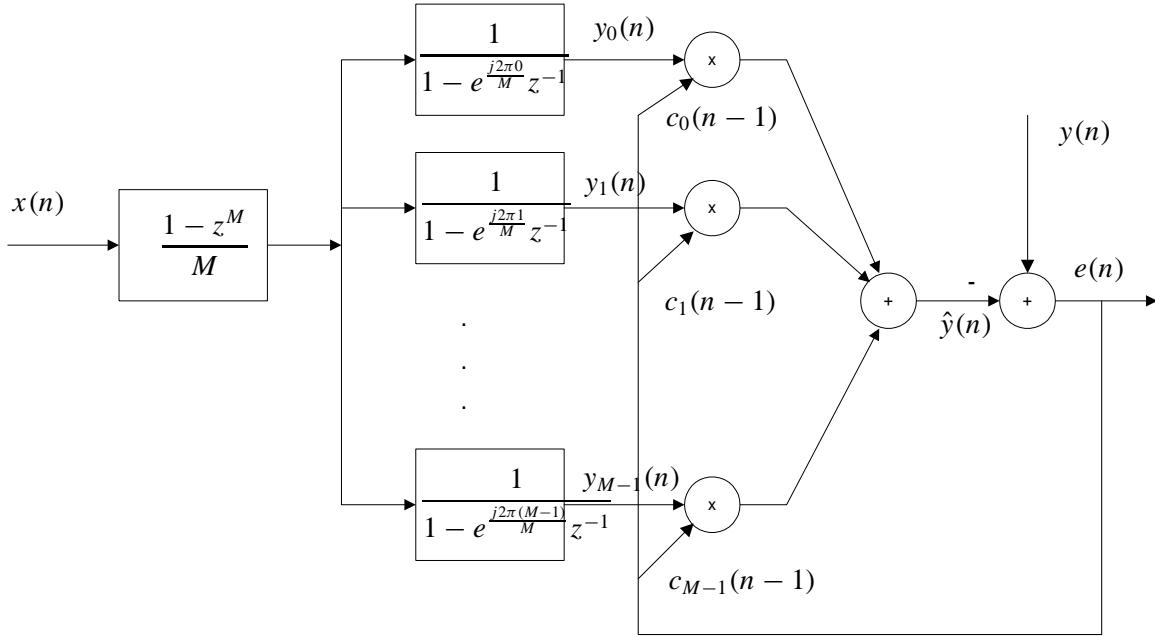


Figure 10.13: Adaptive LMS filter using the frequency sampling structure.

(b) Using the assumptions introduced in Section 10.4.2 show that

$$E\{\tilde{\mathbf{c}}(n)\} = [\mathbf{I} - 2\mu k(2k-1)E\{e_o^{2(k-1)}(n)\}\mathbf{R}]E\{\tilde{\mathbf{c}}(n-1)\}$$

where \mathbf{R} is the input correlation matrix.

To study convergence in the mean we consider the model given by (10.4.13)

$$y(n) = \mathbf{c}_o^T \mathbf{x}(n) + e_o(n)$$

We start by expressing the a priori error as

$$\begin{aligned} e(n) &= y(n) - \mathbf{c}^T(n-1)\mathbf{x}(n) \\ &= \mathbf{c}_o^T \mathbf{x}(n) - \mathbf{c}^T(n-1)\mathbf{x}(n) + e_o(n) \\ &= [\mathbf{c}_o - \mathbf{c}(n-1)]\mathbf{x}(n) + e_o(n) \end{aligned}$$

or

$$e(n) = -\tilde{\mathbf{c}}^T(n-1)\mathbf{x}(n) + e_o(n)$$

where $\tilde{\mathbf{c}}(n) \triangleq \mathbf{c}(n) - \mathbf{c}_o$ is the coefficient error vector. Hence

$$\begin{aligned} e^{2k-1}(n) &= [-\tilde{\mathbf{c}}^T(n-1)\mathbf{x}(n) + e_o(n)]^{2k-1} \\ &= e_o^{2k-1}(n) \left[1 - \frac{\tilde{\mathbf{c}}^T(n-1)\mathbf{x}(n)}{e_o(n)} \right]^{2k-1} \\ &\simeq e_o^{2k-1}(n) \left[1 - (2k-1) \frac{\tilde{\mathbf{c}}^T(n-1)\mathbf{x}(n)}{e_o(n)} \right] \\ &= e_o^{2k-1}(n) - (2k-1)\tilde{\mathbf{c}}^T(n-1)\mathbf{x}(n)e_o^{2k-2}(n) \end{aligned}$$

when $\tilde{\mathbf{c}}(n)$ takes very small values, that is when the coefficients are changing very slowly.

Substituting into the coefficient updating formula, we obtain

$$\tilde{\mathbf{c}}(n) \simeq \tilde{\mathbf{c}}(n-1) + 2k\mu e_o^{2k-1}(n)\mathbf{x}(n) - (2k-1)2\mu k\mathbf{x}(n)\mathbf{x}^T(n)\tilde{\mathbf{c}}(n-1)e_o^{2k-2}(n)$$

or

$$\tilde{\mathbf{c}}(n) \simeq [\mathbf{I} - 2\mu k(2k-1)e_o^{2k-2}(n)\mathbf{x}(n)\mathbf{x}^T(n)]\tilde{\mathbf{c}}(n-1) + 2\mu k e_o^{2k-1}(n)\mathbf{x}(n)$$

Taking the expectation of both sides and using the independence assumptions, we have

$$E\{\tilde{\mathbf{c}}(n)\} \simeq [\mathbf{I} - 2\mu k(2k-1)E\{e_o^{2k-2}(n)\}\mathbf{R}]E\{\tilde{\mathbf{c}}(n-1)\}$$

- (c) Show that the derived algorithm converges in the mean if

$$0 < 2\mu < \frac{1}{k(2k-1)E\{e_o^{2(k-1)}(n)\}\lambda_{\max}}$$

where λ_{\max} is the largest eigenvalue of \mathbf{R} .

Using $\mathbf{R} = \mathbf{Q}\Lambda\mathbf{Q}^T$ and $\mathbf{v}(n) \triangleq \mathbf{Q}^T E\{\tilde{\mathbf{c}}(n)\}$, the last equation becomes

$$\mathbf{v}(n) \simeq [\mathbf{I} - 2\mu k(2k-1)E\{e_o^{2k-2}(n)\}\Lambda]\mathbf{v}(n-1)$$

or

$$v_i(n) \simeq [1 - 2\mu k(2k-1)E\{e_o^{2k-2}(n)\}\lambda_i]v_i(n-1)$$

whose solution is

$$v_i(n) = [1 - 2\mu k(2k-1)E\{e_o^{2k-2}(n)\}\lambda_i]^n v_i(0)$$

This solution converges if

$$|1 - 2\mu k(2k-1)E\{e_o^{2k-2}(n)\}\lambda_{\max}| < 1$$

or equivalently

$$0 < 2\mu < \frac{1}{k(2k-1)E\{e_o^{2k-2}(n)\}\lambda_{\max}}$$

- (d) Show that for $k = 1$ the results in parts (a) to (c) reduce to those for the standard LMS algorithm.

For $k = 1$ we have

$$0 < 2\mu < \frac{1}{\lambda_{\max}}$$

which is identical to (10.4.19).

- 10.15** Consider the noise cancellation system shown in Figure 10.6. The useful signal is a sinusoid $s(n) = \cos(\omega_0 n + \phi)$, where $\omega_0 = \pi/16$ and the phase ϕ is a random variable uniformly distributed from 0 to 2π . The noise signals are given by $v_1(n) = 0.9 v_1(n-1) + w(n)$ and $v_2(n) = -0.75 v_2(n-1) + w(n)$, where the sequences $w(n)$ are WGN(0, 1).

The input signal to the optimum/adaptive filter is $v_2(n)$ while the “desired signal” is $y(n) \triangleq s(n) + v_1(n)$. Hence we will need the following correlations:

$$\begin{aligned} r_{v_2}(l) &= \frac{\sigma_w^2}{1 - (-0.75)^2} (-0.75)^{|l|} = \frac{16}{7} (-0.75)^{|l|} \\ r_y(l) &= r_s(l) + r_{v_1}(l) = \frac{1}{2} \cos \omega_0 l + \frac{\sigma_w^2}{1 - (0.9)^2} (0.9)^{|l|} = \frac{1}{2} \cos \omega_0 l + \frac{100}{19} (0.9)^{|l|} \\ r_{yv_2}(l) &= r_{v_1v_2}(l) = h_1(l) * h_2(-l) * r_w(l) = h_1(l) * h_2(-l) \end{aligned}$$

where $h_1(n) = (0.9)^n u(n)$, $h_2(n) = (-0.75)^n u(n)$, and $r_w(l) = \delta(l)$.

- (a) Design an optimum filter of length M and choose a reasonable value for M_o by plotting the MMSE as a function of M .

The optimum filter of length M is given by

$$\mathbf{R}_{v_2} \mathbf{c}_o = \mathbf{r}_{yv_2}$$

where \mathbf{R}_{v_2} is an $(M \times M)$ autocorrelation matrix and \mathbf{r}_{yv_2} is an $(M \times 1)$ crosscorrelation vector. The MMSE is given by

$$P_o = r_y(0) - \mathbf{c}_o^T \mathbf{r}_{yv_2}$$

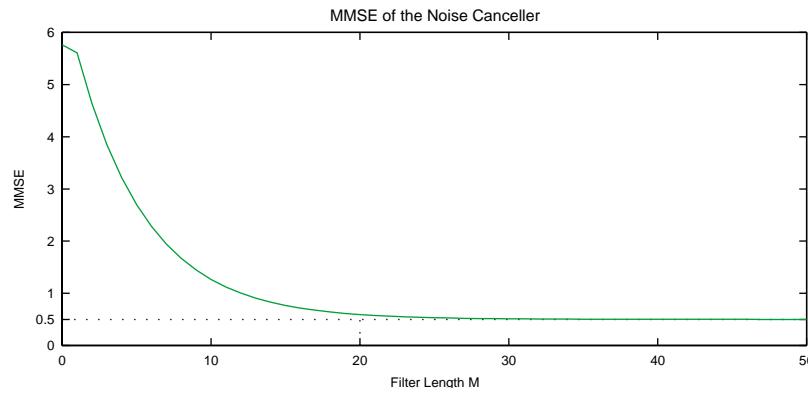
Thus the only quantity remains to be computed is $r_{yv_2}(l)$. Consider, for $l \geq 0$

$$\begin{aligned} r_{yv_2}(l) &= h_1(l) * h_2(-l) = \sum_{k=-\infty}^{\infty} h_1(k) h_2(k-l) \\ &= \sum_{k=l}^{\infty} (0.9)^k (-0.75)^{k-l} = (-0.75)^{-l} \sum_{k=l}^{\infty} (-0.675)^k \\ &= (-0.75)^{-l} (-0.675)^l \sum_{k=0}^{\infty} (-0.675)^k = 0.59701 (0.9)^l, \quad l \geq 0 \end{aligned}$$

The rest of the problem is solved using Matlab. The script is given below and the plot is shown in Figure 10.15a.

```
% System Parameters
omg0 = pi/16; a1 = 0.9; a2 = -0.75; var_w = 1;

% (a) Design of the Optimum filter and its MMSE
Mmax = 50; Po = zeros(Mmax,1);
var_v2 = var_w/(1-a2*a2);
var_v1 = var_w/(1-a1*a1);
ry0 = 0.5*cos(omg0*0)+var_v1;
for M = 1:Mmax
    Rv2 = toeplitz(var_v2*a2.^[0:M-1]);
    ryv2 = (1/(1-a1*a2))*a1.^[0:M-1]';
    co = Rv2\ryv2;
    Po(M) = ry0 - co'*ryv2;
end
Po = [ry0;Po]; M = 0:Mmax;
```

**Figure 10.15a:** Plot of the MSE vs M

```

Hf_1 = figure('units','SCRUN','position',SCRPOS,...  

    'paperunits',PAPUN,'paperposition',[0,0,5,2],...  

    'NumberTitle','off','Name','Pr1015a');  

plot([0,Mmax],[0.5,0.5],'w:',[20,20],[0,Po(21)],...  

    'w:',M,Po,'g');  

axis([0,Mmax,0,6]);  

xlabel('Filter Length M','fontsize',label_fontsize);  

title('MMSE of the Noise Canceller','fontsize',title_fontsize);  

ylabel('MMSE','fontsize',label_fontsize);  

set(gca,'xtick',[0:10:Mmax],'ytick',[0,0.5,1:6]);  
  

% Choose Mo = 20;  

Mo = 20;  

Rv2 = toeplitz(var_v2*a2.^[0:Mo-1]);  

ryv2 = (1/(1-a1*a2))*a1.^[0:Mo-1]';  

co = Rv2\ryv2;

```

From the plot in Figure 10.15a a reasonable value for M_o is 20.

- (b) Design an LMS filter with M_o coefficients and choose the step size μ to achieve a 10 percent misadjustment.

```

% (b) LMS for Noise canceller  

% Generate Desired and Input signals  

N = 1000; n = [0:N-1]';  

phi = 2*pi*rand(1,1); s = cos(omg0*n+phi);  

w = randn(N,1); v1 = filter(1,[1,-a1],w);  

v2 = filter(1,[1,-a2],w); y = s + v1;  

mu = 0.005;  

c0 = zeros(Mo,1);  

[c,e,yhat] = fir1m(v2,y,mu,Mo,c0);

```

- (c) Plot the signals $s(n)$, $s(n) + v_1(n)$, $v_2(n)$, the clean signal $e_o(n)$ using the optimum filter, and the clean signal $e_{lms}(n)$ using the LMS filter, and comment upon the obtained results.

The script is given below and various plots are shown in Figure 10.15c. From these plots we observe that the optimum as well as adaptive filters show a reasonable performance.

```

eo = y - filter(co,1,v2); elms = y - yhat;
subplot('position',[0.1,0.82,0.85,0.09]);
plot(n(500:1000),s(500:1000),'g'); axis([N/2,N,-5,5]);
title('Original Signal s(n)', 'fontsize', title_fontsize);
set(gca,'xtick',[500,1000]);
subplot('position',[0.1,0.64,0.85,0.09]);
plot(n(500:1000),y(500:1000),'m'); axis([N/2,N,-10,10]);
title('Primary Signal s(n)+v_1(n)', 'fontsize', title_fontsize);
set(gca,'xtick',[500,1000]);
subplot('position',[0.1,0.46,0.85,0.09]);
plot(n(500:1000),v2(500:1000),'m'); axis([N/2,N,-10,10]);
title('Secondary Signal v_2(n)', 'fontsize', title_fontsize);
set(gca,'xtick',[500,1000]);
subplot('position',[0.1,0.28,0.85,0.09]);
plot(n(500:1000),eo(500:1000),'g'); axis([N/2,N,-5,5]);
title('Clean Signal e_o(n)', 'fontsize', title_fontsize);
set(gca,'xtick',[500,1000]);
subplot('position',[0.1,0.10,0.85,0.09]);
plot(n(500:1000),elms(500:1000),'g'); axis([N/2,N,-5,5]);
title('Clean Signal e_{LMS}(n)', 'fontsize', title_fontsize);
set(gca,'xtick',[500,1000]);
xlabel('sample index n', 'fontsize', label_fontsize);

```

10.16 A modification of the LMS algorithm, known as the *momentum LMS (MLMS)*, is defined by

$$\mathbf{c}(n) = \mathbf{c}(n-1) + 2\mu e^*(n)\mathbf{x}(n) + \alpha[\mathbf{c}(n-1) - \mathbf{c}(n-2)]$$

where $|\alpha| < 1$ (Roy and Shynk 1990).

- (a) Rewrite the previous equation to show that the algorithm has the structure of a low-pass ($0 < \alpha < 1$) or a high-pass ($-1 < \alpha < 0$) filter.

The coefficient updating equation can be written as

$$[\mathbf{c}(n) - \mathbf{c}(n-1)] = \alpha [\mathbf{c}(n-1) - \mathbf{c}(n-2)] + 2\mu \mathbf{x}(n)e^*(n)$$

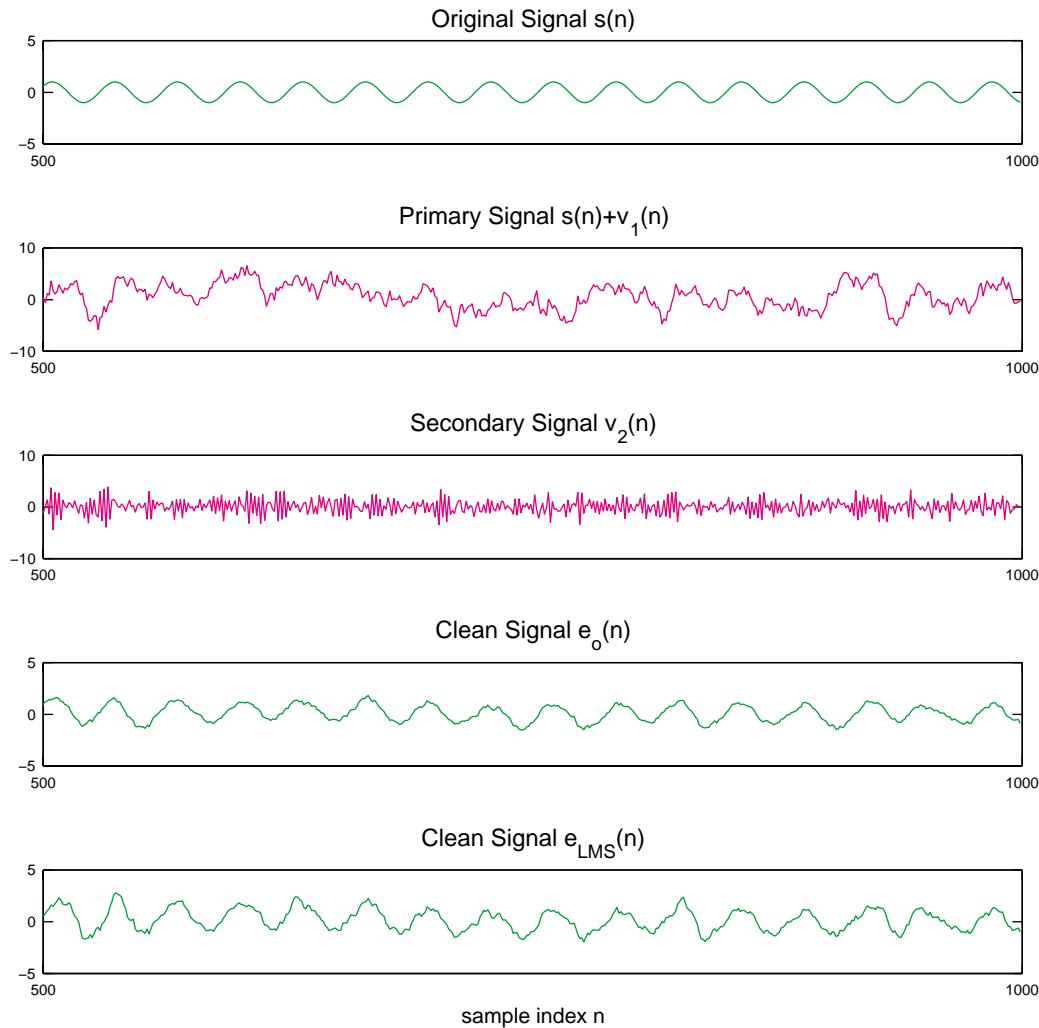
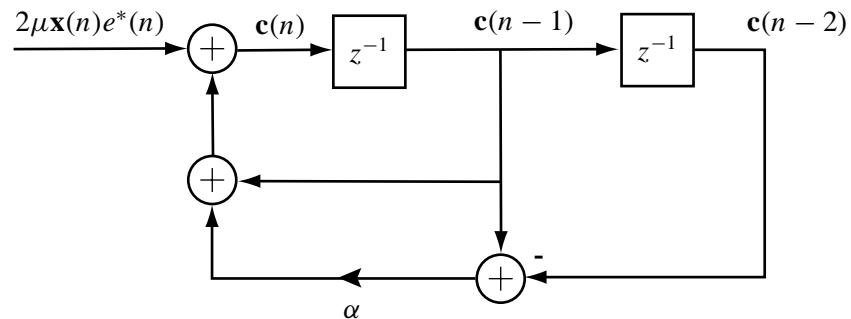
which is a low-pass filter for $0 < \alpha < 1$ and a high-pass filter for $-1 < \alpha < 0$. The structure for this equation is given in Figure 10.16a.

- (b) Explain intuitively the effect of the momentum term $\alpha[\mathbf{c}(n-1) - \mathbf{c}(n-2)]$ on the filter's convergence behavior.

The term $[\alpha \mathbf{c}(n-1) - \mathbf{c}(n-2)] = \alpha \Delta \mathbf{c}(n-1)$ adds an extra amount of change to the LMS updating term $2\mu \mathbf{x}(n)e^*(n)$. During the transient phase $\|\Delta \mathbf{c}(n-1)\|$ is large and accelerates convergence. However, the price paid is a larger amount of misadjustment in steady-state. To overcome this problem we can set $\alpha = 0$, that is we switch to the LMS algorithm, when $\|\Delta \mathbf{c}(n-1)\|$ becomes small.

- (c) Repeat the computer equalization experiment in Section 10.4.4, using both the LMS and the MLMS algorithms for the following cases, and compare their performance:

- i. $W = 3.1, \mu_{\text{lms}} = \mu_{\text{mlms}} = 0.01, \alpha = 0.5$.
- ii. $W = 3.1, \mu_{\text{lms}} = 0.04, \mu_{\text{mlms}} = 0.01, \alpha = 0.5$.
- iii. $W = 3.1, \mu_{\text{lms}} = \mu_{\text{mlms}} = 0.04, \alpha = 0.2$.

**Figure 10.15c:** Signal plots**Figure 10.16a:** The momentum LMS algorithm.

iv. $W = 4$, $\mu_{\text{lms}} = \mu_{\text{mlms}} = 0.03$, $\alpha = 0.3$.

10.17 In Section 10.4.5 we presented the leaky LMS algorithm [see (10.4.88)]

$$\mathbf{c}(n) = (1 - \alpha\mu)\mathbf{c}(n - 1) + \mu e^*(n)\mathbf{x}(n)$$

where $0 < \alpha \ll 1$ is the leakage coefficient.

(a) Show that the coefficient updating equation can be obtained by minimizing

$$P(n) = |e(n)|^2 + \alpha \|\mathbf{c}(n)\|^2$$

We have

$$P(n) = |e(n)|^2 + \alpha \|\mathbf{c}\|^2 = e(n)e^*(n) + \alpha \mathbf{c}^H \mathbf{c}$$

Using results from Appendix B we obtain

$$\nabla_{\mathbf{c}} P(n) = -2\mathbf{x}(n)e^*(n) + 2a\mathbf{c}$$

Hence the a priori LMS coefficient updating is

$$\begin{aligned} \mathbf{c}(n) &= \mathbf{c}(n - 1) - \mu [-2\mathbf{x}(n)e^*(n) + 2a\mathbf{c}(n - 1)] \\ &= (1 - \alpha 2\mu)\mathbf{c}(n - 1) + 2\mu\mathbf{x}(n)e^*(n) \\ &\triangleq (1 - \alpha\mu)\mathbf{c}(n - 1) + \mu\mathbf{x}(n)e^*(n) \end{aligned}$$

if we replace 2μ by μ .

(b) Using the independence assumptions, show that

$$E\{\mathbf{c}(n)\} = [\mathbf{I} - \mu(\mathbf{R} + \alpha\mathbf{I})]E\{\mathbf{c}(n - 1)\} + \mu\mathbf{d}$$

where $\mathbf{R} = E\{\mathbf{x}(n)\mathbf{x}^H(n)\}$ and $\mathbf{d} = E\{\mathbf{x}(n)y^*(n)\}$.

Taking expectations of both sides in the last equation, we have

$$E\{\mathbf{c}(n)\} = (1 - \alpha\mu)E\{\mathbf{c}(n - 1)\} + \mu E\{\mathbf{x}(n)e^*(n)\}$$

Using

$$e(n) = y(n) - \mathbf{c}^T(n - 1)\mathbf{x}(n)$$

we obtain

$$E\{\mathbf{c}(n)\} = (1 - \alpha\mu)E\{\mathbf{c}(n - 1)\} + \mu E\{\mathbf{x}(n)y^*(n)\} - \mu E\{\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{c}(n - 1)\}$$

or using the independence assumptions

$$E\{\mathbf{c}(n)\} = (1 - \alpha\mu)E\{\mathbf{c}(n - 1)\} + \mu\mathbf{d} - \mu\mathbf{R}E\{\mathbf{c}(n - 1)\}$$

where $\mathbf{d} = E\{\mathbf{x}(n)y^*(n)\}$ and $\mathbf{R} = E\{\mathbf{x}(n)\mathbf{x}^H(n)\}$. Thus

$$E\{\mathbf{c}(n)\} = [\mathbf{I} - \mu(\mathbf{R} + \alpha\mathbf{I})]E\{\mathbf{c}(n - 1)\} + \mu\mathbf{d}$$

- (c) Show that if $0 < \mu < 2/(\alpha + \lambda_{\max})$, where λ_{\max} is the maximum eigenvalue of \mathbf{R} , then

$$\lim_{n \rightarrow \infty} E\{\mathbf{c}(n)\} = (\mathbf{R} + \alpha \mathbf{I})^{-1} \mathbf{d}$$

that is, in the steady state $E\{\mathbf{c}(\infty)\} \neq \mathbf{c}_o = \mathbf{R}^{-1}\mathbf{d}$.

The difference equation for $E\{\mathbf{c}(n)\}$ has a finite solution as $n \rightarrow \infty$ if the eigenvalues of $[\mathbf{I} - \mu(\mathbf{R} + \alpha \mathbf{I})]$ are less than one in magnitude, that is, if

$$0 < \mu < \frac{2}{\alpha + \lambda_{\max}}$$

In this case, as $n \rightarrow \infty$, $E\{\mathbf{c}(n)\} \simeq E\{\mathbf{c}(n-1)\}$ and the solution of the difference equation is

$$\lim_{n \rightarrow \infty} E\{\mathbf{c}(n)\} = (\mathbf{R} + \alpha \mathbf{I})^{-1} \mathbf{d}$$

which is identical with the optimum filter only when $\alpha = 0$.

If the input $x(n)$ is replaced by $z(n) = x(n) + v(n)$, where $v(n)$ is uncorrelated white noise with zero mean and variance α , we have

$$\mathbf{R}_z = \mathbf{R} + \alpha \mathbf{I}$$

Therefore, the leaky LMS algorithm is equivalent to the LMS algorithm with white noise of variance α added to its input.

- 10.18** There are various communications and speech signal processing applications that require the use of filters with linear phase (Manolakis et al. 1984). For simplicity, assume that m is even.

- (a) Derive the normal equations for an optimum FIR filter that satisfies the constraints
 - i. $\mathbf{c}_m^{(lp)} = \mathbf{J}\mathbf{c}_m^{(lp)}$ (linear phase)
 - ii. $\mathbf{c}_m^{(cgd)} = -\mathbf{J}\mathbf{c}_m^{(cgd)}$ (constant group delay).
- (b) Show that the obtained optimum filters can be expressed as $\mathbf{c}_m^{(lp)} = \frac{1}{2}(\mathbf{c}_m + \mathbf{J}\mathbf{c}_m)$ and $\mathbf{c}_m^{(cgd)} = \frac{1}{2}(\mathbf{c}_m - \mathbf{J}\mathbf{c}_m)$, where \mathbf{c}_m is the unconstrained optimum filter.
- (c) Using the results in part (b) and the algorithm of Levinson, derive lattice-ladder structure for the constrained optimum filters.
- (d) Repeat parts (a), (b), and (c) for the linear predictor with linear phase, which is specified by $\mathbf{a}_m^{(lp)} = \mathbf{J}\mathbf{a}_m^{(lp)}$.
- (e) Develop an LMS algorithm for the linear-phase filter $\mathbf{c}_m^{(lp)} = \mathbf{J}\mathbf{c}_m^{(lp)}$ and sketch the resulting structure. Can you draw any conclusions regarding the step size and the misadjustment of this filter compared to those of the unconstrained LMS algorithm?

To be completed.

- 10.19** In this problem, we develop and analyze by simulation an LMS-type adaptive lattice predictor introduced in Griffiths (1977). We consider the all-zero lattice filter defined in (7.5.7), which is completely specified by the lattice parameters $\{k_m\}_0^{M-1}$. The input signal is assumed wide-sense stationary.

- (a) Consider the cost function

$$P_m^{\text{fb}} = E\{|e_m^{\text{f}}(n)|^2 + |e_m^{\text{b}}(n)|^2\}$$

which provides the total prediction error power at the output of the m th stage, and show that

$$\frac{\partial P_m^{fb}}{\partial k_m} = 2E\{e_m^{f*}(n)e_{m-1}^b(n-1) + e_{m-1}^{f*}(n)e_m^b(n)\}$$

Consider the lattice recursions

$$\begin{aligned} e_m^f(n) &= e_{m-1}^f(n) + k_{m-1}^* e_{m-1}^b(n-1) \\ e_m^b(n) &= k_{m-1} e_{m-1}^f(n) + e_{m-1}^b(n-1) \end{aligned}$$

If we assume real-valued signals, we have

$$\begin{aligned} \frac{\partial P_m^{fb}}{\partial k_{m-1}} &= E \left\{ 2e_m^f(n) \frac{\partial e_m^f(n)}{\partial k_{m-1}} + 2e_m^b(n) \frac{\partial e_m^b(n)}{\partial k_{m-1}} \right\} \\ &= E \left\{ 2e_m^f(n) e_{m-1}^b(n-1) + 2e_m^b(n) e_{m-1}^f(n) \right\} \end{aligned}$$

For complex-valued signals, we follow Appendix B to obtain

$$\frac{\partial P_m^{fb}}{\partial k_{m-1}^*} = E \left\{ 2e_m^{f*}(n) e_{m-1}^b(n-1) + 2e_m^b(n) e_{m-1}^{f*}(n) \right\}$$

(b) Derive the updating formula

$$k_m(n) = k_{m-1}(n-1) + 2\mu(n)[e_m^{f*}(n)e_{m-1}^b(n-1) + e_{m-1}^{f*}(n)e_m^b(n)]$$

where the normalized step size $\mu(n) = \bar{\mu}/E_{m-1}^b(n)$ is computed in practice by using the formula

$$E_{m-1}^b(n) = \alpha E_{m-1}^b(n-1) + (1-\alpha)[|e_{m-1}^f(n)|^2 + |e_{m-1}^b(n-1)|^2]$$

where $0 < \alpha < 1$. Explain the role and proper choice of α , and determine the proper initialization of the algorithm.

An LMS-type algorithm is obtained by using the instantaneous gradient. Therefore

$$k_{m-1}(n) = k_{m-1}(n-1) - 2\mu(n) \left[e_m^{f*}(n)e_{m-1}^b(n-1) + e_m^b(n)e_{m-1}^{f*}(n) \right]$$

The quantity $E_{m-1}(n)$ represents the total energy of the forward and backward linear prediction errors at the input of the m th stage from start to present time n . The updating formula for $E_{m-1}(n)$ is a first-order recursive low-pass filter with a pole at $z = a$. It provides a filter with exponential memory to help track in a nonstationary SOE.

The algorithm is summarized as follows

Initialization

$$e_m^f(-1) = e_m^b(-1) = 0$$

$$E_{m-1}(-1) = \delta > 0$$

$$k_m(-1) = 0$$

For $n = 0, 1, 2, \dots$

$$e_0^f(n) = e_0^b(n) = x(n)$$

For $m = 0, 1, \dots, M-1$

$$e_m^f(n) = e_{m-1}^f(n) + k_{m-1}^* e_{m-1}^b(n-1)$$

$$e_m^b(n) = k_{m-1} e_{m-1}^f(n) + e_{m-1}^b(n-1)$$

$$E_{m-1}(n) = \alpha E_{m-1}(n-1) + (1-\alpha) \left[e_m^{f*}(n) e_{m-1}^b(n-1) + e_m^b(n) e_{m-1}^{f*}(n) \right]$$

$$k_{m-1}(n) = k_{m-1}(n-1) - 2 \frac{\tilde{\mu}}{E_{m-1}(n)} \left[|e_{m-1}^f(n)|^2 + |e_{m-1}^b(n-1)|^2 \right]$$

- (c) Write a Matlab function to implement the derived algorithm, and compare its performance with that of the LMS algorithm in the linear prediction problem discussed in Example 10.4.1.

To be completed.

- 10.20** Consider a signal $x(n)$ consisting of a harmonic process plus white noise, that is,

$$x(n) = A \cos(\omega_1 n + \phi) + w(n)$$

where ϕ is uniformly distributed from 0 to 2π and $w(n) \sim \text{WGN}(0, \sigma_w^2)$.

- (a) Determine the output power $\sigma_y^2 = E\{y^2(n)\}$ of the causal and stable filter

$$y(n) = \sum_{k=0}^{\infty} h(k) x(n-k)$$

and show that we can cancel the harmonic process using the ideal notch filter

$$H(e^{j\omega}) = \begin{cases} 1 & \omega = \omega_1 \\ 0 & \text{otherwise} \end{cases}$$

Is the obtained ideal notch filter practically realizable? That is, is the system function rational? Why?

The output power of the filter is

$$E\{y^2(n)\} = \sigma_y^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} |H(e^{j\omega})|^2 R_x(e^{j\omega}) d\omega$$

where

$$R_x(e^{j\omega}) = \frac{1}{2} A^2 \delta(e^{j(\omega-\omega_1)}) + \frac{1}{2} A^2 \delta(e^{j(\omega+\omega_1)}) + \sigma_w^2$$

Therefore

$$\sigma_y^2 = A^2 |H(e^{j\omega_1})|^2 + \sigma_w^2 \frac{1}{2\pi} \int_{-\pi}^{\pi} |H(e^{j\omega})|^2 d\omega$$

and the ideal notch filter is given by

$$|H(e^{j\omega})| = \begin{cases} 0, & \omega = \pm\omega_1 \\ 1, & \text{otherwise} \end{cases}$$

As a result of the nature of this discontinuity the ideal notch filter cannot be realized by a rational system function. The output power of the filter is

$$\sigma_y^2 = \begin{cases} \sigma_w^2, & \omega = \pm\omega_1 \\ A^2 + \sigma_w^2, & \text{otherwise} \end{cases}$$

(b) Consider the second-order notch filter (see Problem 2.31)

$$H(z) = \frac{D(z)}{A(z)} = \frac{1 + a z^{-1} + z^{-2}}{1 + a\rho z^{-1} + \rho^2 z^{-2}} = \frac{D(z)}{D(z/\rho)}$$

where $-1 < \rho < 1$ determines the steepness of the notch and $a = 2 \cos \omega_0$ its frequency. We fix ρ , and we wish to design an adaptive filter by adjusting a .

- i. Show that for $\rho \simeq 1$, $\sigma_y^2 = A^2 |H(e^{j\omega_1})|^2 + \sigma_w^2$, and plot σ_y^2 as a function of the frequency ω_0 for $\omega_1 = \pi/6$.
- ii. Evaluate $d\sigma_y^2(a)/da$ and show that the minimum of $\sigma_y^2(a)$ occurs for $a = -2 \cos \omega_1$.

We can easily find that $H(z)$ has zeros on the unit circle at $z_{1,2} = e^{\pm j\omega_0}$ and poles at $p_{1,2} = \rho e^{\pm j\omega_0}$. From the pole-zero plot we can see that

$$|H(e^{j\omega})| \simeq \begin{cases} 0, & \omega = \pm\omega_0 \\ 1, & \text{otherwise} \end{cases}$$

The steepness of the notch increases as ρ approaches one. Therefore, if $\omega_0 = \omega_1$ we have $\sigma_y^2 = A^2 |H(e^{j\omega_1})|^2 + \sigma_w^2$ since the area under $|H(e^{j\omega})|^2$ is close to one. This is illustrated in Figure 10.20b.

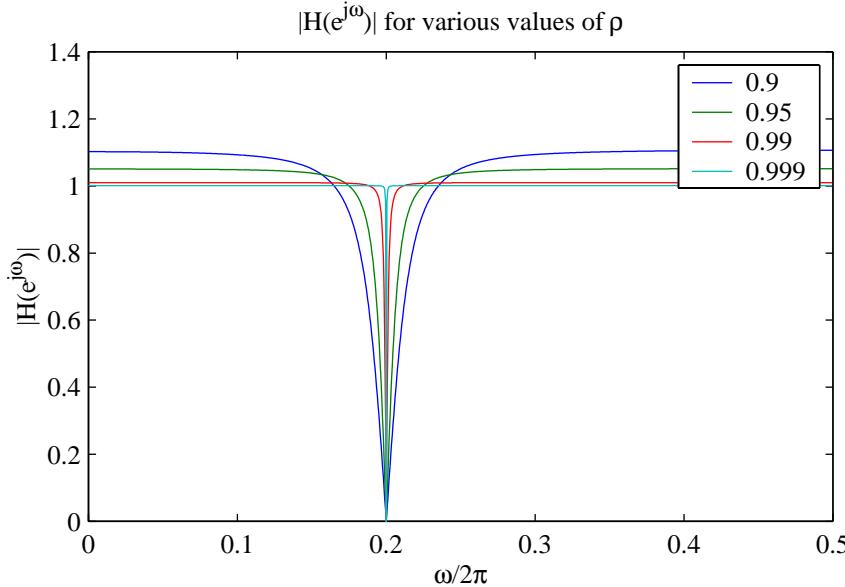


Figure 10.20b: Magnitude response for pole angle $\omega_0 = 2\pi(0.2)$ and various radius.

To find the value of a that minimizes σ_y^2 we compute

$$\frac{\partial \sigma_y^2(a)}{\partial a} = \frac{A^2}{2} \frac{(a + 2 \cos \omega_1) [1 - \rho^3 + a\rho(1 - \rho) \cos \omega_1 - \rho(1 - \rho) \cos 2\omega_1]}{|A(e^{j\omega_1})|^4}$$

This is zero if $a = -2 \cos \omega_1$ or

$$1 - \rho^3 + a\rho(1 - \rho) \cos \omega_1 - \rho(1 - \rho) \cos 2\omega_1 = 0$$

For $-2 < a < 2$ we can show that

$$1 - \rho^3 + a\rho(1 - \rho) \cos \omega_1 - \rho(1 - \rho) \cos 2\omega_1 \geq (1 - \rho^2) > 0$$

which implies that $a = -2 \cos \omega_1$ is the only solution.

- (c) Using a direct-form II structure for the implementation of $H(z)$ and the property

$$dY(z)/da = [dH(z)/da]X(z)$$

show that the following relations

$$\begin{aligned} s_2(n) &= -a(n-1)\rho s_2(n-1) - \rho^2 s_2(n-2) + (1 - gr)s_1(n-1) \\ g(n) &= s_2(n) - \rho s_2(n-2) \\ s_1(n) &= -a(n-1)\rho s_1(n-1) - \rho^2 s_1(n-2) + x(n) \\ y(n) &= s_1(n) + a(n-1)s_1(n-1) + s_1(n-2) \\ a(n) &= a(n-1) - 2\mu y(n)g(n) \end{aligned}$$

constitute an adaptive LMS notch filter. Draw its block diagram realization.

The instantaneous gradient is

$$\frac{\partial y^2(n)}{\partial a} = 2y(n) \frac{\partial y(n)}{\partial a}$$

which leads to the following LMS updating

$$a(n+1) = a(n) - \mu y(n)g(n)$$

where

$$g(n) = \frac{\partial y(n)}{\partial a}$$

We shall compute $g(n)$ in the z-domain. From $Y(z) = H(z)X(z)$ we have

$$\frac{\partial Y(z)}{\partial a} = \frac{\partial H(z)}{\partial a}X(z)$$

and

$$\frac{\partial H(z)}{\partial a} = \frac{\partial}{\partial a} \frac{1 + az^{-1} + z^{-2}}{1 + a\rho z^{-1} + \rho^2 z^{-2}} = \frac{(1 - \rho)z^{-1}(1 - \rho z^{-2})}{(1 + a\rho z^{-1} + \rho^2 z^{-2})^2}$$

The direct form II implementations for the filter and the gradient are

$$Y(z) = (1 + az^{-1} + z^{-2}) \underbrace{\left[\frac{X(z)}{1 + a\rho z^{-1} + \rho^2 z^{-2}} \right]}_{S_1(z)}$$

and

$$G(z) = (1 - \rho z^{-2}) \underbrace{\left[\frac{(1 - \rho)z^{-1}S_1(z)}{1 + a\rho z^{-1} + \rho^2 z^{-2}} \right]}_{S_2(z)}$$

respectively. The corresponding difference equations are

$$\begin{aligned}s_2(n) &= -a(n-1)\rho s_2(n-1) - \rho^2 s_2(n-2) + (1 - gr)s_1(n-1) \\g(n) &= s_2(n) - \rho s_2(n-2) \\s_1(n) &= -a(n-1)\rho s_1(n-1) - \rho^2 s_1(n-2) + x(n) \\y(n) &= s_1(n) + a(n-1)s_1(n-1) + s_1(n-2) \\a(n) &= a(n-1) - 2\mu y(n)g(n)\end{aligned}$$

and provide the complete algorithm for the implementation of the second-order adaptive notch filter. The resulting structure is shown in Figure 10.20c.

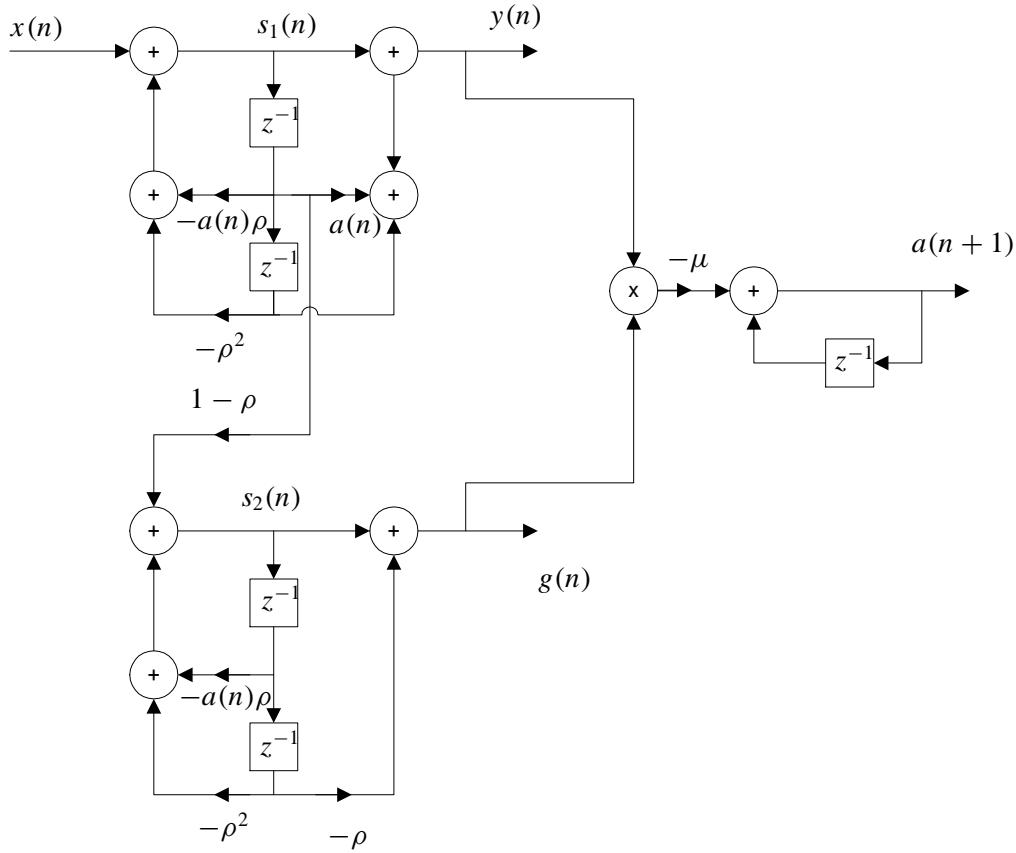


Figure 10.20c: Block diagram implementation of the adaptive notch filter.

- (d) Simulate the operation of the obtained adaptive filter for $\rho = 0.9$, $\omega_1 = \pi/6$, and SNR 5 and 15 dB. Plot $\omega_0(n) = \arccos[-a(n)/2]$ as a function of n , and investigate the tradeoff between convergence rate and misadjustment by experiment with various values of μ .

The MATLAB script for simulating this experiment is given below and Figures 10.20d1 and 10.20d2 show filter input-output signals and frequency tracking for $f_0 = 0.2$, SNR=5dB, $\mu = 0.001$ and $\mu = 0.0025$. The trade-off between speed of convergence and misadjustment is clearly evident.

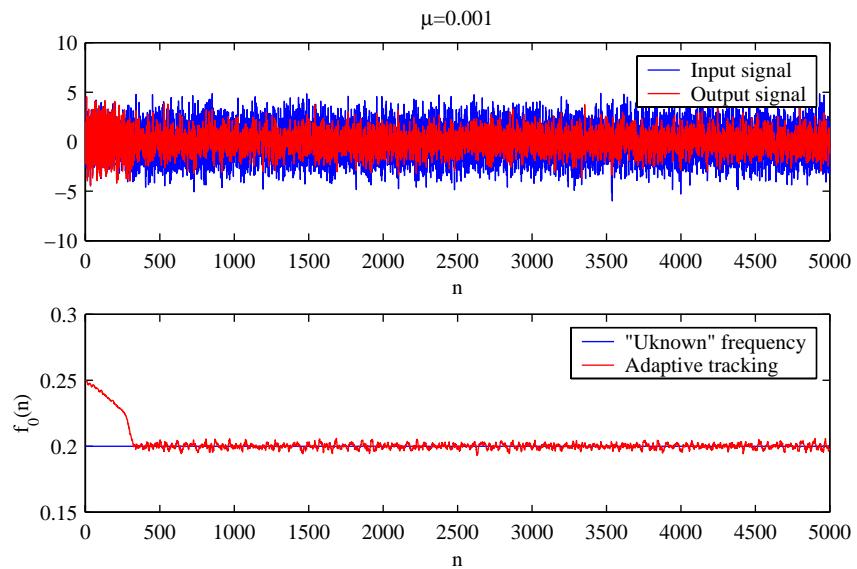


Figure 10.20d1: Tracking characteristics for $\mu = 0.001$.

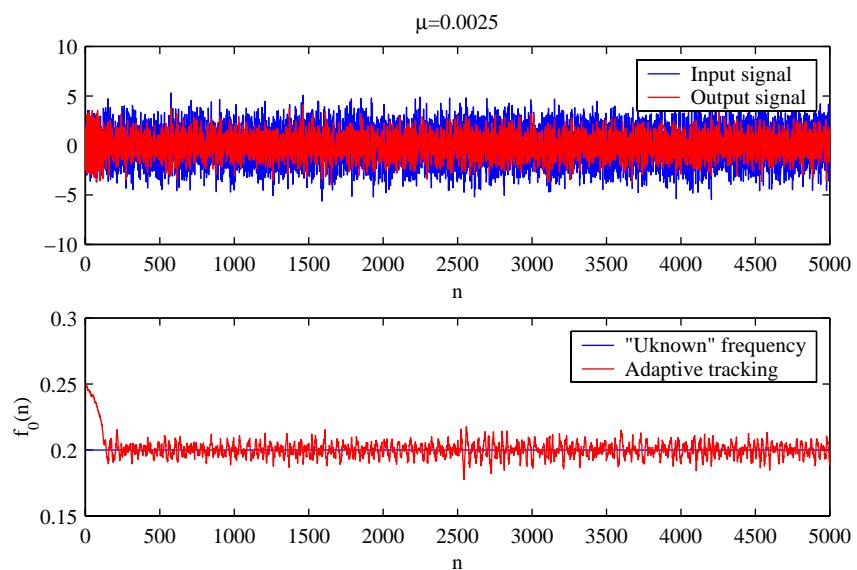


Figure 10.20d2: Tracking characteristics for $\mu = 0.0025$.

10.21 Consider the AR(2) process given in Problem 10.4. We will design the adaptive linear predictor using the RLS algorithm. The adaptive predictor is a second-order one given by $\mathbf{c}(n) = [c_1(n) \ c_2(n)]^T$.

- (a) Develop a Matlab function to implement the RLS algorithm given in Table 10.6:

$$[\mathbf{c}, \mathbf{e}] = \text{rls}(\mathbf{x}, \mathbf{y}, \lambda, \delta, M, \mathbf{c}_0).$$

which computes filter coefficients in \mathbf{c} and the corresponding error in \mathbf{e} given signal \mathbf{x} , desired signal \mathbf{y} , forgetting factor λ , initialization parameter δ , filter order M , and the initial coefficient vector \mathbf{c}_0 . To update $\mathbf{P}(n)$, compute only the upper or lower triangular part and determine the other part by using Hermitian symmetry.

- (b) Generate 500 samples of $x(n)$ and obtain linear predictor coefficients using the above function. Use a very small value for δ (for example, 0.001) and various values of $\lambda = 0.99, 0.95, 0.9, \text{ and } 0.8$. Plot predictor coefficients as a function of time along with the true coefficients for each λ , and discuss your observations. Also compare your results with those in Problem 10.4.
- (c) Repeat each simulation above 1000 times to get corresponding learning curves, which are obtained by averaging respective squared errors $|e(n)|^2$. Plot these curves and compare their steady-state value with the theoretical MSE.

To be completed.

10.22 Consider a system identification problem where we observe the input $x(n)$ and the noisy output $y(n) = y_o(n) + v(n)$, for $0 \leq n \leq N - 1$. The unknown system is specified by the system function

$$H_o(z) = \frac{0.0675 + 0.1349z^{-1} + 0.0675z^{-2}}{1 - 1.1430z^{-1} + 0.4128z^{-2}}$$

and $x(n) \sim \text{WGN}(0, 1)$, $v(n) \sim \text{WGN}(0, 0.01)$, and $N = 300$.

- (a) Model the unknown system using an LS FIR filter, with $M = 15$ coefficients, using the no-windowing method. Compute the total LSE E_{ls} in the interval $n_0 \leq n \leq N - 1$ for $n_0 = 20$.
- (b) Repeat part (a) for $0 \leq n \leq n_0 - 1$ (do not compute E_{ls}). Use the vector $\mathbf{c}(n_0)$ and the matrix $\mathbf{P}(n_0) = \hat{\mathbf{R}}^{-1}(n_0)$ to initialize the CRLS algorithm. Compute the total errors $E_{apr} = \sum_{n=n_0}^{N-1} e^2(n)$ and $E_{apost} = \sum_{n=n_0}^{N-1} \varepsilon^2(n)$ by running the CRLS for $n_0 \leq n \leq N - 1$.
- (c) Order the quantities E_{ls} , E_{apr} , E_{apost} by size and justify the resulting ordering.

To be completed.

10.23 Prove Equation (10.5.25) using the identity $\det(\mathbf{I}_1 + \mathbf{AB}) = \det(\mathbf{I}_2 + \mathbf{BA})$, where identity matrices \mathbf{I}_1 and \mathbf{I}_2 and matrices \mathbf{A} and \mathbf{B} have compatible dimensions.

We have

$$\begin{aligned} \hat{\mathbf{R}}(n) &= \lambda \hat{\mathbf{R}}(n-1) + \mathbf{x}(n)\mathbf{x}^H(n) \\ \lambda \hat{\mathbf{R}}(n-1) &= \hat{\mathbf{R}}(n) - \mathbf{x}(n)\mathbf{x}^H(n) \\ \lambda \hat{\mathbf{R}}^{-1}(n) \hat{\mathbf{R}}(n-1) &= \mathbf{I} - [\hat{\mathbf{R}}^{-1}(n)\mathbf{x}(n)]\mathbf{x}^H(n) \end{aligned}$$

Taking the determinant of both sides gives

$$\begin{aligned} \lambda^M \frac{\det \hat{\mathbf{R}}(n-1)}{\det \hat{\mathbf{R}}(n)} &= \det \left\{ \mathbf{I} - [\hat{\mathbf{R}}^{-1}(n)\mathbf{x}(n)]\mathbf{x}^H(n) \right\} \\ &= \det \left[1 - \mathbf{x}^H(n)\hat{\mathbf{R}}^{-1}(n)\mathbf{x}(n) \right] = \alpha(n) \end{aligned}$$

- 10.24** Derive the normal equations that correspond to the minimization of the cost function (10.5.36), and show that for $\delta = 0$ they are reduced to the standard set (10.5.2) of normal equations. For the situation described in Problem 10.22, run the CRLS algorithm for various values of δ and determine the range of values that provides acceptable performance.

The cost function can be written as

$$E(n) = \delta \lambda^n \mathbf{c}^H \mathbf{c} + E_y(n) - \mathbf{c}^H \hat{\mathbf{d}}(n) - \hat{\mathbf{d}}^H(n) \mathbf{c} + \mathbf{c}^H \hat{\mathbf{R}}(n) \mathbf{c}$$

Computing the gradient with respect to \mathbf{c} gives

$$\frac{\partial E(n)}{\partial \mathbf{c}} = \delta \lambda^n \mathbf{c}^* - \hat{\mathbf{d}}^*(n) + [\hat{\mathbf{R}}(n) \mathbf{c}]^* = \mathbf{0}$$

or

$$[\delta \lambda^n + \hat{\mathbf{R}}(n)] \mathbf{c}(n) = \hat{\mathbf{d}}(n)$$

For $\delta = 0$, we obtain the standard normal equations. It can be easily shown that the matrix $\tilde{\mathbf{R}}(n) = \delta \lambda^n + \hat{\mathbf{R}}(n)$ obeys the recursion

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

Thus, the development of the CRLS algorithm remains unchanged. Clearly, small values of δ do not significantly affect the transient response of the filter and have no effect on the long run due to the exponential forgetting memory.

- 10.25** Modify the CRLS algorithm in Table 10.6 so that its coefficients satisfy the linear-phase constraint $\mathbf{c} = \mathbf{J}\mathbf{c}^*$. For simplicity, assume that $M = 2L$; that is, the filter has an even number of coefficients. To be completed.

- 10.26** Following the approach used in Section 7.5.1 to develop the structure shown in Figure 7.1, derive a similar structure based on the Cholesky (*not* the LDL^H) decomposition.

The LS filter is specified by $\hat{\mathbf{R}}(n)\mathbf{c}(n) = \hat{\mathbf{d}}(n)$. Using the Cholesky decomposition $\hat{\mathbf{R}}(n) = \tilde{\mathbf{L}}_m(n)\tilde{\mathbf{L}}_m^H(n)$ and the definitions in Section 10.6.1, we obtain

$$e_m(n) = y(n) - \hat{y}_m(n) = y(n) - \tilde{\mathbf{k}}_m^H(n-1)\tilde{\mathbf{w}}_m(n)$$

or

$$\hat{y}_m(n) = \hat{y}_{m-1}(n) + \tilde{k}_m^*(n-1)\tilde{w}_m(n)$$

Using the last equation and the elements of the triangular matrix $\tilde{\mathbf{B}}_m(n) \triangleq \tilde{\mathbf{L}}_m^{-1}(n)$, we can easily modify Figure 7.1 to obtain the desired structure.

- 10.27** Show that the partitioning (10.7.3) of $\hat{\mathbf{R}}_{m+1}(n)$ to obtain the same partitioning structure as (10.7.2) is possible only if we apply the prewindowing condition $\mathbf{x}_m(-1) = \mathbf{0}$. What is the form of the partitioning if we abandon the prewindowing assumption?

We have

$$\begin{aligned} \hat{\mathbf{R}}_{m+1}(n) &= \sum_{j=0}^n \lambda^{n-j} \mathbf{x}_{m+1}(j) \mathbf{x}_{m+1}^H(j) \\ &= \sum_{j=0}^n \lambda^{n-j} \begin{bmatrix} \mathbf{x}_m(i) \\ x(j-m) \end{bmatrix} \begin{bmatrix} \mathbf{x}_m^H(j) & x^*(j-m) \end{bmatrix} \\ &= \begin{bmatrix} \hat{\mathbf{R}}_m(n) & \hat{\mathbf{r}}_m^b(n) \\ \hat{\mathbf{r}}_m^{bH}(n) & \hat{\rho}_m^b(n) \end{bmatrix} \end{aligned}$$

where

$$\begin{aligned}\hat{\mathbf{r}}_m^b(n) &= \sum_{j=0}^n \lambda^{n-j} \mathbf{x}_m(j) x^*(j-m) \\ \hat{\rho}_m^b(n) &= \sum_{j=0}^n \lambda^{n-j} |x(j-m)|^2\end{aligned}$$

Similarly, using the partitioning

$$\mathbf{x}_{m+1}(j) = \begin{bmatrix} x(j) \\ \mathbf{x}_m(j-1) \end{bmatrix}$$

we have

$$\hat{\mathbf{R}}_{m+1}(n) = \begin{bmatrix} \hat{\rho}_m^f(n) & \hat{\mathbf{r}}_m^{fH}(n) \\ \hat{\mathbf{r}}_m^f(n) & \hat{\mathbf{R}}_m(n-1) \end{bmatrix}$$

where

$$\begin{aligned}\sum_{j=0}^n \lambda^{n-j} \mathbf{x}_m(j-1) \mathbf{x}_m^H(j-1) &= \underbrace{\lambda^n \mathbf{x}_m(-1) \mathbf{x}_m^H(-1)}_{\text{pre-windowing} \Rightarrow 0} + \hat{\mathbf{R}}_m(n-1) \\ \hat{\mathbf{r}}_m^f(n) &= \sum_{j=0}^n \lambda^{n-j} \mathbf{x}_m(j-1) x^*(j) \\ \hat{\rho}_m^f(n) &= \sum_{j=0}^n \lambda^{n-j} |x(j)|^2\end{aligned}$$

10.28 Derive the normal equations and the LSE formulas given in Table 10.11 for the FLP and the BLP methods.

For the FLP, the error is

$$\varepsilon_m^f(n) = x(n) + \mathbf{a}_m^H(n) \mathbf{x}_m(n-1)$$

and the cost function is

$$E_m^f(n) = \sum_{j=0}^n \lambda^{n-j} |\varepsilon_m^f(j)|^2$$

With the goal to minimize the cost function, the normal equations can be found directly

$$\begin{aligned}E_m^f(n) &= \sum_{j=0}^n \lambda^{n-j} |x(n) + \mathbf{a}_m^H(n) \mathbf{x}_m(n-1)|^2 \\ &= \sum_{j=0}^n \lambda^{n-j} x(n) x^*(n) + \sum_{j=0}^n \lambda^{n-j} \mathbf{a}_m^H(n) \mathbf{x}_m(n-1) x^*(n) \\ &\quad + \sum_{j=0}^n \lambda^{n-j} x(n) \mathbf{x}_m^H(n-1) \mathbf{a}_m(n) + \sum_{j=0}^n \lambda^{n-j} \mathbf{a}_m^H(n) \mathbf{x}_m(n-1) \mathbf{x}_m^H(n-1) \mathbf{a}_m(n) \\ &= E_x(n) + \mathbf{a}_m^H(n) \hat{\mathbf{r}}_m^f(n) + \hat{\mathbf{r}}_m^{fH}(n) \mathbf{a}_m(n) + \mathbf{a}_m^H(n) \hat{\mathbf{R}}_m(n-1) \mathbf{a}_m(n)\end{aligned}$$

In order to minimize the above cost function, the normal equations are

$$\hat{\mathbf{R}}_m(n-1)\mathbf{a}_m(n) = -\hat{\mathbf{r}}_m^f(n)$$

and the least square error (LSE) is

$$E_m^f(n) = E_x(n) + \mathbf{a}_m^H(n)\hat{\mathbf{r}}_m^f(n)$$

Similarly for the BLP, the error is

$$\varepsilon_m^b(n) = x(n-m) + \mathbf{b}_m^H(n)\mathbf{x}_m(n)$$

and the cost function is

$$E_m^b(n) = \sum_{j=0}^n \lambda^{n-j} |\varepsilon_m^b(j)|^2$$

$$\begin{aligned} E_m^b(n) &= \sum_{j=0}^n \lambda^{n-j} |x(n-m) + \mathbf{b}_m^H(n)\mathbf{x}_m(n)|^2 \\ &= \sum_{j=0}^n \lambda^{n-j} x(n-m)x^*(n-m) + \sum_{j=0}^n \lambda^{n-j} \mathbf{b}_m^H(n)\mathbf{x}_m(n)x^*(n-m) \\ &\quad + \sum_{j=0}^n \lambda^{n-j} x(n-m)\mathbf{x}_m^H(n)\mathbf{b}_m(n) + \sum_{j=0}^n \lambda^{n-j} \mathbf{b}_m^H(n)\mathbf{x}_m(n)\mathbf{x}_m^H(n)\mathbf{b}_m(n) \\ &= E_x(n-m) + \mathbf{b}_m^H(n)\hat{\mathbf{r}}_m^b(n) + \hat{\mathbf{r}}_m^{bH}(n)\mathbf{b}_m(n) + \mathbf{b}_m^H(n)\hat{\mathbf{R}}_m(n)\mathbf{b}_m(n) \end{aligned}$$

In order to minimize the above cost function, the normal equations are

$$\hat{\mathbf{R}}_m(n)\mathbf{b}_m(n) = -\hat{\mathbf{r}}_m^b(n)$$

and the least square error (LSE) is

$$E_m^b(n) = E_x(n-m) + \mathbf{b}_m^H(n)\hat{\mathbf{r}}_m^b(n)$$

- 10.29** Derive the FLP and BLP a priori and a posteriori updating formulas given in Table 10.12.

To be completed.

- 10.30** Modify Table 10.14 for the FAEST algorithm, to obtain a table for the FTF algorithm, and write a MATLAB function for its implementation. Test the obtained function, using the equalization experiment in Example 10.5.2.

To obtain the FTF algorithm, we replace $\bar{\alpha}_m(n)$ in Table 10.14 by $1/\alpha_m(n)$ and Equation (h) with

$$\alpha_{m+1}(n) = \alpha_m(n-1) - \frac{|\varepsilon_m^f(n)|^2}{E_m^f(n)}$$

and Equation (i) with

$$\alpha_m(n) = \frac{\alpha_{m+1}(n)}{1 - \alpha_{m+1}(n)\bar{g}_{m+1}^{(m+1)}(n)e_m^{b*}(n)}$$

The Matlab script is shown below and the learning curve in Figure 10.30.

```
%% Problem 10.30
%%
%%
%%
%%

close all
clear

W=2.9;
N=1000;
Nmc=5;
M=11;
lambda=0.99;
varv=0.001;
h=zeros(3,1);
er=zeros(N,Nmc);
e2a = zeros(1,N);

h(1)=0.5*(1+cos(2*pi*(1-2)/W));
h(2)=0.5*(1+cos(2*pi*(2-2)/W));
h(3)=0.5*(1+cos(2*pi*(3-2)/W));

% Learning curves

hc=[0 h(1) h(2) h(3)]';
n0=7;

t=(1:N)';
for i=1:Nmc

y=sign(rand(N,1)-0.5);
v=sqrt(varv)*randn(N,1);
x=filter(hc,1,y)+v;
yd=zeros(N,1);
yd(n0:N)=y(1:N-n0+1);

% Time Initialization
aold=zeros(M,1); bold=zeros(M,1); cold=zeros(M,1); xold=zeros(M,1);
alphaold=1; gbarold=zeros(M,1);
Efold=10^3; Ebold=10^3;
xv=zeros(M,1);

for n=1:N

% Order Initialization

%ef(1)=x(n);
```

```

%eb(1)=x(n);
%e(1)=yd(n);
%alpha(1)=1;
%alphaold=alpha(1);

% Gain and predictor update
ef(n)=x(n) + aold'*xold;
epsilonf(n) = ef(n)*alphaold;
a = aold - gbarold.*conj(epsilonf(n));
Ef(n) = lambda*Efold + epsilonf(n)*conj(ef(n));
gM1 = [0; gbarold] + (ef(n)/(lambda*Efold))*[1; aold];
eb(n) = lambda*Ebold*gM1(end);
gbar = gM1(1:end-1) - gM1(end).*bold;
alphaM1 = alphaold - abs(epsilonf(n))^2/Ef(n);
alpha = alphaM1/(1 - alphaM1*gM1(end)*conj(eb(n)));
epsilonfb(n) = eb(n)*alpha;
b = bold - gbar.*conj(epsilonfb(n));
Eb(n) = lambda*Ebold + epsilonfb(n)*conj(eb(n));

% Filter update
xv=[x(n); xold(1:end-1)];
e(n) = y(n) - cold'*xv;
epsilon(n) = e(n)*alpha;
c = cold + gbar.*conj(epsilon(n));

% Time Update
aold=a; bold=b; cold=c; xold=xv;
alphaold=alpha; gbarold=gbar;
Efold=Ef(n); Ebold=Eb(n);

end
i
e2a=e2a+e.^2;
end

e2a=e2a./Nmc;

t=(1:N)';
%plot(t,Ef,'r',t,Eb,'b');
%text(250,300,'$\leftarrow$ fontsize{9pt} $E^f$');
%text(100,200,'$E^b \rightarrow$ fontsize{9pt} ');

Jmin1=1;

subplot('position',[0.1,0.55,0.85,0.4]);
semilogy(t,e2a,'r'); %, ...
axis([0,N,10^(-4),10^0]);
title('(a) MSE Learning Curve','fontsize',10);

```

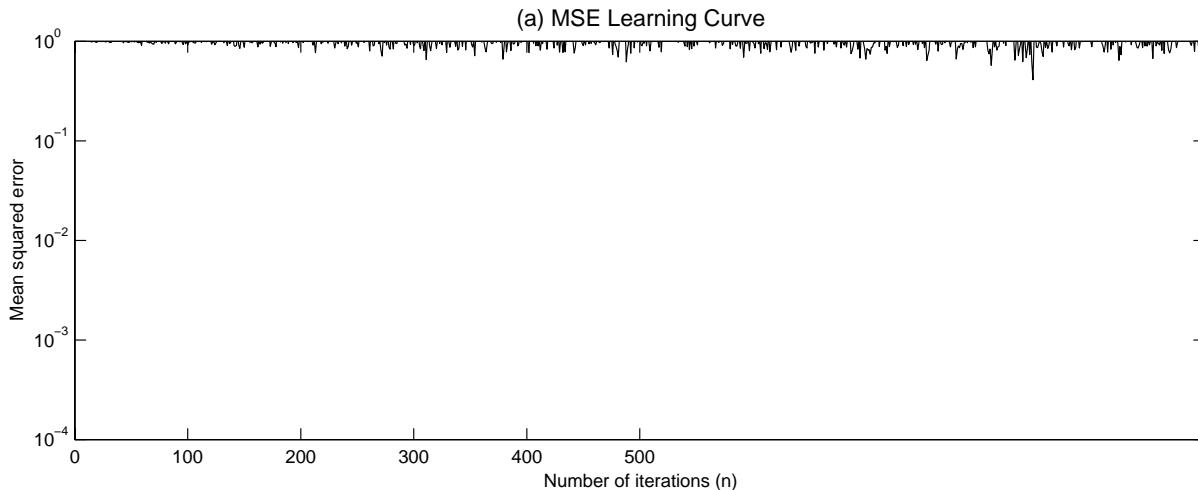


Figure 10.30: Learning curve in Problem P10.30

```

xlabel('Number of iterations (n)', 'fontsize',8);
ylabel('Mean squared error', 'fontsize',8);
set(gca,'xtick',[0:100:500], 'fontsize',8);

print -deps2 P1030.eps

```

- 10.31** If we wish to initialize the fast RLS algorithms (fast Kalman, FAEST, and FTF) using an exact method, we need to collect a set of data $\{\mathbf{x}(n), \mathbf{y}(n)\}_0^{n_0}$ for any $n_0 > M$.

- Identify the quantities needed to start the FAEST algorithm at $n = n_0$. Form the normal equations and use the LDL^H decomposition method to determine these quantities.
- Write a Matlab function `faestexact.m` that implements the FAEST algorithm using the exact initialization procedure described in part (a).
- Use the functions `faest.m` and `faestexact.m` to compare the two different initialization approaches for the FAEST algorithm in the context of the equalization experiment in Example 10.5.2. Use $n_0 = 1.5M$ and $n_0 = 3M$. Which value of δ gives results closest to the exact initialization method?

To be completed.

- 10.32** Using the order-recursive approach introduced in Section 7.3.1, develop an order-recursive algorithm for the solution of the normal equations (10.5.2), and check its validity by using it to initialize the FAEST algorithm, as in Problem 10.31. *Note:* In Section 7.3.1 we could not develop a closed-form algorithm because some recursions required the quantities $\mathbf{b}_m(n-1)$ and $E_m^b(n-1)$. Here we can avoid this problem by using time recursions.

Section 7.3.1 provides an incomplete order recursive algorithm for computing the optimum filter. The algorithm can be completed by using time updatings to determine $\mathbf{b}(n-1)$ and $E_m^b(n-1)$. These can be computed using

time recursions from Table 10.12. The complete algorithm is as follows

Time recursions

$$\begin{aligned}\varepsilon_m^b(n) &= x(n-m) + \mathbf{b}_m^H(n)\mathbf{x}_m(n) \\ \mathbf{b}_m(n) &= \mathbf{b}_m(n-1) - \bar{\mathbf{g}}_m(n)\varepsilon_m^{b*}(n) \\ e_m^b(n) &= \varepsilon_m^b(n)\bar{\alpha}_m(n) \\ E_m^b(n) &= \lambda E_m^b(n-1) + e_m^b(n)\varepsilon_m^{b*}(n)\end{aligned}$$

Order recursions

$$\begin{aligned}\beta_m(n) &= \mathbf{b}_m^H(n-1)\mathbf{r}_m^f(n) + r_{m+1}^f(n) \\ \beta_m^c(n) &= \mathbf{b}_m^H(n)\mathbf{d}_m(n) + d_{m+1}(n) \\ k_m^f(n) &= -\frac{\beta_m(n)}{E_m^b(n-1)} \\ k_m^b(n) &= -\frac{\beta_m^*(n)}{E_m^f(n)} \\ k_m^c(n) &= \frac{\beta_m^c(n)}{E_m^b(n)} \\ \mathbf{a}_{m+1}(n) &= \begin{bmatrix} \mathbf{a}_m(n) \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{b}_m(n-1) \\ 1 \end{bmatrix} k_m^f(n) \\ \mathbf{b}_{m+1}(n) &= \begin{bmatrix} 0 \\ \mathbf{b}_m(n-1) \end{bmatrix} + \begin{bmatrix} 1 \\ \mathbf{a}_m(n) \end{bmatrix} k_m^b(n) \\ \mathbf{c}_{m+1}(n) &= \begin{bmatrix} \mathbf{c}_m(n) \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{b}_m(n) \\ 1 \end{bmatrix} k_m^c(n) \\ E_{m+1}^f(n) &= E_m^f(n) + \beta_m^*(n)k_m^f(n) \\ E_{m+1}^b(n) &= E_m^b(n-1) + \beta_m(n)k_m^b(n) \\ \bar{\mathbf{g}}_{m+1}(n) &= \begin{bmatrix} \bar{\mathbf{g}}_m(n) \\ 0 \end{bmatrix} + \frac{e_m^b(n)}{\lambda E_m^b(n-1)} \begin{bmatrix} \mathbf{b}_m(n-1) \\ 1 \end{bmatrix} \\ \bar{\alpha}_{m+1}(n) &= \bar{\alpha}_m(n) + \frac{|e_m^b(n)|^2}{\lambda E_m^b(n-1)}\end{aligned}$$

The last two equations are obtained from (10.7.17) and (10.7.24), respectively.

- 10.33** In this problem we discuss several quantities that can serve to warn of ill behavior in fast RLS algorithms for FIR filters.

- (a) Show that the variable

$$\eta_m(n) \triangleq \frac{\alpha_{m+1}(n)}{\alpha_m(n)} = \frac{\lambda E_m^b(n-1)}{E_m^b(n)} = 1 - g_{m+1}^{(m+1)}(n)e_m^{b*}(n)$$

satisfies the condition $0 \leq \eta_m(n) \leq 1$.

Since $0 < \lambda \leq 1$ and $E_m^b(n) \geq 0 \Rightarrow \eta_m(n) \geq 0$. From (10.5.24) we have $0 < \alpha_m(n) \leq 1$ and from Table 10.12(j) equation

$$E_m^b(n) = \lambda E_m^b(n-1) + \alpha_m(n) |e_m^b(n)|^2$$

which implies that $E_m^b(n) \geq \lambda E_m^b(n-1)$. Therefore we can see that $0 \leq \eta_m(n) \leq 1$.

(b) Prove the relations

$$\alpha_m(n) = \lambda^m \frac{\det \hat{\mathbf{R}}_m(n-1)}{\det \hat{\mathbf{R}}_m(n)} \quad E_m^f(n) = \frac{\det \hat{\mathbf{R}}_{m+1}(n)}{\det \hat{\mathbf{R}}_m(n-1)} \quad E_m^b(n) = \frac{\det \hat{\mathbf{R}}_{m+1}(n)}{\det \hat{\mathbf{R}}_m(n)}$$

We have

$$\begin{aligned} \hat{\mathbf{R}}(n) &= \lambda \hat{\mathbf{R}}(n-1) + \mathbf{x}(n)\mathbf{x}^H(n) \\ \lambda \hat{\mathbf{R}}(n-1) &= \hat{\mathbf{R}}(n) - \mathbf{x}(n)\mathbf{x}^H(n) \\ \lambda \hat{\mathbf{R}}^{-1}(n) \hat{\mathbf{R}}(n-1) &= \mathbf{I} - [\hat{\mathbf{R}}^{-1}(n)\mathbf{x}(n)]\mathbf{x}^H(n) \end{aligned}$$

Taking the determinant of both sides gives

$$\begin{aligned} \lambda^M \frac{\det \hat{\mathbf{R}}(n-1)}{\det \hat{\mathbf{R}}(n)} &= \det \left\{ \mathbf{I} - [\hat{\mathbf{R}}^{-1}(n)\mathbf{x}(n)]\mathbf{x}^H(n) \right\} \\ &= \det \left[1 - \mathbf{x}^H(n)\hat{\mathbf{R}}^{-1}(n)\mathbf{x}(n) \right] = \alpha(n) \end{aligned}$$

Using the identities

$$\begin{aligned} \det \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} &= \det A_{11} \det (A_{22} - A_{21}A_{11}^{-1}A_{12}) \\ &= \det A_{22} \det (A_{11} - A_{12}A_{22}^{-1}A_{21}) \end{aligned}$$

in conjunction with (10.7.2) and (10.7.3) we can easily obtain the desired results.

(c) Show that

$$\alpha_m(n) = \lambda^m \frac{E_m^b(n)}{E_m^f(n)}$$

and use it to explain why the quantity $\eta_m^\alpha(n) = E_m^f(n) - \lambda^m E_m^b(n)$ can be used as a warning variable.

We have

$$\alpha_m(n) = \lambda^m \frac{\det \hat{\mathbf{R}}_m(n-1)}{\det \hat{\mathbf{R}}_m(n)} = \lambda^m \frac{\frac{\det \hat{\mathbf{R}}_{m+1}(n)}{E_m^f(n)}}{\frac{\det \hat{\mathbf{R}}_{m+1}(n)}{E_m^b(n)}} = \lambda^m \frac{E_m^b(n)}{E_m^f(n)}$$

Hence

$$\eta_m^\alpha(n) = E_m^f(n) - \alpha_m(n)E_m^f(n) = [1 - \alpha_m(n)]E_m^f(n)$$

and since $0 \leq \alpha_m(n) \leq 1$ we should have, in theory, that $\eta_m^\alpha(n) \geq 0$. Therefore, we can monitor $\eta_m^\alpha(n)$ and issue a warning when it becomes negative.

(d) Explain how the quantities

$$\eta_{\bar{g}}(n) \triangleq \bar{g}_{M+1}^{(M+1)}(n) - \frac{e^b(n)}{\lambda E^b(n-1)}, \text{ and}$$

$$\eta_b(n) \triangleq e^b(n) - \lambda E^b(n-1) \bar{g}_{M+1}^{(M+1)}(n)$$

can be used as warning variables.

From (10.7.19), $\eta_{\bar{g}}(n)$ should theoretically be zero. From (10.7.20), $\eta_b(n)$ should theoretically be zero. Therefore, the deviation of these quantities from zero can be used as a warning indicator.

- 10.34** When the desired response is $y(j) = \delta(j - k)$, that is, a spike at $j = k$, $0 \leq k \leq n$, the LS filter $\mathbf{c}_m^{(k)}$ is known as a *spiking filter* or as an LS inverse filter (see Section 8.3).

NOTE: The subscript m is not necessary and it is dropped for convenience.

- (a) Determine the normal equations and the LSE $E_m^{(k)}(n)$ for the LS filter $\mathbf{c}_m^{(k)}$.

When $y(j) = \delta(j - k)$, $0 \leq k \leq n$ we have

$$\begin{aligned}\hat{\mathbf{R}}(n) &= \sum_{j=0}^n \lambda^{n-j} \mathbf{x}(j) \mathbf{x}^H(j) \\ \hat{\mathbf{d}}^{(k)}(n) &= \sum_{j=0}^n \lambda^{n-j} \mathbf{x}(j) \delta(j - k) = \lambda^{n-k} \mathbf{x}(k)\end{aligned}$$

Hence, we have

$$\begin{aligned}\hat{\mathbf{R}}(n) \mathbf{c}^{(k)}(n) &= \lambda^{n-k} \mathbf{x}(k) \\ E^{(k)}(n) &= \lambda^{n-k} - \lambda^{n-k} \mathbf{x}^H(k) \hat{\mathbf{R}}^{-1}(n) \mathbf{x}(k) \lambda^{n-k}\end{aligned}$$

for the normal equations and the LSE.

- (b) Show that $\mathbf{c}_m^{(n)} = \mathbf{g}_m(n)$ and $E_m^{(n)}(n) = \alpha_m(n)$ and explain their meanings.

For $k = n$ we obtain

$$\begin{aligned}\hat{\mathbf{R}}(n) \mathbf{c}^{(n)}(n) &= \mathbf{x}(k) \Rightarrow \mathbf{c}^{(n)}(n) = \mathbf{g}(n) \text{ (adaptation gain)} \\ E^{(n)}(n) &= 1 - \mathbf{x}^H(n) \hat{\mathbf{R}}^{-1}(n) \mathbf{x}(n) = \alpha(n) \text{ (angle variable)}\end{aligned}$$

Therefore, the adaptation gain acts like a spiking filter (see Section 8.3).

The LS filter can be expressed as

$$\mathbf{c}(n) = \hat{\mathbf{R}}^{-1}(n) \sum_{j=0}^n \lambda^{n-j} \mathbf{x}(j) y^*(j) = \sum_{j=0}^n \left[\lambda^{n-j} \hat{\mathbf{R}}^{-1}(n) \mathbf{x}(j) \right] y^*(j) = \sum_{j=0}^n \mathbf{c}^{(j)}(n) y^*(j)$$

that is, the LS filter is a linear combination of spiking filters weighted by each individual sample of the desired response.

- (c) Use the interpretation $\alpha_m(n) = E_m^{(n)}(n)$ to show that $0 \leq \alpha_m(n) \leq 1$.

The worst spiking filter performance corresponds to $\mathbf{c}^{(n)}(n) = \mathbf{0}$ which implies $E^{(n)}(n) = 1$. Hence, $0 \leq E^{(n)}(n) = \alpha(n) \leq 1$.

- (d) Show that $\mathbf{a}_m(n) = \sum_{k=0}^n \mathbf{c}_m^{(k)}(n-1) x(k)$ and explain its meaning.

We have

$$\mathbf{a}(n) = -\hat{\mathbf{R}}^{-1}(n) \sum_{j=0}^n \lambda^{n-j} \mathbf{x}(j-1) x^*(j) = -\sum_{j=0}^n \mathbf{c}^{(j)}(n-1) x^*(j)$$

therefore the FLP is a linear combination of delayed, by one sampling interval, spiking filters weighted by the signal.

10.35 Derive Equations (10.7.33) through (10.7.35) for the a posteriori LS lattice-ladder structure, shown in Figure 10.38, starting with the partitionings (10.7.1) and the matrix by inversion by partitioning relations (10.7.7) and (10.7.8).

Starting with (10.7.1)

$$\mathbf{x}_{m+1}(n) = \begin{bmatrix} \mathbf{x}_m(n) \\ x(n-m) \end{bmatrix} = \begin{bmatrix} x(n) \\ \mathbf{x}_m(n-1) \end{bmatrix}$$

The FLP error is

$$\varepsilon_{m+1}^f(n) = x(n) + \mathbf{a}_{m+1}^H(n)\mathbf{x}_{m+1}(n-1)$$

Using (7.3.27), the order-recursive lattice-ladder forward error $\varepsilon_{m+1}^f(n)$ is found directly

$$\begin{aligned} \varepsilon_{m+1}^f(n) &= x(n) + \left(\begin{bmatrix} \mathbf{a}_m(n) \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{b}_m(n-1) \\ 1 \end{bmatrix} k_m^f(n) \right)^H \mathbf{x}_{m+1}(n-1) \\ &= x(n) + \mathbf{a}_m^H(n)\mathbf{x}_m(n-1) + [\mathbf{b}_m^H(n-1)\mathbf{x}_m(n-1) + x(n-1-m)]k_m^{f*}(n) \\ &= \varepsilon_m^f(n) + k_m^{f*}(n)\varepsilon_m^b(n-1) \end{aligned}$$

Similarly for the BLP error

$$\varepsilon_{m+1}^b(n) = x(n-m) + \mathbf{b}_{m+1}^H(n)\mathbf{x}_{m+1}(n)$$

Using (7.3.23), the backward error $\varepsilon_{m+1}^b(n)$ is

$$\begin{aligned} \varepsilon_{m+1}^b(n) &= x(n-m) + \left(\begin{bmatrix} 0 \\ \mathbf{b}_m(n-1) \end{bmatrix} + \begin{bmatrix} 1 \\ \mathbf{a}_m(n) \end{bmatrix} k_m^b(n) \right)^H \mathbf{x}_{m+1}(n) \\ &= x(n-m) + \mathbf{b}_m^H(n-1)\mathbf{x}_m(n-1) + [\mathbf{a}_m^H(n)\mathbf{x}_m(n-1) + x(n)]k_m^{b*}(n) \\ &= \varepsilon_m^b(n-1) + k_m^{b*}(n)\varepsilon_m^f(n) \end{aligned}$$

Lastly, for the ladder error

$$\varepsilon_{m+1}(n) = y(n) + \mathbf{c}_{m+1}^H(n)\mathbf{x}_{m+1}(n)$$

Using (7.3.15), the order-recursive ladder error is

$$\begin{aligned} \varepsilon_{m+1}(n) &= y(n) - \left(\begin{bmatrix} \mathbf{c}_m(n) \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{b}_m(n) \\ 1 \end{bmatrix} k_m^c(n) \right)^H \mathbf{x}_{m+1}(n) \\ &= y(n) - \mathbf{c}_m^H(n)\mathbf{x}_m(n) + [\mathbf{b}_m^H(n)\mathbf{x}_m(n) + x(n-m)]k_m^{c*}(n) \\ &= \varepsilon_m(n) - k_m^{c*}(n)\varepsilon_m^b(n) \end{aligned}$$

10.36 Prove relations (10.7.45) and (10.7.46) for the updating of the ladder partial correlation coefficient $\beta_m^c(n)$.

Starting with (10.7.40), and using time updating formulas

$$\begin{aligned} \beta_m^c(n) &= \mathbf{b}_m^H(n)\hat{\mathbf{d}}_m(n) + \hat{d}_{m+1}(n) \\ &= \mathbf{b}_m^H(n)[\lambda\hat{\mathbf{d}}_m(n-1) + \mathbf{x}(n)y^*(n)] + [\lambda\hat{d}_{m+1}(n-1) + x(n-m)y^*(n)] \\ &= \lambda\mathbf{b}_m^H(n)\hat{\mathbf{d}}_m(n-1) + \varepsilon_m^b(n)y^*(n) + \lambda\hat{d}_{m+1}(n-1) \\ &= \lambda[\mathbf{b}_m^H(n-1) - \varepsilon_m^b(n)\bar{\mathbf{g}}_m(n)]\hat{\mathbf{d}}_m(n-1) + \lambda\hat{d}_{m+1}(n-1) + \varepsilon_m^b(n)y^*(n) \\ &= \lambda[\mathbf{b}_m^H(n-1)\hat{\mathbf{d}}_m(n-1) + \hat{d}_{m+1}(n-1)] + \varepsilon_m^b(n)[y^*(n) - \lambda\bar{\mathbf{g}}_m(n)\hat{\mathbf{d}}_m(n-1)] \\ &= \lambda\beta_m^c(n-1) + \varepsilon_m^b(n)[y^*(n) - \lambda\bar{\mathbf{g}}_m(n)\hat{\mathbf{R}}_m(n-1)\mathbf{c}_m(n-1)] \\ &= \lambda\beta_m^c(n-1) + \varepsilon_m^b(n)[y^*(n) - \mathbf{x}_m^H(n)\mathbf{c}_m(n-1)] \\ &= \lambda\beta_m^c(n-1) + \varepsilon_m^b(n)e_m^*(n) \end{aligned}$$

10.37 In Section 7.3.1 we derived order-recursive relations for the FLP, BLP, and FIR filtering MMSEs.

- (a) Following the derivation of (7.3.36) and (7.3.37), derive similar order-recursive relations for $E_m^f(n)$ and $E_m^b(n)$.

$$\begin{aligned} E_{m+1}^f(n) &= E_x(n) + \hat{\mathbf{r}}_{m+1}^{fH}(n)\mathbf{a}_{m+1}(n) \\ &= E_x(n) + [\hat{\mathbf{r}}_m^{fH}(n) \quad \hat{r}_{m+1}^{f*}(n)] \left(\begin{bmatrix} \mathbf{a}_m(n) \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{b}_m(n-1) \\ 1 \end{bmatrix} k_m^f(n) \right) \\ &= E_x(n) + \hat{\mathbf{r}}_m^{fH}(n)\mathbf{a}_m(n) + [\hat{\mathbf{r}}_m^{fH}(n)\mathbf{b}_m(n-1) + \hat{r}_{m+1}^{f*}(n)]k_m^f(n) \\ &= E_m^f(n) + \beta_m^*(n)k_m^f(n) \end{aligned}$$

$$\begin{aligned} E_{m+1}^b(n) &= E_x(n-m-1) + \hat{\mathbf{r}}_{m+1}^{bH}(n)\mathbf{b}_{m+1}(n) \\ &= E_x(n-m-1) + [\hat{r}_{m+1}^{b*}(n) \quad \hat{\mathbf{r}}_m^{bH}(n-1)] \left(\begin{bmatrix} 0 \\ \mathbf{b}_m(n-1) \end{bmatrix} + \begin{bmatrix} 1 \\ \mathbf{a}_m(n) \end{bmatrix} k_m^b(n) \right) \\ &= E_x(n-m-1) + \hat{\mathbf{r}}_m^{bH}(n-1)\mathbf{b}_m(n-1) + [\hat{\mathbf{r}}_m^{bH}(n-1)\mathbf{a}_m(n) + \hat{r}_{m+1}^{b*}(n)]k_m^b(n) \\ &= E_m^b(n-1) + \beta_m^*(n)k_m^b(n) \end{aligned}$$

- (b) Show that we can obtain a complete LS lattice-ladder algorithm by replacing, in Table 10.15, the time-recursive updatings of $E_m^f(n)$ and $E_m^b(n)$ with the obtained order-recursive relations.

This is the a posteriori LS lattice-ladder algorithm, which is shown in Table 10.15. Replacing the following steps in Table 10.15 will produce an order-recursive relation

$$\begin{aligned} (b) \quad \beta_{m+1}(n) &= \lambda\beta_{m+1}(n-1) + \frac{\varepsilon_m^b(n-1)\varepsilon_m^{f*}(n)}{\alpha_m(n-1)} \\ (c) \quad E_{m+1}^f(n) &= E_m^f(n) + \beta_m^*(n)k_m^f(n) \\ (d) \quad E_{m+1}^b(n) &= E_m^b(n-1) + \beta_m^*(n)k_m^b(n) \\ (e) \quad k_{m+1}^f(n) &= \frac{-\beta_{m+1}(n)}{E_{m+1}^b(n-1)} \\ (f) \quad k_{m+1}^b(n) &= \frac{-\beta_{m+1}^*(n)}{E_{m+1}^f(n)} \end{aligned}$$

- (c) Write a Matlab function for this algorithm, and verify it by using the equalization experiment in Example 10.5.2.

10.38 Derive the equations for the a priori RLS lattice-ladder algorithm given in Table 10.16, and write a MATLAB function for its implementation. Test the function by using the equalization experiment in Example 10.5.2.

Using (10.7.1), (7.3.27) and (e) from Table 10.12, the a priori FLP error is

$$\begin{aligned} e_{m+1}^f(n) &= x(n) + \mathbf{a}_{m+1}^H(n-1)\mathbf{x}_{m+1}(n-1) \\ &= x(n) + \left(\begin{bmatrix} \mathbf{a}_m(n-1) \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{b}_m(n-1) \\ 1 \end{bmatrix} k_m^f(n-1) \right)^H \begin{bmatrix} \mathbf{x}_m(n-1) \\ x(n-m-1) \end{bmatrix} \\ &= x(n) + \mathbf{a}_m^H(n-1)\mathbf{x}_m(n-1) + k_m^{f*}(n-1)(\mathbf{b}_{m-1}(n-1)\mathbf{x}_m(n-1) + x(n-m-1)) \\ &= e_m^f(n) + k_m^{f*}(n-1)(\mathbf{b}_{m-1}(n-1)\mathbf{x}_m(n-1) + x(n-m-1)) \\ &= e_m^f(n) + k_m^{f*}(n-1)e_m^b(n-1) \end{aligned}$$

Similarly for the a priori BLP error

$$\begin{aligned}
 e_{m+1}^b(n) &= x(n-m-1) + \mathbf{b}_{m+1}^H(n-1)\mathbf{x}_{m+1}(n) \\
 &= x(n-m-1) + \left(\begin{bmatrix} 0 \\ \mathbf{b}_m(n-2) \end{bmatrix} + \begin{bmatrix} 1 \\ \mathbf{a}_m(n-1) \end{bmatrix} k_m^b(n-1) \right)^H \begin{bmatrix} x(n) \\ \mathbf{x}_m(n-1) \end{bmatrix} \\
 &= x(n-m-1) + \mathbf{b}_m^H(n-2)\mathbf{x}_m(n-1) + k_m^{b*}(n-1)(\mathbf{a}_m(n-1)\mathbf{x}_m(n-1) + x(n)) \\
 &= e_m^b(n-1) + k_m^{b*}(n-1)(x(n) + \mathbf{a}_m^H(n-1)\mathbf{x}_m(n-1)) \\
 &= e_m^b(n-1) + k_m^{b*}(n-1)e_m^f(n)
 \end{aligned}$$

The Matlab function is given below and the MSE learning curve is shown in Figure 10.38.

```

%% Problem 10.38
%% Implement table 10.16%%
%%
%%
%%
%%
%%
%
close all
clear all

W=2.9;
N=1000;
Nmc=250;
M=11;
lambda=0.99;
varv=0.001;
h=zeros(3,1);
er=zeros(N,Nmc);

h(1)=0.5*(1+cos(2*pi*(1-2)/W));
h(2)=0.5*(1+cos(2*pi*(2-2)/W));
h(3)=0.5*(1+cos(2*pi*(3-2)/W));

% Learning curves

hc=[0 h(1) h(2) h(3)]';
n0=7;

t=(1:N)';
for i=1:Nmc

y=sign(rand(N,1)-0.5);
v=sqrt(varv)*randn(N,1);
x=filter(hc,1,y)+v;
yd=zeros(N,1);
yd(n0:N)=y(1:N-n0+1);

```

```

% Time Initialization

Ef=ones(M+1,1)*10^(3);
Eb=Ef;
Ebold=Ef; Efold=Ef;
ef=zeros(M+1,1); eb=zeros(M+1,1);
ebold=eb;
kf=zeros(M,1); kb=zeros(M,1);
kfold=kf; kbold=kb;
b=zeros(M+1,1); bc=zeros(M+1,1);
bold=b; bcold=bc;
e=zeros(M+1,1); kc=zeros(M,1);
kcold=kc;
alpha=zeros(M+1,1);
alphaold=alpha;
for n=1:N

    % Order Initialization

    ef(1)=x(n);
    eb(1)=x(n);
    e(1)=yd(n);
    alpha(1)=1;
    alphaold(1)=alpha(1);

    for m=1:M
        %Lattice
        ef(m+1) = ef(m) + conj(kfold(m))*ebold(m);
        eb(m+1) = ebold(m) + conj(kbold(m))*ef(m);
        %b(m) = lambda*bold(m) + (ebold(m)*conj(ef(m)))*alphaold(m);
        Ef(m) = lambda*Efold(m) + alphaold(m)*conj(ef(m))*ef(m);
        Eb(m) = lambda*Ebold(m) + alpha(m)*conj(eb(m))*eb(m);
        kf(m) = kfold(m) - (alphaold(m)*ebold(m)*conj(ef(m+1)))/Ebold(m);
        kb(m) = kbold(m) - (alphaold(m)*ef(m)*conj(eb(m+1)))/Ef(m);
        alpha(m+1) = alpha(m) - abs(alpha(m)*eb(m))*abs(alpha(m)*eb(m))/Eb(m);

        % Ladder
        %bc(m)=lambda*bcold(m) + (eb(m)*conj(ef(m)))*alpha(m);
        e(m+1)=e(m)-conj(kcold(m))*eb(m);
        kc(m)= kcold(m)+(alpha(m)*eb(m)*conj(e(m+1)))/Eb(m);
    end

    er(n,i)=e(M+1); erf(n,i)=ef(M); erb(n,i)=eb(M);
    Erf(n,i)=Eb(1); Erb(n,i)=Eb(M); al(n,i)=alpha(M);

    % Time Delay
    Ebold=Eb; Efold=Ef;
    ebold=eb; efold=ef;

```

```

bold=b; bcold=bc;
kfold=kf; kbold=kb; kcold=kc;
alphaold=alpha;
end
i
end
er2=er.^2;
er2m=mean(er2,2);

t=(1:N)';
%plot(t,Erf,'r',t,Erb,'b');
%text(250,300,'leftarrow\fontsize{9pt} E^f');
%text(100,200,'E^b \rightarrow\fontsize{9pt} ');
Jmin1=1;

%subplot('position',[0.1,0.55,0.85,0.4]);
semilogy(t,er2m,'r');
axis([0,N,10^(-4), 10^0]);
title(['MSE Learning Curve (\lambda=.99, W=' num2str(W) ')'], 'fontsize',10);
xlabel('Number of iterations (n)', 'fontsize',8);
ylabel('Mean squared error', 'fontsize',8);
set(gca,'xtick',[0:100:N], 'fontsize',8);

print -deps2 P1038.eps

```

- 10.39** Derive the equations for the a priori RLS lattice-ladder algorithm with error feedback (see Table 10.7), and write a Matlab function for its implementation. Test the function by using the equalization experiment in Example 10.5.2.

$$\begin{aligned}
k_m^c(n) &= \frac{\beta_m^c(n)}{E_m^b(n)} = \lambda \frac{\beta_m^c(n-1)}{E_m^b(n)} + \frac{\alpha_m e_m^b(n) e_m^*(n)}{E_m^b(n)} \\
&= \lambda \frac{\beta_m^c(n-1)}{E_m^b(n-1)} \frac{E_m^b(n-1)}{E_m^b(n)} + \frac{\alpha_m e_m^b(n) e_m^*(n)}{E_m^b(n)}
\end{aligned}$$

where $k_m^c(n-1) = \frac{\beta_m^c(n-1)}{E_m^b(n-1)}$. Now

$$\lambda E_m^b(n-1) = E_m^b(n) - \alpha_m e_m^b(n) e_m^{b*}(n)$$

so, the ladder parameter can be simplified to

$$\begin{aligned}
k_m^c(n) &= \frac{k_m^c(n-1)}{E_m^b(n)} [E_m^b(n) - \alpha_m e_m^b(n) e_m^{b*}(n)] + \frac{\alpha_m e_m^b(n) e_m^*(n)}{E_m^b(n)} \\
&= k_m^c(n-1) - k_m^c(n-1) \frac{\alpha_m e_m^b(n) e_m^{b*}(n)}{E_m^b(n)} + \frac{\alpha_m e_m^b(n) e_m^*(n)}{E_m^b(n)} \\
&= k_m^c(n-1) + \frac{\alpha_m e_m^b(n)}{E_m^b(n)} [e_m^*(n) - k_m^c(n-1) e_m^{b*}(n)] \\
&= k_m^c(n-1) + \frac{\alpha_m e_m^b(n) e_{m+1}^*(n)}{E_m^b(n)}
\end{aligned}$$

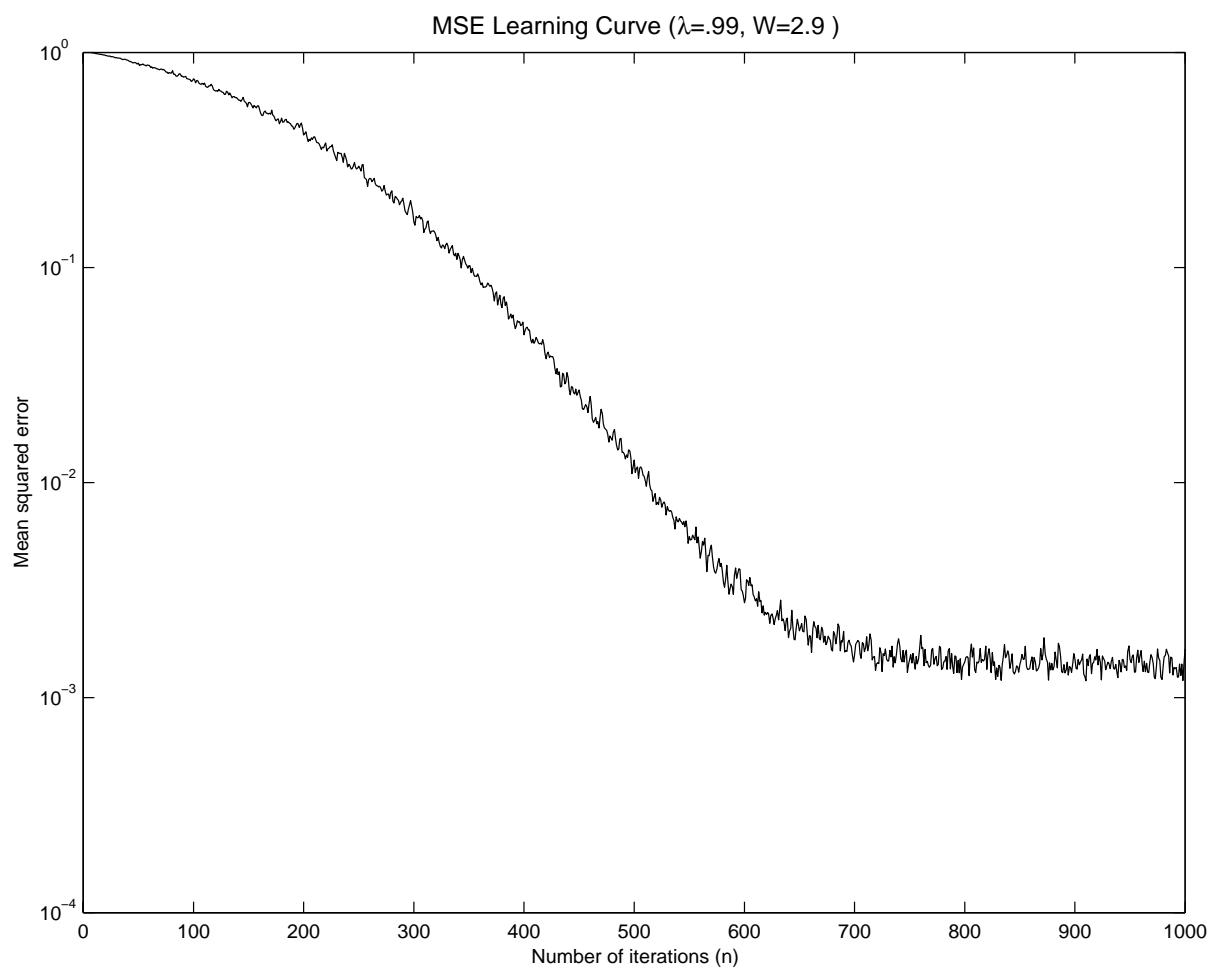


Figure 10.38: MSE learning curve in P10.38

Similarly for the lattice parameters $k_m^f(n)$ and $k_m^b(n)$

$$\begin{aligned}
 k_m^f(n) &= -\frac{\beta_m(n)}{E_m^b(n-1)} \\
 &= -\frac{\lambda\beta_m(n-1) + \alpha_m(n-1)e_m^b(n-1)e_m^{f*}(n)}{E_m^b(n-1)} \\
 &= -\lambda\frac{\beta_m(n-1)}{E_m^b(n-2)}\frac{E_m^b(n-2)}{E_m^b(n-1)} - \frac{\alpha_m(n-1)e_m^b(n-1)e_m^{f*}(n)}{E_m^b(n-1)} \\
 &= \frac{k_m^f(n-1)}{E_m^b(n-1)}[E_m^b(n-1) - \alpha_m(n-1)e_m^b(n-1)e_m^{b*}(n)] - \frac{\alpha_m(n-1)e_m^b(n-1)e_m^{f*}(n)}{E_m^b(n-1)} \\
 &= k_m^f(n-1) - \frac{\alpha_m(n-1)e_m^b(n-1)}{E_m^b(n-1)}[e_m^f(n) + k_m^f(n-1)e_m^{b*}(n-1)] \\
 &= k_m^f(n-1) - \frac{\alpha_m(n-1)e_m^b(n-1)e_{m+1}^{f*}(n)}{E_m^b(n-1)}
 \end{aligned}$$

and

$$\begin{aligned}
 k_m^b(n) &= -\frac{\beta_m^*(n)}{E_m^f(n)} \\
 &= -\frac{\lambda\beta_m^*(n-1) + \alpha_m(n-1)e_m^b(n-1)e_m^{f*}(n)}{E_m^f(n)} \\
 &= -\lambda\frac{\beta_m^*(n-1)}{E_m^f(n-1)}\frac{E_m^f(n-1)}{E_m^f(n)} - \frac{\alpha_m(n-1)e_m^b(n-1)e_m^{f*}(n)}{E_m^f(n)} \\
 &= \frac{k_m^b(n-1)}{E_m^f(n)}[E_m^f(n) - \alpha_m(n-1)e_m^f(n)e_m^{f*}(n)] - \frac{\alpha_m(n-1)e_m^b(n-1)e_m^{f*}(n)}{E_m^f(n)} \\
 &= k_m^b(n-1) - \frac{\alpha_m(n-1)e_m^f(n)}{E_m^f(n)}[e_m^{b*}(n-1) + k_m^b(n-1)e_m^{f*}(n)] \\
 &= k_m^b(n-1) - \frac{\alpha_m(n-1)e_m^b(n-1)e_{m+1}^{b*}(n)}{E_m^f(n)}
 \end{aligned}$$

The Matlab script is given below and the MSE learning curve is shown in Figure 10.39.

```

%% Problem 10.38
%% Implement table 10.16%%
%%
%%
%%
%%

```

```

close all
clear all

W=2.9;
N=1000;
NmC=250;
M=11;

```

```

lambda=0.99;
varv=0.001;
h=zeros(3,1);
er=zeros(N,Nmc);

h(1)=0.5*(1+cos(2*pi*(1-2)/W));
h(2)=0.5*(1+cos(2*pi*(2-2)/W));
h(3)=0.5*(1+cos(2*pi*(3-2)/W));

% Learning curves

hc=[0 h(1) h(2) h(3)]';
n0=7;

t=(1:N)';
for i=1:Nmc

y=sign(rand(N,1)-0.5);
v=sqrt(varv)*randn(N,1);
x=filter(hc,1,y)+v;
yd=zeros(N,1);
yd(n0:N)=y(1:N-n0+1);

% Time Initialization

Ef=ones(M+1,1)*10^(3);
Eb=Ef;
Ebold=Ef; Efold=Ef;
ef=zeros(M+1,1); eb=zeros(M+1,1);
ebold=eb;
kf=zeros(M,1); kb=zeros(M,1);
kfold=kf; kbold=kb;
b=zeros(M+1,1); bc=zeros(M+1,1);
bold=b; bcold=bc;
e=zeros(M+1,1); kc=zeros(M,1);
kcold=kc;
alpha=zeros(M+1,1);
alphaold=alpha;
for n=1:N

% Order Initialization

ef(1)=x(n);
eb(1)=x(n);
e(1)=yd(n);
alpha(1)=1;
alphaold(1)=alpha(1);

```

```

for m=1:M
    %Lattice
    ef(m+1) = ef(m) + conj(kfold(m))*ebold(m);
    eb(m+1) = ebold(m) + conj(kbold(m))*ef(m);
    b(m) = lambda*bold(m) + (ebold(m)*conj(ef(m)))*alphaold(m);
    Ef(m) = lambda*Efold(m) + alphaold(m)*conj(ef(m))*ef(m);
    Eb(m) = lambda*Ebold(m) + alpha(m)*conj(eb(m))*eb(m);
    kf(m) = -b(m)/Ebold(m);
    kb(m) = -conj(b(m))/Ef(m);
    alpha(m+1) = alpha(m) - abs(eb(m))*abs(eb(m))/Eb(m);

    % Ladder
    bc(m)=lambda*bcold(m) + (eb(m)*conj(ef(m)))*alpha(m);
    e(m+1)=e(m)-conj(kcold(m))*eb(m);
    kc(m)=bc(m)/Eb(m);
end

er(n,i)=e(M+1);erf(n,i)=ef(M);erb(n,i)=eb(M);
Erf(n,i)=Eb(1); Erfb(n,i)=Eb(M); al(n,i)=alpha(M);

% Time Delay
Ebold=Eb; Efold=Ef;
ebold=eb; efold=ef;
bold=b; bcold=bc;
kfold=kf; kbold=kb;
alphaold=alpha;
end
i
end
er2=er.^2;
er2m=mean(er2,2);

t=(1:N)';
%plot(t,Erf,'r',t,Erb,'b');
%text(250,300,'leftarrow\fontsize{9pt} E^f');
%text(100,200,'E^b \rightarrow\fontsize{9pt} ');

Jmin1=1;

%subplot('position',[0.1,0.55,0.85,0.4]);
semilogy(t,er2m,'r');
axis([0,N,10^(-4), 10^0]);
title(['MSE Learning Curve (\lambda=.99, W=' num2str(W) ')'], 'fontsize',10);
xlabel('Number of iterations (n)', 'fontsize',8);
ylabel('Mean squared error', 'fontsize',8);
set(gca,'xtick',[0:100:N], 'fontsize',8);

print -deps2 P1039.eps

```

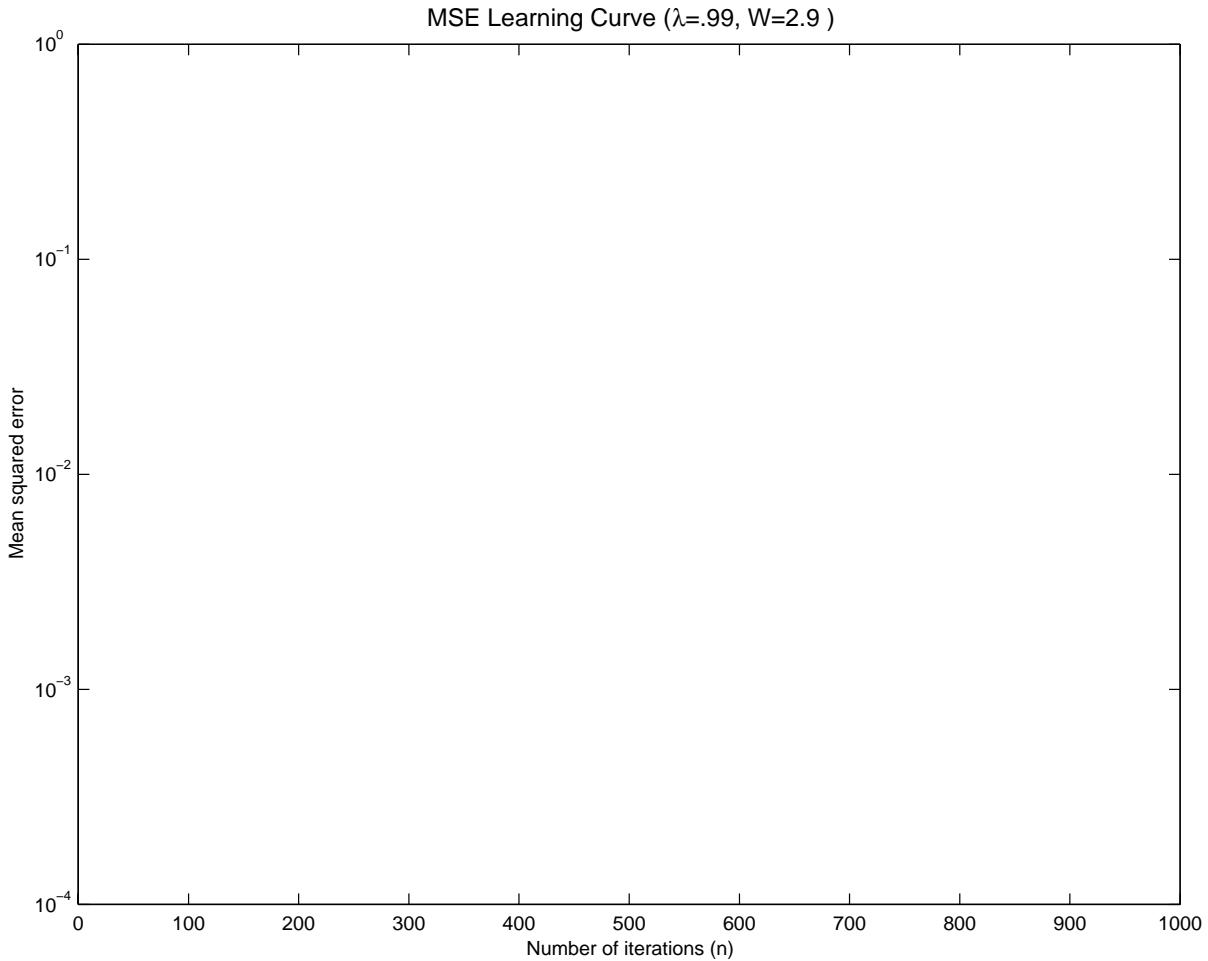


Figure 10.39: MSE learning curve in P10.39

- 10.40** Derive the equations for the a posteriori RLS lattice-ladder algorithm with error feedback (Ling et al. 1986) and write a Matlab function for its implementation. Test the function by using the equalization experiment in Example 10.5.2.

To be completed.

- 10.41** The a posteriori and the a priori RLS lattice-ladder algorithms need the conversion factor $\alpha_m(n)$ because the updating of the quantities $E_m^f(n)$, $E_m^b(n)$, $\beta_m(n)$, and $\beta_m^c(n)$ requires both the a priori and a posteriori errors. Derive a double (a priori and a posteriori) lattice-ladder RLS filter that avoids the use of the conversion factor by updating both the a priori and the a posteriori prediction and filtering errors.

Combining the a posteriori (Table 10.15) and a priori (Table 10.16) lattice-ladder algorithms, and removing dependence on α_m . The lattice parameters are

$$k_m^f(n) = \frac{-\beta_m(n)}{E_m^b(n-1)}$$

and

$$k_m^b(n) = \frac{-\beta_m^*(n)}{E_m^f(n)}$$

where the time updating of $\beta_m(n)$ is

$$\beta_m(n) = \lambda \beta_m(n-1) + \varepsilon_m^b(n-1) e_m^{f*}(n)$$

and the minimum error energies are

$$\begin{aligned} E_m^f(n) &= \lambda E_m^f(n-1) + \varepsilon_m^f(n) e_m^{f*}(n) \\ E_m^b(n) &= \lambda E_m^b(n-1) + \varepsilon_m^b(n) e_m^{b*}(n) \end{aligned}$$

The a priori and a posteriori errors become

$$\begin{aligned} e_{m+1}^f(n) &= e_m^f(n) + k_m^{f*}(n-1) e_m^b(n-1) \\ e_{m+1}^b(n) &= e_m^b(n-1) + k_m^{b*}(n-1) e_m^f(n) \\ \varepsilon_{m+1}^f(n) &= \varepsilon_m^f(n) + k_m^{f*}(n) \varepsilon_m^b(n-1) \\ \varepsilon_{m+1}^b(n) &= \varepsilon_m^b(n-1) + k_m^{b*}(n) \varepsilon_m^f(n) \end{aligned}$$

The ladder parameters are

$$\begin{aligned} k_m^c(n) &= \frac{-\beta_m^c(n)}{E_m^b(n)} \\ \beta_m^c(n) &= \lambda \beta_m^c(n-1) + \varepsilon_m^b(n) e_m^{*}(n) \\ e_{m+1}(n) &= e_m(n) - k_m^{c*}(n-1) e_m^b(n) \\ \varepsilon_{m+1}(n) &= \varepsilon_m(n) - k_m^{c*}(n) \varepsilon_m^b(n) \end{aligned}$$

- 10.42** Program the RLS Givens lattice-ladder filter with square roots (see Table 10.18), and study its use in the adaptive equalization experiment of Example 10.5.2.

The Matlab script is given below and the MSE learning curve is shown in Figure 10.42.

```
%% Problem 10.38
%% Implement table 10.16%%
%%
%%
%%
%%
%%
%
close all
clear all

W=2.9;
N=1000;
NmC=250;
M=11;
lambda=0.99;
varv=0.001;
h=zeros(3,1);
er=zeros(N,NmC);

h(1)=0.5*(1+cos(2*pi*(1-2)/W));
h(2)=0.5*(1+cos(2*pi*(2-2)/W));
h(3)=0.5*(1+cos(2*pi*(3-2)/W));
```

```

% Learning curves

hc=[0 h(1) h(2) h(3)]';
n0=7;

t=(1:N)';
for i=1:Nmc

    y=sign(rand(N,1)-0.5);
    v=sqrt(varv)*randn(N,1);
    x=filter(hc,1,y)+v;
    yd=zeros(N,1);
    yd(n0:N)=y(1:N-n0+1);

    % Time Initialization

    Ef=ones(M+1,1)*10^(3);
    Eb=Ef;
    Ebold=Ef; Efold=Ef;
    ef=zeros(M+1,1); eb=zeros(M+1,1);
    ebold=eb;
    kf=zeros(M,1); kb=zeros(M,1);
    kfold=kf; kbold=kb;
    cf=zeros(M+1,1); sf=zeros(M+1,1);
    cb=zeros(M+1,1); sb=zeros(M+1,1);
    cfold=cf; cbold=cb; sfold=sf; sbold=sb;
    e=zeros(M+1,1); kc=zeros(M,1);
    kcold=kc;
    alpha=zeros(M+1,1);
    alphaold=alpha;

    for n=1:N

        % Order Initialization

        ef(1)=x(n);
        eb(1)=x(n);
        e(i)=yd(n);
        alpha(1)=1;
        alphaold(1)=alpha(1);

        for m=1:M
            %Lattice
            Ef(m) = (lambda*Efold(m)^2 + conj(ef(m))*ef(m))^(1/2);
            cf(m) = lambda^(1/2)*Efold(m)/Ef(m);
            sf(m) = ef(m)/Ef(m);
            Eb(m) = (lambda*Ebold(m)^2 + conj(eb(m))*eb(m))^(1/2);
            cb(m) = lambda^(1/2)*Ebold(m)/Eb(m);

```

```

sb(m) = eb(m)/Eb(m);
ef(m+1) = cbold(m)*ef(m) + lambda^(1/2)*sbold(m)*conj(kfold(m));
kf(m) = lambda^(1/2)*cbold(m)*kfold(m) - sbold(m)*conj(ef(m));
eb(m+1) = cf(m)*ebold(m) + lambda^(1/2)*sf(m)*conj(kbold(m));
kb(m) = lambda^(1/2)*cf(m)*kbold(m) - sf(m)*conj(ebold(m));
alpha(m+1) = alpha(m) - abs(alpha(m)*eb(m))*abs(alpha(m)*eb(m))/Eb(m);

% Ladder
e(m+1) = cb(m)*e(m) - lambda^(1/2)*sb(m)*kcold(m);
kc(m)= lambda^(1/2)*cb(m)*kcold(m) + sb(m)*conj(e(m));
end

er(n,i)=e(M+1);erf(n,i)=ef(M);erb(n,i)=eb(M);
Erf(n,i)=Eb(1); Erb(n,i)=Eb(M); al(n,i)=alpha(M);

% Time Delay
cbold=cb; cfold=cf;
sbold=sb; sfold=sf;
Ebold=Eb; Efold=Ef;
ebold=eb; efold=ef;
kfold=kf; kbold=kb; kcold=kc;
alphaold=alpha;
end
i
end
er2=er.^2;
er2m=mean(er2,2);

t=(1:N)';
%plot(t,Erf,'r',t,Erb,'b');
%text(250,300,' \leftarrow\ font size{9pt} E^f');
%text(100,200,' E^b \rightarrow\ font size{9pt} ');

Jmin1=1;

%subplot('position',[0.1,0.55,0.85,0.4]);
semilogy(t,er2m,'r');
axis([0,N,10^(-4), 10^0]);
title(['MSE Learning Curve (\lambda=.99, W=' num2str(W) ')'], 'font size', 10);
xlabel('Number of iterations (n)', 'font size', 8);
ylabel('Mean squared error', 'font size', 8);
set(gca,'xtick',[0:100:N], 'font size', 8);

print -deps2 P1042.eps

```

- 10.43** Derive the formulas and program the RLS Givens lattice-ladder filter without square roots (see Table 10.18), and study its use in the adaptive equalization experiment of Example 10.5.2.

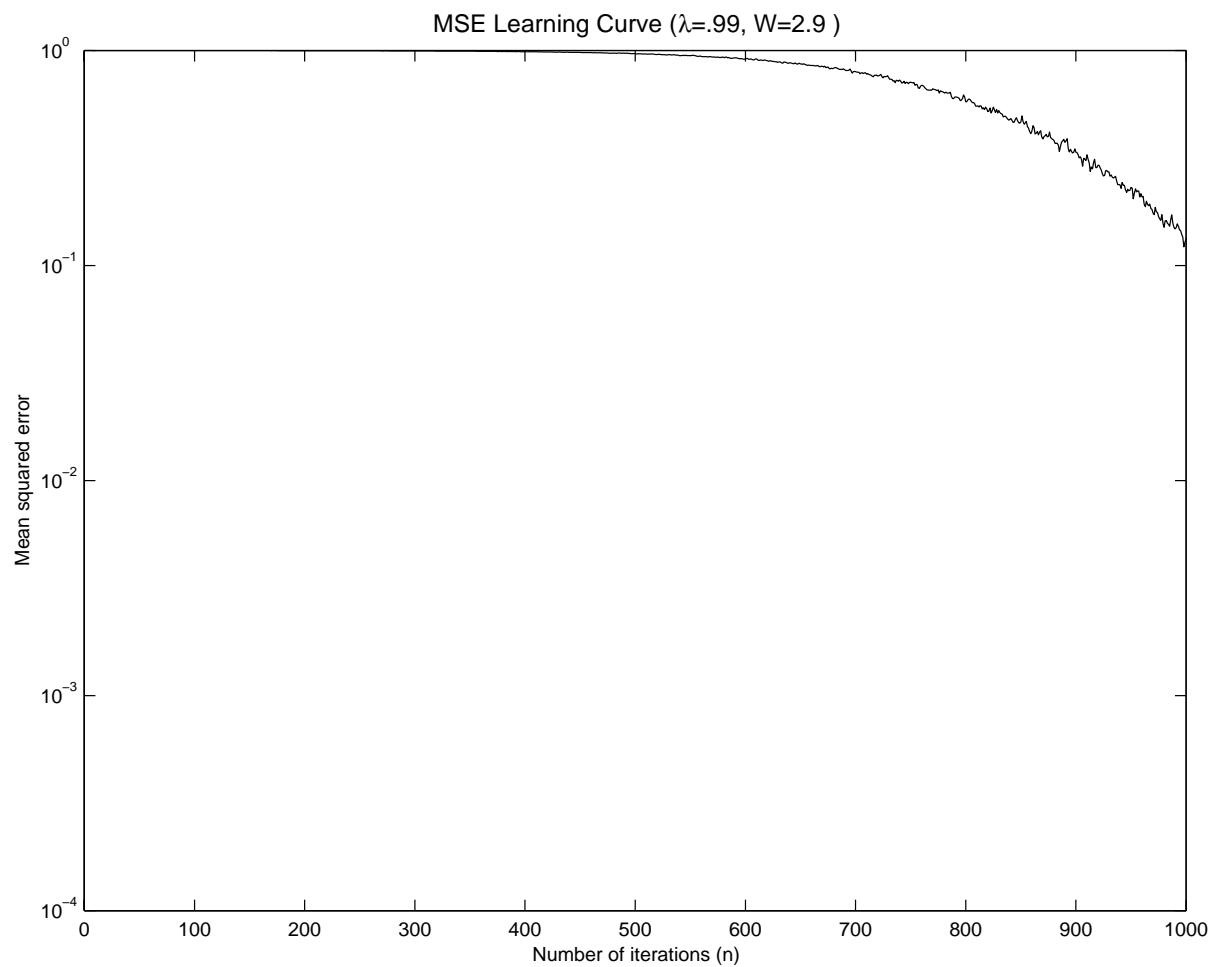


Figure 10.42: MSE learning curve in P10.42

The square root-free version is a modification of the error feedback form of the a priori LS lattice-ladder algorithm. If we define

$$c_m^b(n) = \frac{\lambda E_m^b(n-1)}{E_m^b(n)} = |\tilde{c}_m^b(n)|^2$$

and

$$s_m^b(n) = \frac{\alpha_m(n)e_m^b(n)}{E_m^b(n)}$$

then using (10.7.50), the ladder coefficient is

$$k_m^c(n) = c_m^b(n)k_m^c(n-1) + s_m^b(n)e_m^*(n)$$

Also the lattice parameters are found using the error feedback a priori updating with

$$k_m^f(n) = c_m^b(n-1)k_m^f(n-1) - s_m^b(n-1)e_m^{f*}(n)$$

and

$$k_m^b(n) = c_m^f(n)k_m^b(n-1) - s_m^f(n)e_m^{b*}(n-1)$$

where

$$c_m^f(n) = \frac{\lambda E_m^f(n-1)}{E_m^f(n)} = |\tilde{c}_m^f(n)|^2$$

and

$$s_m^f(n) = \frac{\alpha_m(n-1)e_m^f(n)}{E_m^f(n)}$$

are the forward rotation parameters.

The Matlab script is given below and the MSE learning curve is shown in Figure 10.43.

```
%% Problem 10.38
%% Implement table 10.16%%
%%
%%
%%
%%
%%
```

```
close all
clear all
```

```
W=2.9;
N=1000;
Nmc=250;
M=11;
lambda=0.99;
varv=0.001;
h=zeros(3,1);
```

```

er=zeros(N,Nmc);

h(1)=0.5*(1+cos(2*pi*(1-2)/W));
h(2)=0.5*(1+cos(2*pi*(2-2)/W));
h(3)=0.5*(1+cos(2*pi*(3-2)/W));

% Learning curves

hc=[0 h(1) h(2) h(3)]';
n0=7;

t=(1:N)';
for i=1:Nmc

y=sign(rand(N,1)-0.5);
v=sqrt(varv)*randn(N,1);
x=filter(hc,1,y)+v;
yd=zeros(N,1);
yd(n0:N)=y(1:N-n0+1);

% Time Initialization

Ef=ones(M+1,1)*10^(3);
Eb=Ef;
Ebold=Ef; Efold=Ef;
ef=zeros(M+1,1); eb=zeros(M+1,1);
ebold=eb;
kf=zeros(M,1); kb=zeros(M,1);
kfold=kf; kbold=kb;
cf=zeros(M+1,1); sf=zeros(M+1,1);
cb=zeros(M+1,1); sb=zeros(M+1,1);
cfold=cf; cbold=cb; sfold=sf; sbold=sb;
e=zeros(M+1,1); kc=zeros(M,1);
kcold=kc;
alpha=zeros(M+1,1);
alphaold=alpha;

for n=1:N

% Order Initialization

ef(1)=x(n);
eb(1)=x(n);
e(1)=yd(n);
alpha(1)=1;
alphaold(1)=alpha(1);

for m=1:M

```

```

%Lattice
Ef(m) = lambda*Efold(m) + alphaold(m)*conj(ef(m))*ef(m);
cf(m) = lambda*Efold(m)/Ef(m);
sf(m) = alphaold(m)*ef(m)/Ef(m);
Eb(m) = lambda*Ebold(m) + alpha(m)*conj(eb(m))*eb(m);
cb(m) = lambda*Ebold(m)/Eb(m);
sb(m) = alpha(m)*eb(m)/Eb(m);
ef(m+1) = ef(m) + conj(kfold(m))*ebold(m);
kf(m) = cbold(m)*kfold(m) - sbold(m)*conj(ef(m));
eb(m+1) = ebold(m) + conj(kbold(m))*ef(m);
kb(m) = cf(m)*kbold(m) - sf(m)*conj(ebold(m));
alpha(m+1) = alpha(m) - abs(alpha(m)*eb(m))*abs(alpha(m)*eb(m))/Eb(m);

% Ladder
e(m+1)=e(m)-conj(kcold(m))*eb(m);
kc(m)= cb(m)*kcold(m) + sb(m)*conj(e(m));
end

er(n,i)=e(M+1);erf(n,i)=ef(M);erb(n,i)=eb(M);
Erf(n,i)=Eb(1); Erb(n,i)=Eb(M); al(n,i)=alpha(M);

% Time Delay
cbold=cb; cfold=cf;
sbold=sb; sfold=sf;
Ebold=Eb; Efold=Ef;
ebold=eb; efold=ef;
kfold=kf; kbold=kb; kcold=kc;
alphaold=alpha;
end
i
end

er2=er.^2;
er2m=mean(er2,2);

t=(1:N)';
%plot(t,Erf,'r',t,Erb,'b');
%text(250,300,'leftarrow\fontsize{9pt} E^f');
%text(100,200,'E^b \rightarrow\fontsize{9pt} ');
Jmin1=1;

%subplot('position',[0.1,0.55,0.85,0.4]);
semilogy(t,er2m,'r');
axis([0,N,10^(-4), 10^0]);
title(['MSE Learning Curve (\lambda=.99, W=' num2str(W) ')'],'fontsize',10);
xlabel('Number of iterations (n)', 'fontsize',8);
ylabel('Mean squared error', 'fontsize',8);

```

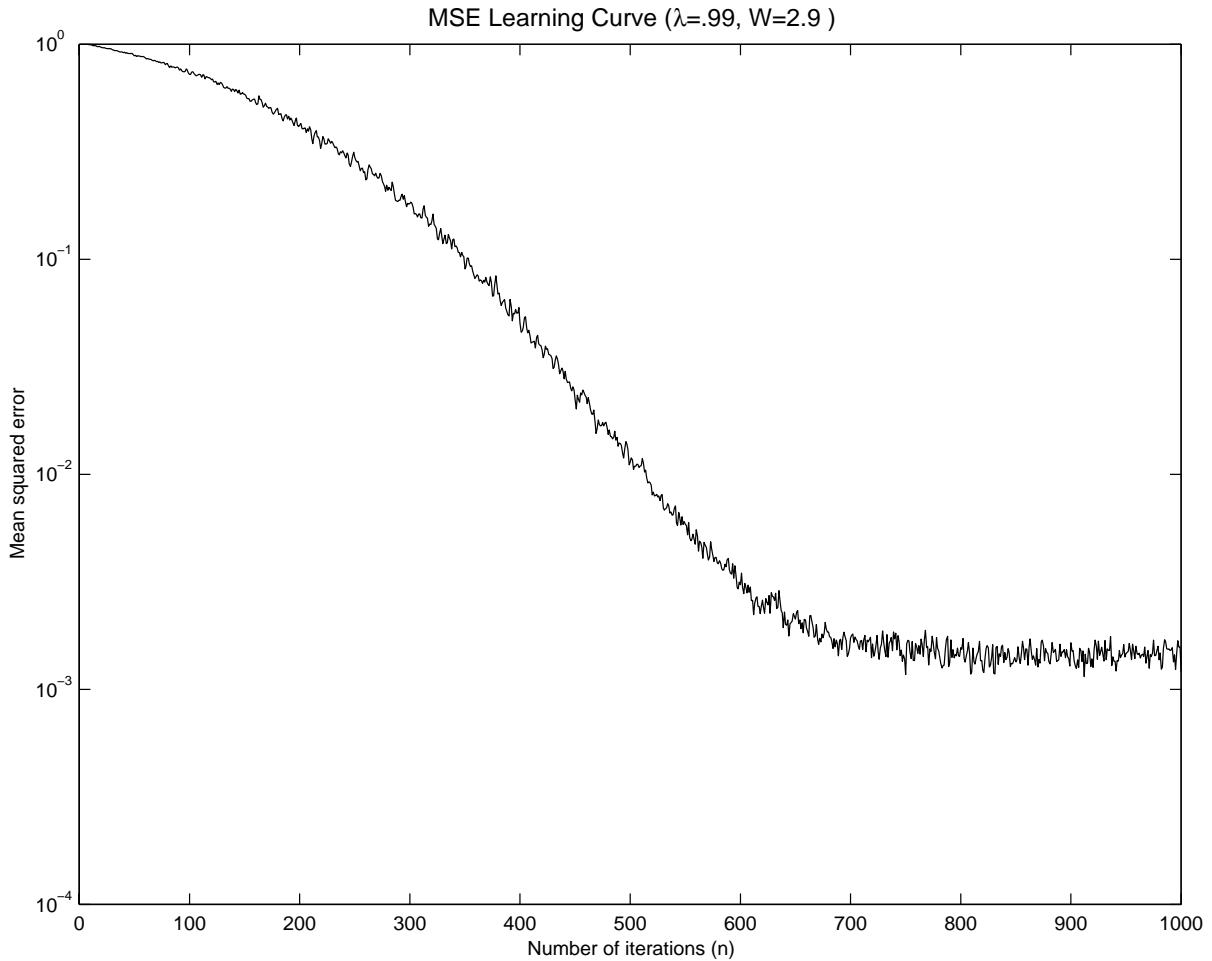


Figure 10.43: MSE learning curve in P10.43

```

set(gca,'xtick',[0:100:N], 'fontsize',8);

print -deps2 P1043.eps

```

- 10.44** In this problem we discuss the derivation of the normalized lattice-ladder RLS algorithm, which uses a smaller number of time and order updating recursions and has better numerical behavior due to the normalization of its variables.

- (a) Define the energy and angle normalized variables

$$\bar{e}_m^f(n) = \frac{\varepsilon_m^f(n)}{\sqrt{\alpha_m(n)}\sqrt{E_m^f(n)}} \quad \bar{e}_m^b(n) = \frac{\varepsilon_m^b(n)}{\sqrt{\alpha_m(n)}\sqrt{E_m^b(n)}} \quad \bar{e}_m(n) = \frac{\varepsilon_m(n)}{\sqrt{\alpha_m(n)}\sqrt{E_m(n)}}$$

$$\bar{k}_m(n) = \frac{\beta_m(n)}{\sqrt{E_m^f(n)}\sqrt{E_m^b(n-1)}} \quad \bar{k}_m^c(n) = \frac{\beta_m^c(n)}{\sqrt{E_m(n)}\sqrt{E_m^b(n)}}$$

and show that the normalized errors and the partial correlation coefficients $\bar{k}_m(n)$ and $\bar{k}_m^c(n)$ have magnitude less than 1.

(b) Derive the following normalized lattice-ladder RLS algorithm:

$$E_0^f(-1) = E_0(-1) = \delta > 0$$

For $n = 0, 1, 2, \dots$

$$E_0^f(n) = \lambda E_0^f(n-1) + |x(n)|^2, \quad E_0(n) = \lambda E_0(n-1) + |y(n)|^2$$

$$\bar{e}_0^f(n) = \bar{e}_0^b(n) = \frac{x(n)}{\sqrt{E_0^f(n)}}, \quad \bar{e}_0(n) = \frac{y(n)}{\sqrt{E_0(n)}}$$

For $m = 0$ to $M-1$

$$\bar{k}_m(n) = \sqrt{1 - |\bar{e}_m^f(n)|^2} \sqrt{1 - |\bar{e}_m^b(n-1)|^2} \bar{k}_m(n-1) + \bar{e}_m^{f*}(n) \bar{e}_m^b(n-1)$$

$$\bar{e}_{m+1}^f(n) = \left(\sqrt{1 - |\bar{e}_m^b(n-1)|^2} \sqrt{1 - |\bar{k}_m(n)|^2} \right)^{-1} [\bar{e}_m^f(n) - \bar{k}_m(n) \bar{e}_m^b(n-1)]$$

$$\bar{e}_{m+1}^b(n) = \left(\sqrt{1 - |\bar{e}_m^f(n)|^2} \sqrt{1 - |\bar{k}_m(n)|^2} \right)^{-1} [\bar{e}_m^b(n-1) - \bar{k}_m(n) \bar{e}_m^f(n)]$$

$$\bar{k}_m^c(n) = \sqrt{1 - |\bar{e}_m(n)|^2} \sqrt{1 - |\bar{e}_m^b(n)|^2} \bar{k}_m^c(n-1) + \bar{e}_m^{*}(n) \bar{e}_m^b(n)$$

$$\bar{e}_{m+1}(n) = \left(\sqrt{1 - |\bar{e}_m^b(n)|^2} \sqrt{1 - |\bar{k}_m^c(n)|^2} \right)^{-1} [\bar{e}_m(n) - \bar{k}_m^c(n) \bar{e}_m^b(n)]$$

(c) Write a Matlab function to implement the derived algorithm, and test its validity by using the equalization experiment in Example 10.5.2.

To be completed.

10.45 Prove (10.4.46) by direct manipulation of (10.6.35).

From (10.6.35) follows that the last diagonal element of $\mathbf{Q}(n)$ is equal to $\tilde{\alpha}(n)$. Therefore, using the partitioning

$$\mathbf{Q}(n) = \begin{bmatrix} \tilde{\mathbf{Q}}(n) & \mathbf{q}(n) \\ \mathbf{q}^H(n) & \tilde{\alpha}(n) \end{bmatrix}$$

and (10.6.35) we obtain

$$\begin{aligned} \mathbf{q}^H(n) \sqrt{\lambda} \tilde{\mathbf{R}}(n-1) + \tilde{\alpha}(n) \mathbf{x}^H(n) &= \mathbf{0}^H \\ \mathbf{q}^H(n) \sqrt{\lambda} \tilde{\mathbf{k}}(n-1) + \tilde{\alpha}(n) y^*(n) &= \tilde{e}^*(n) \end{aligned}$$

Solving the top equation

$$\mathbf{q}^H(n) \sqrt{\lambda} = -\tilde{\alpha}(n) \mathbf{x}^H(n) \tilde{\mathbf{R}}^{-1}(n-1)$$

and substituting to the bottom one, gives

$$\tilde{\alpha}(n) [y^*(n) - \mathbf{x}^H(n) \mathbf{c}(n-1)] = \tilde{e}^*(n) \Rightarrow \tilde{\alpha}(n) = \frac{\tilde{e}(n)}{e(n)}$$

because $\mathbf{c}(n-1) = \tilde{\mathbf{R}}^{-1}(n-1) \tilde{\mathbf{k}}(n-1)$ and $e(n) = y(n) - \mathbf{c}^H(n-1) \mathbf{x}(n)$.

10.46 Derive the formulas for the QR-RLS lattice predictor (see Table 10.18), using the approach introduced in Section 10.6.3 (Yang and Böhme 1992).

To be completed.

- 10.47** Demonstrate how the systolic array in Figure 10.55, which is an extension of the systolic array structure shown in Figure 10.36, can be used to determine the LS error $e(n)$ and the LS coefficient vector $\mathbf{c}(n)$. Determine the functions to be assigned to the dotted-line computing elements and the inputs with which they should be supplied.

To be completed.

- 10.48 (new version)** The implementation of adaptive filters using multiprocessing involves the following steps: (1) partitioning of the overall computational job into individual tasks, (2) allocation of computational and communications tasks to the processors, and (3) synchronization and control of the processors. Figure 10.56 shows a cascade multiprocessing architecture used for adaptive filtering. To avoid *latency* (i.e., a delay between the filter's input and output that is larger than the sampling interval), each processor should complete its task in time less than the sampling period and use results computed by the preceding processor and the scalar computational unit at the previous sampling interval. This is accomplished by the unit delays inserted between the processors.

- (a) Draw the block diagram of a section that updates one coefficient of the FLP using equations (a)-(h), given in Table 10.13. Then, use the obtained diagram to explain why the fast Kalman algorithm does not satisfy the multiprocessing requirements.

The block diagram of equation (g) is given in Figure 10.48a1 while that of (h) is given in Figure 10.48a2.

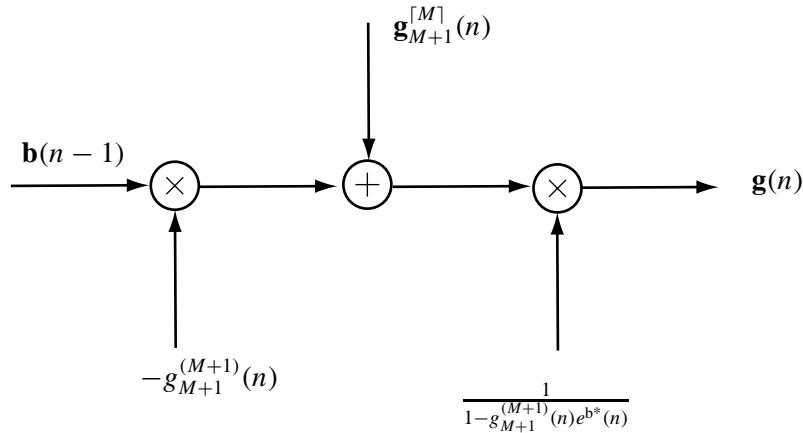


Figure 10.48a1: Block diagram for equation (g)

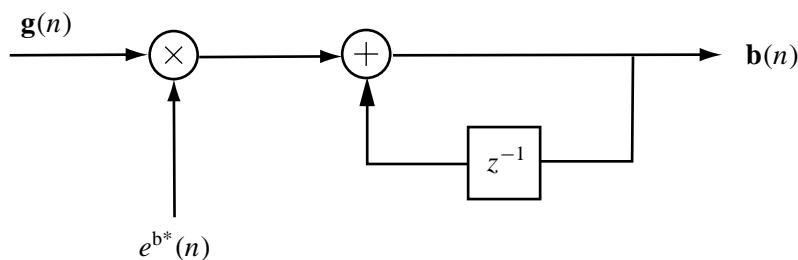


Figure 10.48a2: Block diagram for equation (h).

Combining these diagrams with equations (a)-(f) in Table 10.13 leads to the block diagram 10.48(a). The lack of delay elements in the paths for $g_{M+1}^{[M]}(n)$ and $g_{M+1}(n)$ violates the multiprocessing architecture.

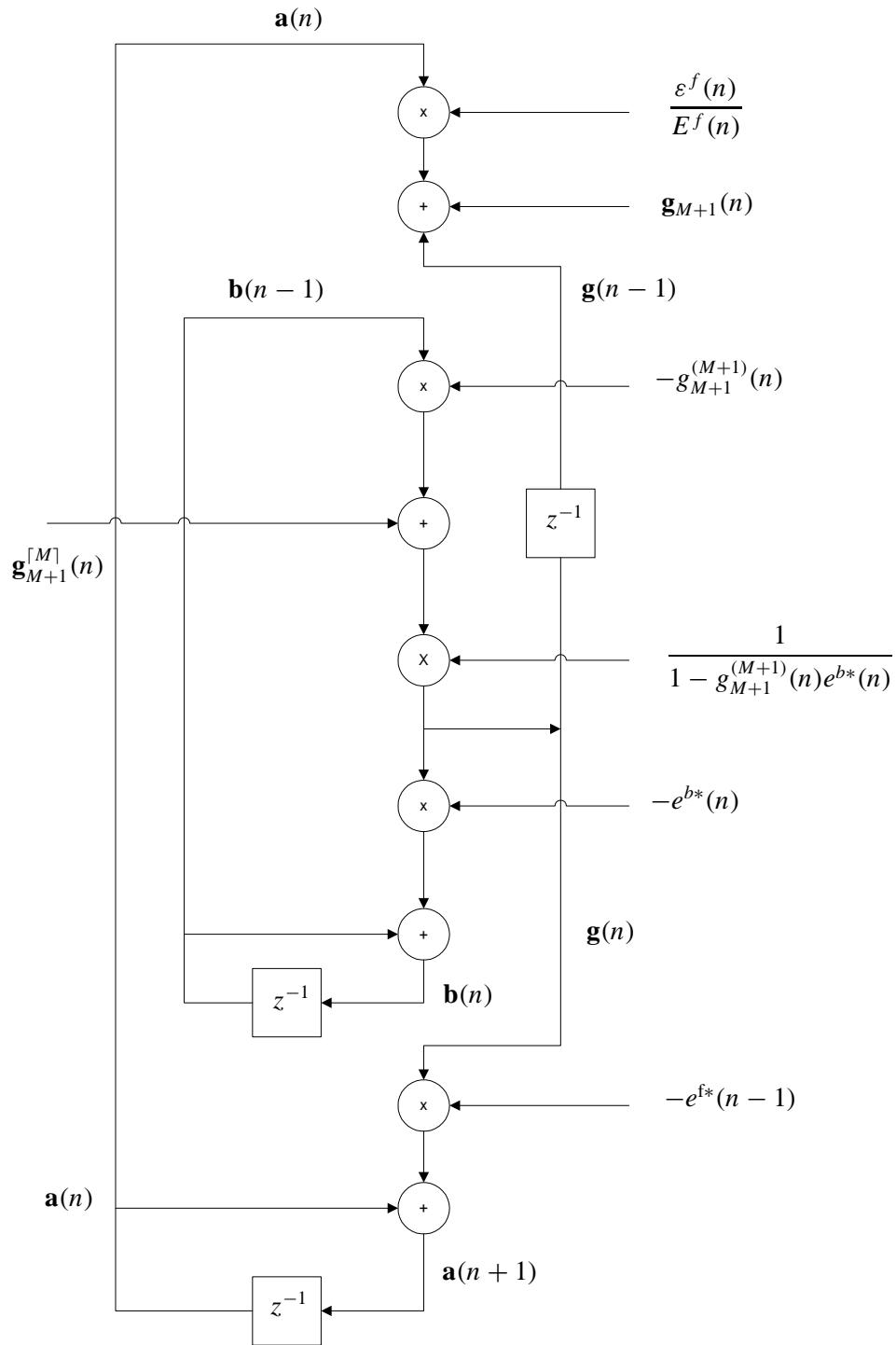


Figure 10.48a2: Multiprocessing element for the fast Kalman algorithm using equations (a)-(h) in table 10.13.

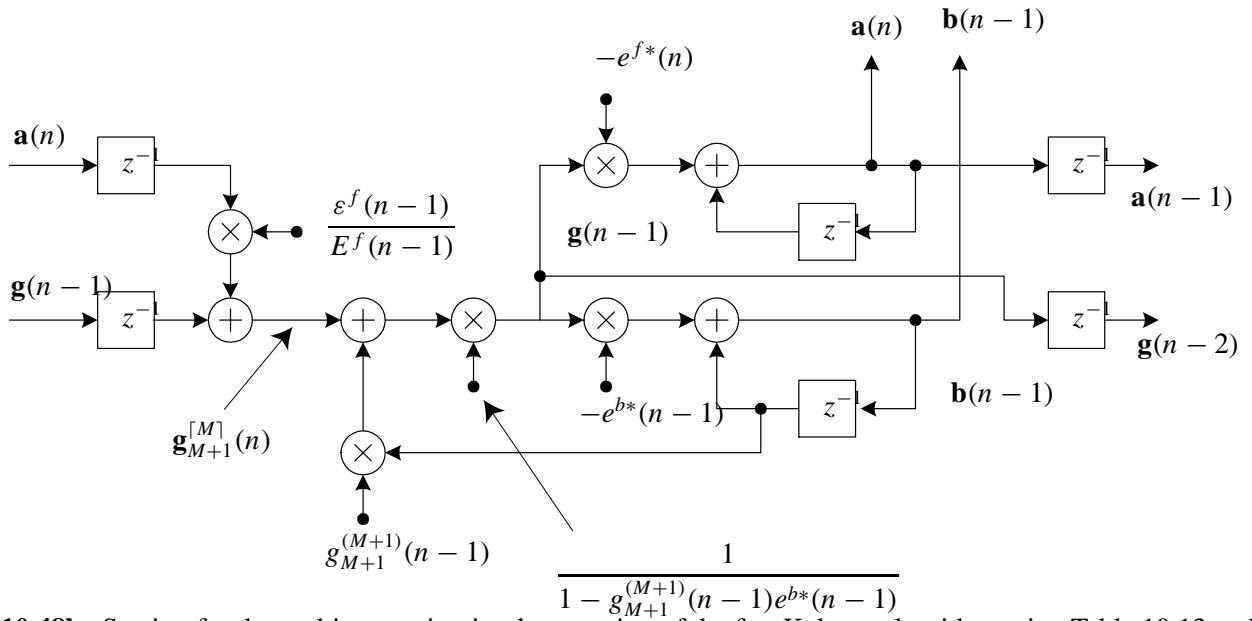


Figure 10.48b: Section for the multiprocessing implementation of the fast Kalman algorithm using Table 10.13 and equations (g) and (h).

- (b) Rearrange the formulas in Table 10.13 as follows: (e), (g), (h), (a), (b), (c), (d), (f). Replace n by $n - 1$ in (e), (k), and (l). Show that the resulting algorithm leads to the multiprocessing section shown in Figure 10.57. Identify the input-output quantities and the various multiplier factors and show that this section can be used in the multiprocessing architecture shown in Figure 10.56. Each processor in Figure 10.56 can be assigned to execute one or more of the designed sections. Note: You may find useful the discussions in Lawrence and Tewksbury (1983) and in Manolakis and Patel (1992).

Working as in (a) we obtain the results shown in Figure 10.48(b).

Comments:

The multiplier factors $\varepsilon^f(n-1)/E^f(n-1)$, $g_{M+1}^{(M+1)}(n)$, $1/\left[1 - g_{M+1}^{(M+1)}(n-1)e_m^{b*}(n-1)\right]$, $-e^{b*}(n-1)$, and $-e^{f*}(n)$ at each element, should be available after the processing of the sample $x(n-1)$ has been completed and the sample $x(n)$ has arrived. To start processing the sample $x(n)$ all these quantities should be available to each processing section.

- $e^f(n-1)$, $e^b(n-1)$, $E^f(n-1)$ are available from the previous sampling interval.
- $\varepsilon^f(n-1)/E^f(n-1)$ and $g_{M+1}^{(M+1)}(n)$ are computed at the first section as the first and last elements of $\mathbf{g}_{M+1}(n)$. However, since they are used by all processors they should be computed separately at the beginning. Then, we compute $1/\left[1 - g_{M+1}^{(M+1)}(n-1)e_m^{b*}(n-1)\right]$. Therefore, these quantities should be computed “outside” the sections before we start processing the sample $x(n)$.
- $e^f(n)$ should be available before each section finishes the computation of $\mathbf{b}(n-1)$ (see Figure ?). The solution is to compute $\hat{x}(n) = \mathbf{a}^H(n-1)\mathbf{x}(n-1)$ at the previous processing cycle and then compute $e^f(n) = x(n) + \hat{x}(n)$ after the arrival of $x(n)$.

- (c) Prove the formulas

$$\mathbf{b}(n) = \frac{\mathbf{b}(n-1) - \mathbf{g}_{M+1}^{[M]}(n)e^{b*}(n)}{1 - g_{M+1}^{(M+1)}(n)e^{b*}(n)} \quad (k)$$

$$\mathbf{g}(n) = \mathbf{g}_{M+1}^{[M]}(n) - g_{M+1}^{(M+1)}(n)\mathbf{b}(n) \quad (l)$$

and show that they can be used to replace formulas (g) and (h) in Table 10.13.

If we eliminate $\mathbf{g}_m(n)$ from (10.7.13) using (10.7.14), we obtain equations (k) and (l).

- (d) Draw a block diagram of a single multiprocessing section for the obtained algorithm and show that this section can be used in the architecture of Figure 10.56.

Figure 10.48(c) shows the updating section for the fast Kalman algorithm described by (a)-(f), (k), (l). The lack of delay elements in the paths for $\mathbf{g}_{M+1}^{[M]}(n)$ and $\mathbf{g}_{M+1}(n)$ violates the multiprocessing requirements. The multiprocessing section for this version of the fast Kalman algorithm is given in Figure 10.48(d).

- (e) Rearrange the formulas (a)-(l), in Table 10.14, to obtain a version of the FAEST algorithm that complies with the multiprocessing architecture of Figure 10.56 and draw a block diagram of the resulting updating section.

Repeat (a) and (b).

- 10.49 (new version)** Show that the LMS algorithm given in Table 10.3 satisfies the multiprocessing architecture in Figure 10.56.

If we modify the LMS algorithm as follows

$$\begin{aligned}\mathbf{c}(n) &= \mathbf{c}(n-1) + 2\mu \mathbf{x}(n-1) e^*(n-1) \\ \hat{y}(n) &= \mathbf{c}^H(n) \mathbf{x}(n) \\ e(n) &= y(n) - \hat{y}(n)\end{aligned}$$

we can easily see that the LMS algorithm complies with the multiprocessing architecture of Figure 10.56.

- 10.50** Show that the a priori RLS linear prediction lattice (i.e., without the ladder part) algorithm with error feedback complies with the multiprocessing architecture of Figure 10.56. Explain why the addition of the ladder part violates the multiprocessing architecture. Can we rectify these violations? (See Lawrence and Tewksbury 1983.)

The lattice predictor has a modular structure

$$\begin{aligned}e_0^f(n) &= e_0^b(n) = x(n) \\ e_{m+1}^f(n) &= e_m^f(n) + k_m^{f*}(n-1) e_m^b(n-1) \quad 0 \leq m < M-1 \\ e_{m+1}^b(n) &= e_m^b(n-1) + k_m^{b*}(n-1) e_m^f(n) \quad 0 \leq m < M-1\end{aligned}$$

Multiprocessing is impossible because there is no delay in the $e_m^f(n)$ path (if $e_m^f(n)$ could be replaced by $e_m^f(n-1)$ there would be no problem). To overcome this problem, we notice that

$$e_m^f(n) = x(n) + \mathbf{a}_m^H(n-1) \mathbf{x}_m(n-1) \triangleq x(n) - \hat{x}_m(n)$$

which combined with $e_{m+1}^f(n) = e_m^f(n) + k_m^{f*}(n-1) e_m^b(n-1)$ gives

$$\hat{x}_{m+1}(n) = \hat{x}_m(n) + k_m^{f*}(n-1) e_m^b(n-1)$$

which is initialized $\hat{x}_0(n) = 0$. Since $e_0^f(n) = x(n)$ only $x(n)$ is not available at time n . Thus, instead of $e_{m+1}^f(n)$ we can compute $\hat{x}_{m+1}(n)$ in advance, store it and have it ready for use at time n when $x(n)$ becomes available. These observations lead to the lattice section shown in Figure 10.50.

- 10.51** The fixed-length sliding window RLS algorithm is given in (10.8.4) through (10.8.10).

- (a) Derive the above equations of this algorithm (see Manolakis et al. 1987).

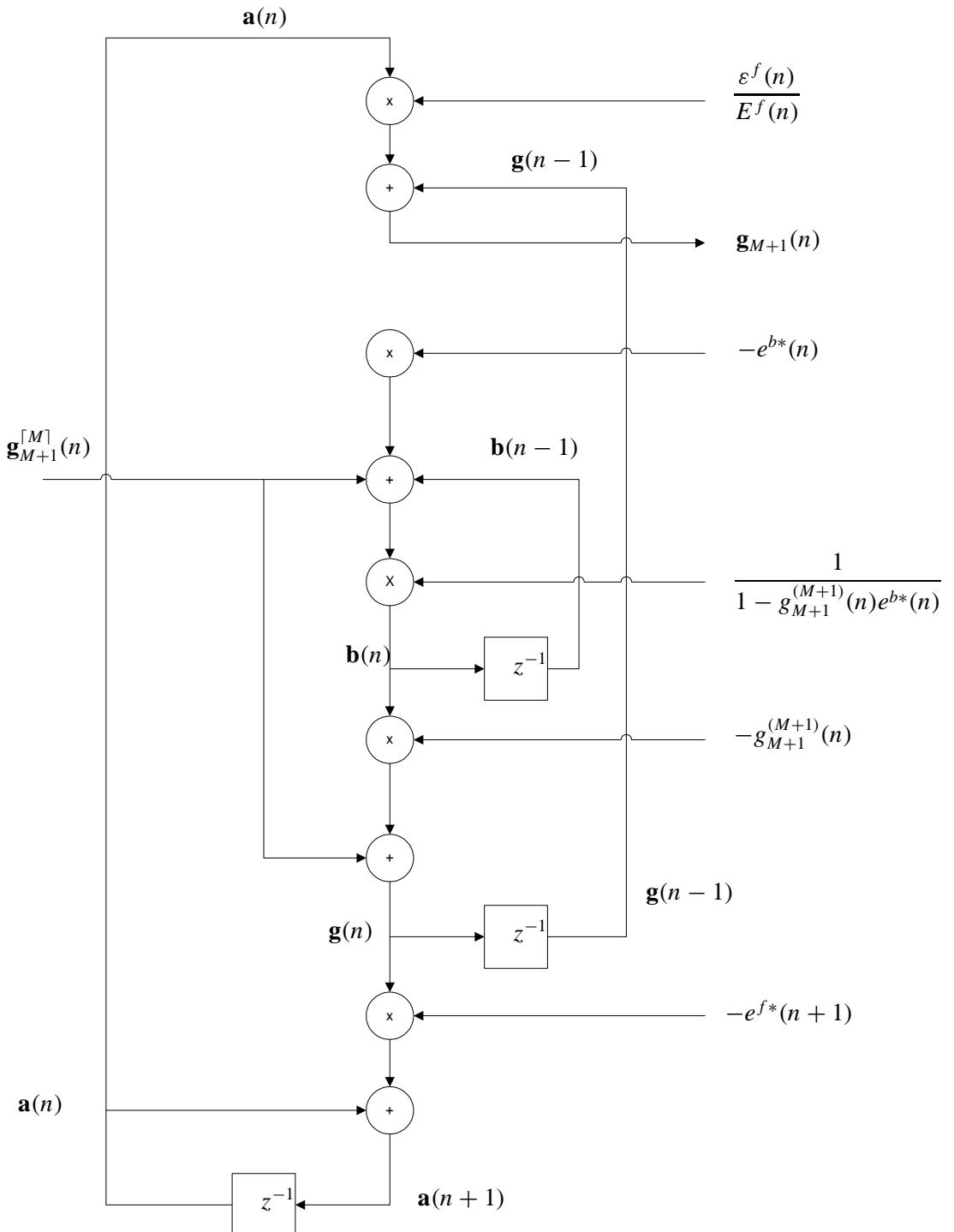


Figure 10.48c: Section for the multiprocessing implementation of the fast Kalman algorithm using Table 10.13 and equations (k) and (l).

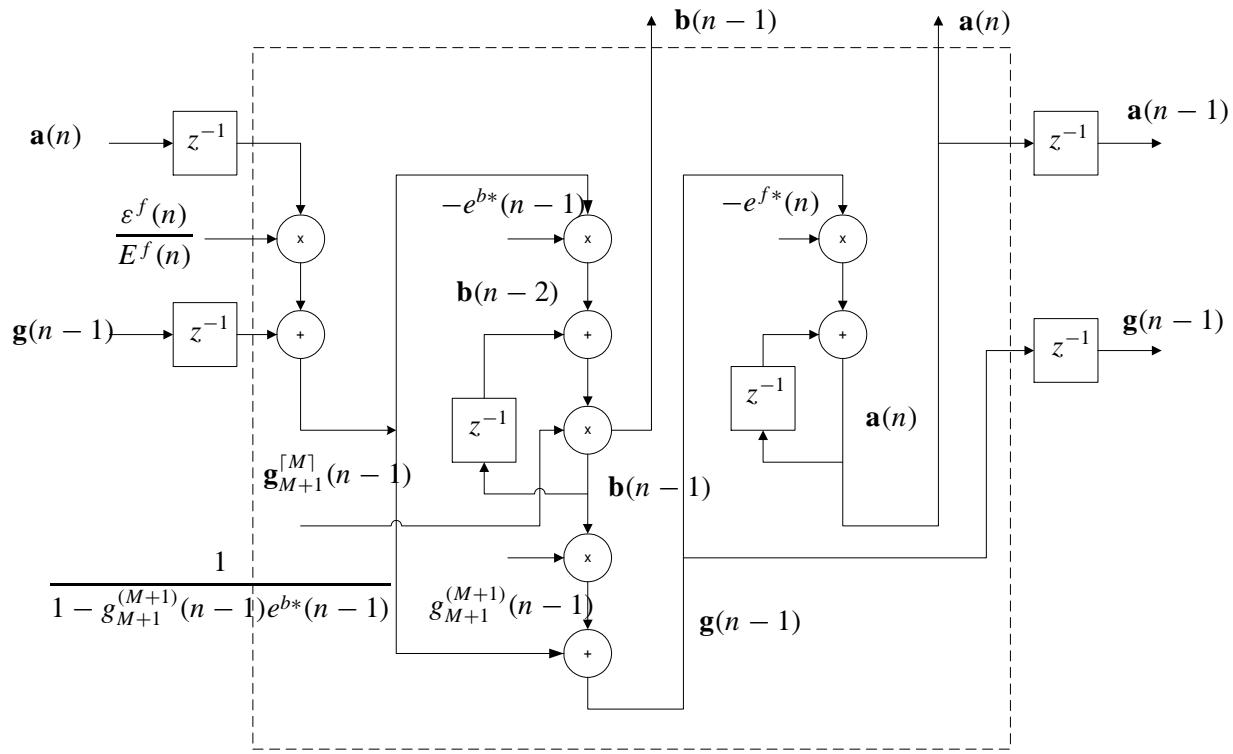


Figure 10.48c: Section for the multiprocessing implementation of the fast Kalman algorithm using Table 10.13 and equations (k) and (l) This is the version proposed by Lawrence and Tewksbury.

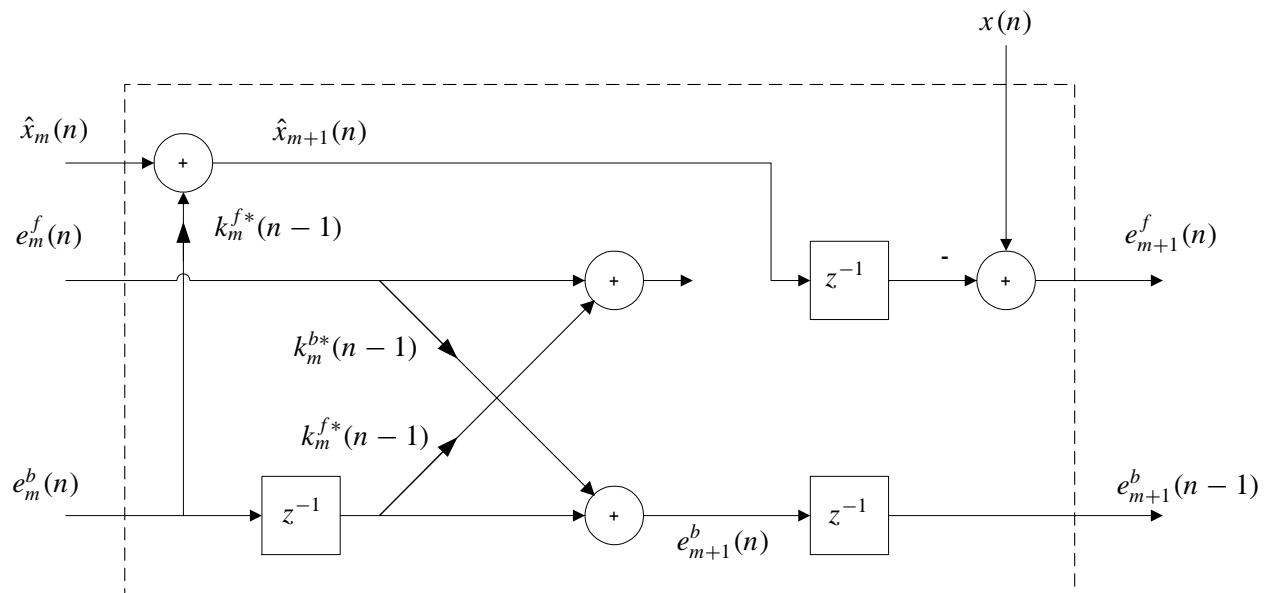


Figure 10.50: Multiprocessing session for the a-priori RLS lattice filter.

- (b) Develop a Matlab function to implement the algorithm

```
[c, e] = slwrls(x, y, L, delta, M, c0)
```

where L is the fixed length of the window.

- (c) Generate 500 samples of the following nonstationary process

$$x(n) = \begin{cases} w(n) + 0.95x(n-1) - 0.9x(n-2) & 0 \leq n < 200 \\ w(n) - 0.95x(n-1) - 0.9x(n-2) & 200 \leq n < 300 \\ w(n) + 0.95x(n-1) - 0.9x(n-2) & n \geq 300 \end{cases}$$

where $w(n)$ is a zero-mean, unit-variance white noise process. We want to obtain a second-order linear predictor using adaptive algorithms. Use the sliding window RLS algorithm on the data and choose $L = 50$ and 100 . Obtain plots of the filter coefficients and mean square error.

- (d) Now use the growing memory RLS algorithm by choosing $\lambda = 1$. Compare your results with the sliding-window RLS algorithm.
(e) Finally, use the exponentially growing memory RLS by choosing $\lambda = (L-1)/(L+1)$ that produces the same MSE. Compare your results.

To be completed.

- 10.52** Consider the definition of the MSD $\mathcal{D}(n)$ in (10.2.29) and that of the trace of a matrix (A.2.16).

- (a) Show that $D(n) = \text{tr}\{\Phi(n)\}$, where $\Phi(n)$ is the correlation matrix of $\tilde{\mathbf{c}}(n)$.
(b) For the evolution of the correlation matrix in (10.8.58), show that

$$\mathcal{D}(\infty) \simeq \mu M \sigma_v^2 + \frac{\text{tr}(\mathbf{R}^{-1} \mathbf{R}_\psi)}{4\mu}$$

To be completed.

- 10.53** Consider the analysis model given in Figure 10.42. Let the parameters of this model be as follows:

$$\begin{aligned} \mathbf{c}_o(n) \text{ model parameters: } \mathbf{c}_o(0) &= \begin{bmatrix} 0.9 \\ -0.8 \end{bmatrix} & M &= 2 & \rho &= 0.95 \\ \boldsymbol{\psi}(n) &\sim \text{WGN}(\mathbf{0}, \mathbf{R}_\psi) & \mathbf{R}_\psi &= (0.01)^2 \mathbf{I} \end{aligned}$$

Signal $\mathbf{x}(n)$ parameters $\mathbf{x}(n) \sim \text{WGN}(\mathbf{0}, \mathbf{R}) \quad \mathbf{R} = \mathbf{I}$

Noise $v(n)$ parameters $v(n) \sim \text{WGN}(0, \sigma_v^2) \quad \sigma_v = 0.1$

Simulate the system, using three values of μ that show slow, matched, and optimum adaptations of the LMS algorithm.

- (a) Obtain the tracking plots similar to Figure 10.43 for each of the above three adaptations.
(b) Obtain the learning curve plots similar to Figure 10.44 for each of the above three adaptations.

To be completed.

10.54 Consider the analysis model given in Figure 10.42. Let the parameters of this model be as follows

$$\mathbf{c}_o(n) \text{ model parameters } \mathbf{c}_o(0) = \begin{bmatrix} 0.9 \\ -0.8 \end{bmatrix} \quad M = 2 \quad \rho = 0.95$$

$$\boldsymbol{\psi}(n) \sim \text{WGN}(\mathbf{0}, \mathbf{R}_\psi) \quad \mathbf{R}_\psi = (0.01)^2 \mathbf{I}$$

$$\text{Signal } \mathbf{x}(n) \text{ parameters } \mathbf{x}(n) \sim \text{WGN}(\mathbf{0}, \mathbf{R}) \quad \mathbf{R} = \mathbf{I}$$

$$\text{Noise } v(n) \text{ parameters } v(n) \sim \text{WGN}(0, \sigma_v^2) \quad \sigma_v = 0.1$$

Simulate the system, using three values of μ that show slow, matched, and optimum adaptations of the RLS algorithm.

- (a) Obtain the tracking plots similar to Figure 10.49 for each of the above three adaptations.
- (b) Obtain the learning curve plots similar to Figure 10.50 for each of the above three adaptations.
- (c) Compare your results with those obtained in Problem 10.53.

To be completed.

10.55 Consider the time-varying adaptive equalizer shown in Figure 10.58 in which the time variation of the channel impulse response is given by

$$h(n) = \rho h(n-1) + \sqrt{1-\rho} \eta(n), \text{ with}$$

$$\rho = 0.95 \quad \eta(n) \sim \text{WGN}(0, \sqrt{10}) \quad h(0) = 0.5$$

Let the equalizer be a single-tap equalizer and $v(n) \sim \text{WGN}(0, 0.1)$.

- (a) Simulate the system for three different adaptations; that is, choose μ for slow, matched, and fast adaptations of the LMS algorithm.
- (b) Repeat part (a), using the RLS algorithm.

To be completed.

Array Processing

- 11.1** The 2 element array with inter-element spacing of d has a signal impinging on it from an angle of $\phi = 0^\circ$ at a rate of propagation c .

- (a) If the wavelength of the propagating signal is given by

$$\lambda = \frac{c}{F_c}$$

where F_c is the frequency of the signal, then the difference in the arrival time of the signal at the two sensors is

$$\begin{aligned}\tau &= \frac{d}{\lambda} \sin \phi \\ &= 0\end{aligned}$$

since $\phi = 0^\circ$. The signal arrives at the two sensors at the same time.

- (b) The same delay of 0 seconds between sensors is produced by any other signals for which $\sin \phi = 0$. Therefore, signals impinging on the array from $\phi = 180^\circ$ will also produce a delay of zero.
- (c) With a single sensor we cannot determine the angle from which a signal arrives because we cannot determine a difference in arrival time. In other words, signals from all directions will have the same characteristics.

- 11.2** This problem is a MATLAB based problem. Below is the MATLAB code and beampatterns at $\phi = 45^\circ$ and $\phi = 60^\circ$. Note that mechanical steering is modeled as the beampattern for electronic steering to broadside and shifting the angle to the desired mechanical steering angle. In both cases we observe a smaller mainbeam for the mechanical steering than the electronic steering due to the fact that spatial frequency is a function of $\sin \phi$ and not simply ϕ . This effect is more pronounced the greater the steering angle, so it is more noticeable for the case of $\phi = 60^\circ$.

```
%  
% Problem 11.2  
%  
clear  
  
M = 20; % number of elements  
spacing = 0.5; % lambda/2 spacing  
  
% First part of problem has a steering angle of 45 degrees  
phi_steer = 45; % steering direction  
  
% Compute the steering vectors for electronic and mechanical steering  
v_elec = exp(-j*2*pi*spacing*[0:(M-1)'] * sin(phi_steer*pi/180)) / sqrt(M);  
v_mech = exp(-j*2*pi*spacing*[0:(M-1)'] * sin(0*pi/180)) / sqrt(M);  
  
% Angles for beampattern
```

```

phi = -90:90;
V = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi*pi/180))/sqrt(M);

% Compute the beampatterns for electronic and mechanical steering
V_elec = V'*v_elec;
V_mech = V'*v_mech;

% Mechanical steering angles are simply the true angles shifted by
% the mechanical steering angle
phi_mech = phi + phi_steer; % mechanical steering angles for beampattern
index_toobig = find(phi_mech > 90);
phi_mech(index_toobig) = phi_mech(index_toobig)-180;
[phi_mech,index] = sort(phi_mech);
V_mech = V_mech(index);

% Plot out the beampatterns in decibels
figure
plot(phi,20*log10(abs(V_elec)), 'b',...
      phi_mech,20*log10(abs(V_mech)), 'r--',...
      'linewidth',2.5);
xlabel('Angle (deg)', 'fontweight','bold', 'fontsize',20);
ylabel('Power Response (dB)', 'fontweight','bold', 'fontsize',20);
set(gca, 'fontweight','bold', 'fontsize',18);
axis([-90 90 -50 0]);
grid;
set(gca, 'xtick', -90:15:90);

% Repeat for a steering angle of 60 degrees
phi_steer = 60; % steering direction

% Compute the steering vectors for electronic and mechanical steering
v_elec = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi_steer*pi/180))/sqrt(M);
v_mech = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(0*pi/180))/sqrt(M);

% Angles for beampattern
phi = -90:90;
V = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi*pi/180))/sqrt(M);

% Compute the beampatterns for electronic and mechanical steering
V_elec = V'*v_elec;
V_mech = V'*v_mech;

% Mechanical steering angles are simply the true angles shifted by
% the mechanical steering angle
phi_mech = phi + phi_steer; % mechanical steering angles for beampattern
index_toobig = find(phi_mech > 90);
phi_mech(index_toobig) = phi_mech(index_toobig)-180;
[phi_mech,index] = sort(phi_mech);
V_mech = V_mech(index);

```

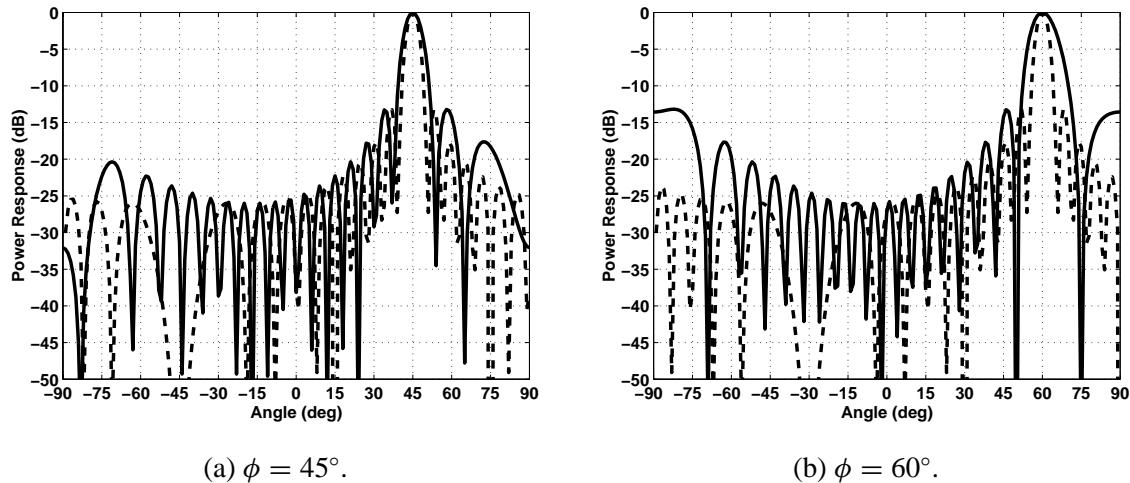


Figure 11.2a: Beampatterns for a $M = 20 \lambda/2$ -spaced ULA with electronic steering (solid) and mechanical steering (dashed).

```
% Plot out the beampatterns in decibels
figure
plot(phi,20*log10(abs(V_elec)), 'b',...
    phi_mech,20*log10(abs(V_mech)), 'r--',...
    'linewidth',2.5);
xlabel('Angle (deg)', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Power Response (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca, 'fontweight', 'bold', 'fontsize', 18);
axis([-90 90 -50 0]);
grid
set(gca, 'xtick', -90:15:90);
```

- 11.3** This problem is a MATLAB based problem. Below is the MATLAB code and three output figures. The code is broken up into parts (a), (b), and (c) and presented sequentially along with the accompanying figures.

(a) Both signals are clearly visible in the steered responses. The expected steered response is so close to the estimated steered response that the two overlay in Figure 11.3a.

```

%
% Problem 11.3
%
clear

M = 50; % number of elements
spacing = 0.5; % lambda/2 spacing
N = 1000; % number of samples

%%%%%%%%%%%%%
%
% Part A

```

```

%
%%%%%%%%%%%%%%%
% The angles in degrees of the two spatially propagating signals
phi1 = -10;
phi2 = 30;

% The power levels of the signals and the noise
P1_db = 20;
P2_db = 25;
Pn = 0;

% Compute the array response vectors of the two signals
v1 = exp(-j*2*pi*spacing*[0:(M-1)']*sin(phi1*pi/180))/sqrt(M);
v2 = exp(-j*2*pi*spacing*[0:(M-1)']*sin(phi2*pi/180))/sqrt(M);

% Generate the signals 1 and 2 and the noise
s1 = (10^(P1_db/20)) * v1 * ((randn(1,N) + j*randn(1,N))/sqrt(2));
s2 = (10^(P2_db/20)) * v2 * ((randn(1,N) + j*randn(1,N))/sqrt(2));
w = (10^(Pn/20)) * (randn(M,N) + j*randn(M,N))/sqrt(2); % Noise

% Add signals and noise to get the total array signal
x = s1 + s2 + w;

% Compute the true covariance matrix
% (remember to divide by 10 not 20 for powers in the covariance)
R_x = (10^(P1_db/10)) * v1 * v1' + (10^(P2_db/10)) * v2 * v2' + ...
      (10^(Pn/10)) * eye(M);

% Generate the spatial matched filters to compute the steered
% response for angles from -90 to 90 degrees
phi = -90:0.2:90;
C_smf = exp(-j*2*pi*spacing*[0:(M-1)']*sin(phi*pi/180))/sqrt(M);

% Compute the steered response
for k = 1:length(phi)
    R_steer_est(k) = mean(abs(C_smf(:,k)' * x).^2,2);
    R_steer_true(k) = abs(C_smf(:,k)' * R_x * C_smf(:,k));
end

% Plot out the steered response in decibels
figure
plot(phi,10*log10(R_steer_est),'b','linewidth',2.5);
xlabel('Angle (deg)', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Steered Response (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca, 'fontweight', 'bold', 'fontsize', 18);
axis([-90 90 -5 65]);
grid
set(gca, 'xtick', -90:15:90);

```

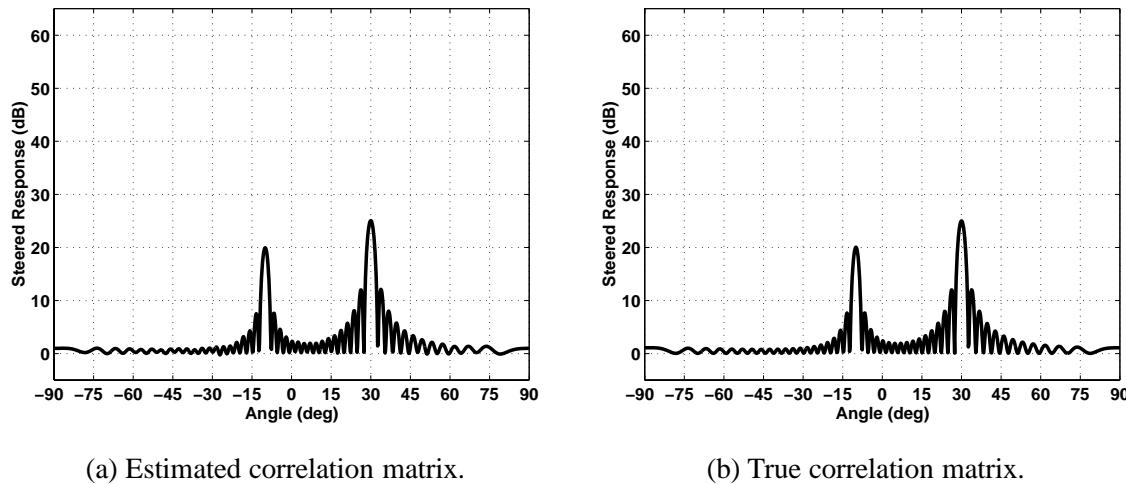


Figure 11.3a: Steered responses with estimated and true correlation matrices for scenario in Problem 11.3(a) with signals at $\phi = -10^\circ$ and $\phi = 30^\circ$ with powers of 20 and 25 dB, respectively.

```
% Plot out the steered response in decibels
figure
plot(phi,10*log10(R_steer_true), 'b', 'linewidth', 2.5);
xlabel('Angle (deg)', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Steered Response (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca, 'fontweight', 'bold', 'fontsize', 18);
axis([-90 90 -5 65]);
grid
set(gca, 'xtick', -90:15:90);
```

(b) The beampattern for the beamformer steered to $\phi = 30^\circ$ is shown in Figure 11.3b. The gain at $\phi = 30^\circ$ is one (0 dB) so that the steered response in Figure 11.3a shows the correct power of the 25 dB signal at $\phi = 30^\circ$. The same is true of the signal at $\phi = -10^\circ$. Elsewhere, we see the two signals leaked through the first few principal sidelobes in the steered response near the two sources, but otherwise the response is near 0 dB, the noise level, since the two signals are not strong enough to leak through the further out sidelobes of the beamformer.

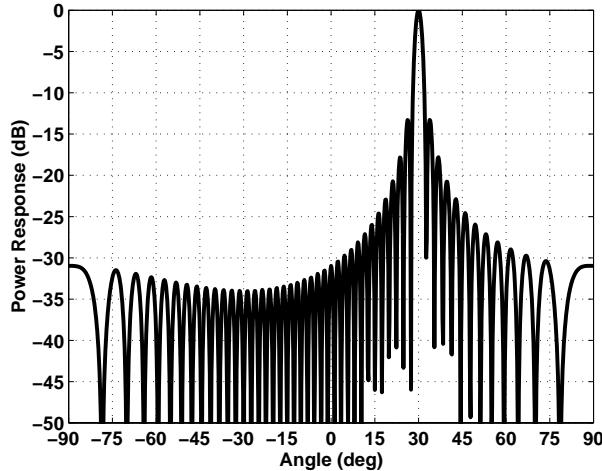


Figure 11.3b: The beampattern for a $M = 50 \lambda/2$ -spaced ULA steered to $\phi = 30^\circ$.

```
% Compute the beampatterns for electronic and mechanical steering
C_bp = V'*c_smf;
```

```
% Plot out the steered response in decibels
figure
plot(phi,20*log10(abs(C_bp)), 'b', 'linewidth', 2.5);
xlabel('Angle (deg)', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Power Response (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca, 'fontweight', 'bold', 'fontsize', 18);
axis([-90 90 -50 0]);
grid
set(gca, 'xtick', -90:15:90);
```

(c) The steered response of the spatial matched filter is shown in Figure 11.3c. Clearly the 60-dB signal is so strong that it leaks through sidelobes of all the beamformers used to compute the steered response and the 25 dB signal at $\phi = 30^\circ$ is no longer visible. Using a tapered beamformer with a Chebyshev window with -65 -dB sidelobes is sufficient to suppress this 60-dB signal when not steered to $\phi = -10^\circ$, as seen in the steered response in Figure 11.3c. Note, however, that a slight loss is incurred for both signals and that each one spreads out slightly further in angle as opposed to the untapered spatial matched filter.

```
%%%%%%%%%%%%%%%
%
```

```
% Part C
%
```

```
%%%%%%%%%%%%%%%
%
```

```
% The power levels of the signals and the noise
```

```
P1_db = 60;
```

```
% Generate the signals 1 and 2 and the noise
```

```
s1 = (10^(P1_db/20)) * v1 * ((randn(1,N) + j*randn(1,N))/sqrt(2));
```

```
s2 = (10^(P2_db/20)) * v2 * ((randn(1,N) + j*randn(1,N))/sqrt(2));
```

```
w = (10^(Pn/20)) * (randn(M,N) + j*randn(M,N))/sqrt(2); % Noise

% Add signals and noise to get the total array signal
x = s1 + s2 + w;

% Compute the true covariance matrix
% (remember to divide by 10 not 20 for powers in the covariance)
R_x = (10^(P1_db/10)) * v1 * v1' + (10^(P2_db/10)) * v2 * v2' + ...
(10^(Pn/10)) * eye(M);

% Generate the spatial matched filters to compute the steered
% response for angles from -90 to 90 degrees
phi = -90:0.2:90;
C_smf = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi*pi/180))/sqrt(M);

% Use a tapered beamformer with 65 dB sidelobe levels
SLL_taper = 65;
t = chebwin(M,SLL_taper);
for m = 1:length(phi)
    C_tbf(:,m) = diag(t)*C_smf(:,m);
    C_tbf(:,m) = C_tbf(:,m)/sqrt(C_tbf(:,m)'*C_tbf(:,m));
end

% Compute the steered response
for k = 1:length(phi)
    R_steer_smf(k) = mean(abs(C_smf(:,k)' * x).^2,2);
    R_steer_tbf(k) = mean(abs(C_tbf(:,k)' * x).^2,2);
end

% Plot out the steered response in decibels
figure
plot(phi,10*log10(R_steer_smf),'b','linewidth',2.5);
xlabel('Angle (deg)', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Steered Response (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca,'fontweight','bold','fontsize',18);
axis([-90 90 -5 65]);
grid
set(gca,'xtick',-90:15:90);

% Plot out the steered response in decibels
figure
plot(phi,10*log10(R_steer_tbf),'b','linewidth',2.5);
xlabel('Angle (deg)', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Steered Response (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca,'fontweight','bold','fontsize',18);
axis([-90 90 -5 65]);
grid
set(gca,'xtick',-90:15:90);
```

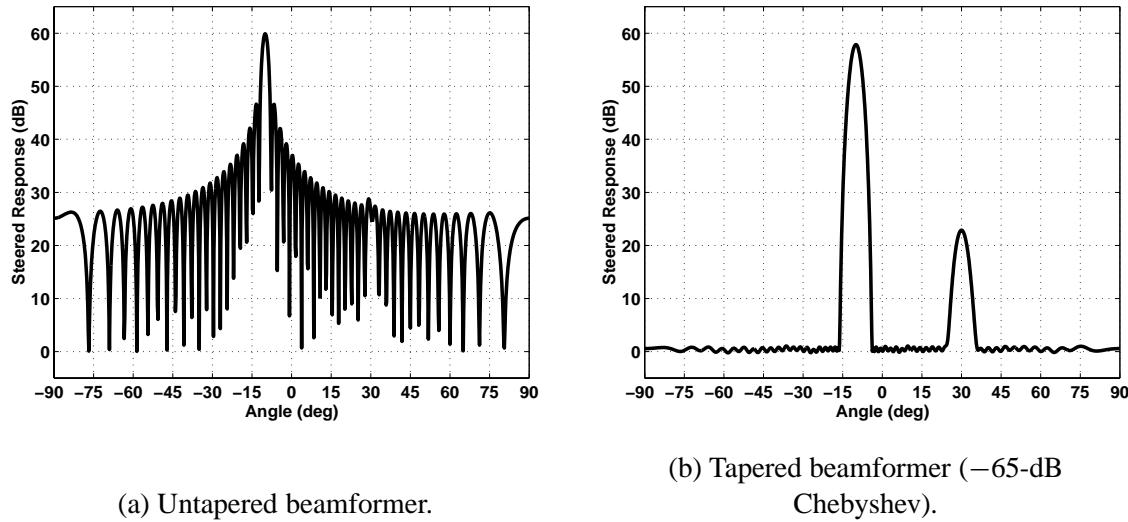


Figure 11.3c: Steered responses for scenario in Problem 11.3(c) with signals at $\phi = -10^\circ$ and $\phi = 30^\circ$ with powers of 60 and 25 dB, respectively.

11.4 This problem is a MATLAB based problem. Below is the MATLAB code and six output figures. Note that for this problem the signals should be at $\phi = 0^\circ$ and $\phi = 3^\circ$. If your text does not have this correction, please pass this along to the students. The code is broken up into parts (a), (b), (c), and (d) and presented sequentially along with the accompanying figures.

(a) The two signal at $\phi = 0^\circ$ and $\phi = 3^\circ$ are not resolved in the steered response. The cause is the limited aperture and the proximity of the two signals in angle.

```
%  
% Problem 11.4  
%  
clear  
  
M = 30; % number of elements  
spacing = 0.5; % element spacing in wavelengths  
N = 1000; % number of samples  
  
%%%%%%%%%%%%%%%  
%  
% Part A  
%  
%%%%%%%%%%%%%%%  
  
% The angles in degrees of the two spatially propagating signals  
phi1 = 0;  
phi2 = 3;  
  
% The power levels of the signals and the noise  
P1_db = 20;  
P2_db = 20;  
Pn = 0;
```

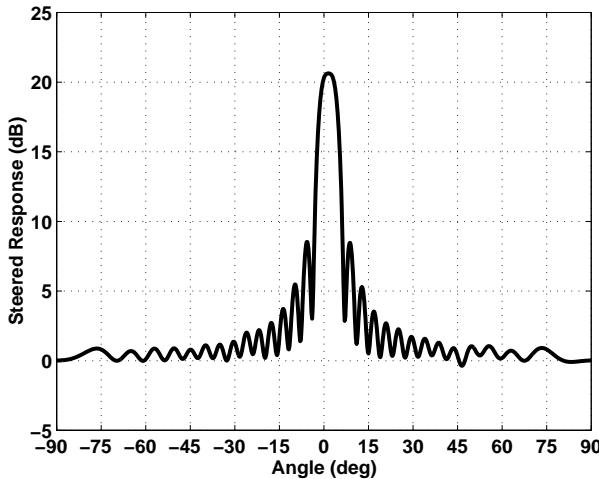


Figure 11.4a: Steered responses for scenario in Problem 11.4(a) for an array with $M = 30 \lambda/2$ -spaced elements with signals at $\phi = 0^\circ$ and $\phi = 3^\circ$ both with powers of 20 dB.

```
% Compute the array response vectors of the two signals
v1 = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi1*pi/180))/sqrt(M);
v2 = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi2*pi/180))/sqrt(M);

% Generate the signals 1 and 2 and the noise
s1 = (10^(P1_db/20)) * v1 * ((randn(1,N) + j*randn(1,N))/sqrt(2));
s2 = (10^(P2_db/20)) * v2 * ((randn(1,N) + j*randn(1,N))/sqrt(2));
w = (10^(Pn/20)) * (randn(M,N) + j*randn(M,N))/sqrt(2); % Noise

% Add signals and noise to get the total array signal
x = s1 + s2 + w;

% Generate the spatial matched filters to compute the steered
% response for angles from -90 to 90 degrees
phi = -90:0.2:90;
C_smf = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi*pi/180))/sqrt(M);

% Compute the steered response
R_steer_est = mean(abs(C_smf' * x).^2,2);

% Plot out the steered response in decibels
figure
plot(phi,10*log10(R_steer_est),'b','linewidth',2.5);
xlabel('Angle (deg)', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Steered Response (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca,'fontweight','bold','fontsize',18);
axis([-90 90 -5 25]);
grid
set(gca,'xtick',-90:15:90);
```

(b) Now the two signals have been resolved in the steered response because we have doubled the aperture of

the array and thus improved its resolution by a factor of two.

```
%%%%%%%%%%%%%%%
%
% Part B
%
%%%%%%%%%%%%%%%
% Clear out the entire workspace and start with 60 elements
clear

M = 60; % number of elements
spacing = 0.5; % element spacing in wavelengths
N = 1000; % number of samples

% The angles in degrees of the two spatially propagating signals
phi1 = 0;
phi2 = 3;

% The power levels of the signals and the noise
P1_db = 20;
P2_db = 20;
Pn = 0;

% Compute the array response vectors of the two signals
v1 = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi1*pi/180))/sqrt(M);
v2 = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi2*pi/180))/sqrt(M);

% Generate the signals 1 and 2 and the noise
s1 = (10^(P1_db/20)) * v1 * ((randn(1,N) + j*randn(1,N))/sqrt(2));
s2 = (10^(P2_db/20)) * v2 * ((randn(1,N) + j*randn(1,N))/sqrt(2));
w = (10^(Pn/20)) * (randn(M,N) + j*randn(M,N))/sqrt(2); % Noise

% Add signals and noise to get the total array signal
x = s1 + s2 + w;

% Generate the spatial matched filters to compute the steered
% response for angles from -90 to 90 degrees
phi = -90:0.2:90;
C_smf = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi*pi/180))/sqrt(M);

% Compute the steered response
R_steer_est = mean(abs(C_smf' * x).^2,2);

% Plot out the steered response in decibels
figure
plot(phi,10*log10(R_steer_est),'b','linewidth',2.5);
xlabel('Angle (deg)', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Steered Response (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca, 'fontweight', 'bold', 'fontsize', 18);
```

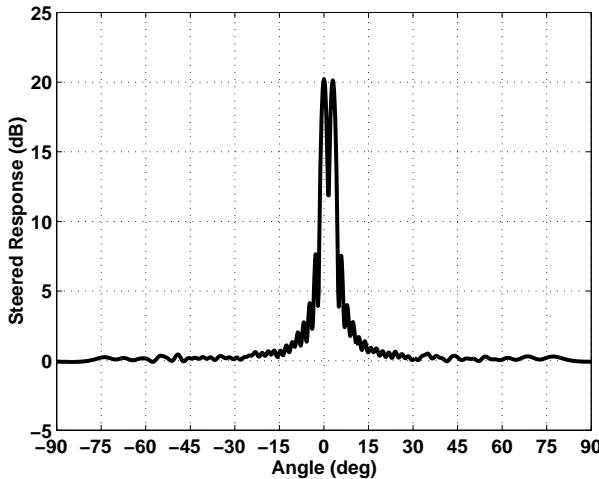


Figure 11.4b: Steered responses for scenario in Problem 11.4(a) for an array with $M = 60 \lambda/2$ -spaced elements with signals at $\phi = 0^\circ$ and $\phi = 3^\circ$ both with powers of 20 dB.

```
axis([-90 90 -5 25]);
grid
set(gca,'xtick',-90:15:90);
```

(c) Despite the fact that the array only has $M = 30$ elements as in part (a), we are able to resolve the two signals. The reason is that the aperture for the $M = 30 \lambda$ -spaced element array is equal to the aperture of the $M = 60 \lambda/2$ -spaced element. Note that in this case due to the increased element spacing we observe spatial aliasing in the steered response in the form of the additional peaks at $\phi \approx 75^\circ$ and $\phi \approx 90^\circ$

```
%%%%%%%%%%%%%
%
% Part C
%
%%%%%%%%%%%%%
% Clear out the entire workspace and start with 30 elements but with
% element spacing of lambda
clear

M = 30; % number of elements
spacing = 1; % element spacing in wavelengths
N = 1000; % number of samples

% The angles in degrees of the two spatially propagating signals
phi1 = 0;
phi2 = 3;

% The power levels of the signals and the noise
P1_db = 20;
P2_db = 20;
Pn = 0;

% Compute the array response vectors of the two signals
```

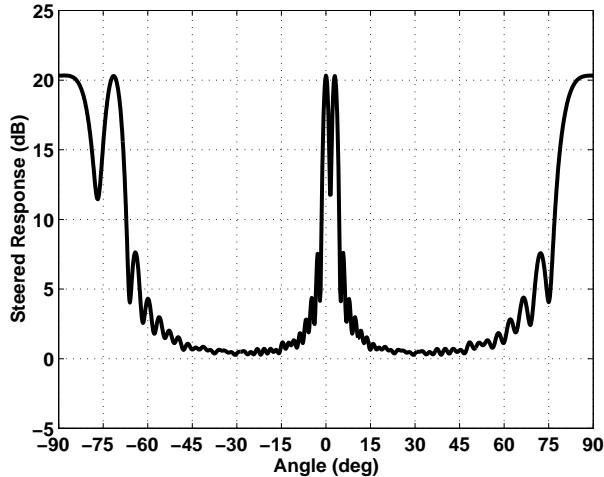


Figure 11.4c: Steered responses for scenario in Problem 11.4(a) for an array with $M = 60 \lambda$ -spaced elements with signals at $\phi = 0^\circ$ and $\phi = 3^\circ$ both with powers of 20 dB.

```

v1 = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi1*pi/180))/sqrt(M);
v2 = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi2*pi/180))/sqrt(M);

% Generate the signals 1 and 2 and the noise
s1 = (10^(P1_db/20)) * v1 * ((randn(1,N) + j*randn(1,N))/sqrt(2));
s2 = (10^(P2_db/20)) * v2 * ((randn(1,N) + j*randn(1,N))/sqrt(2));
w = (10^(Pn/20)) * (randn(M,N) + j*randn(M,N))/sqrt(2); % Noise

% Add signals and noise to get the total array signal
x = s1 + s2 + w;

% Generate the spatial matched filters to compute the steered
% response for angles from -90 to 90 degrees
phi = -90:0.2:90;
C_smf = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi*pi/180))/sqrt(M);

% Compute the steered response
R_steer_est = mean(abs(C_smf' * x).^2,2);

% Plot out the steered response in decibels
figure
plot(phi,10*log10(R_steer_est),'b','linewidth',2.5);
xlabel('Angle (deg)', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Steered Response (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca,'fontweight','bold','fontsize',18);
axis([-90 90 -5 25]);
grid
set(gca,'xtick',-90:15:90);

```

(d) The beampatterns for the spatial matched filter (beamformer) at $\phi = 0^\circ$ used to compute the steered response are plotted in Figure 11.4d. Increasing the number of $\lambda/2$ -spaced elements from $M = 30$ to $M = 60$ increases the resolution of the beamformer by a factor of two as is evidenced by a mainbeam with half the size. In the

case of the $M = 30 \lambda$ -spaced element array, the mainbeam size is the same as for the $M = 60 \lambda/2$ -spaced element array since the aperture of the two arrays is the same. However, in the case of the λ element spacing, we observe a grating lobe at $\phi = 90^\circ$ for the beamformer steered to $\phi = 0^\circ$.

```
%%%%%%%%%%%%%
%
% Part D
%
%%%%%%%%%%%%%
% Clear out the entire workspace and compute the beampatterns for
% the spatial matched filters in parts a, b, and c steered to array broadside
clear

phi_steer = 0; % steering angle (degrees) for spatial matched filter

% Compute beamformer from part a
M = 30; % number of elements
spacing = 0.5; % element spacing in wavelengths
c_bf1 = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi_steer*pi/180))/sqrt(M);

% The angles at which to evaluate the beampattern
phi = -90:0.2:90;
V = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi*pi/180))/sqrt(M);
C_bp1 = V'*c_bf1; % beampattern

% Compute beamformer from part a
M = 60; % number of elements
spacing = 0.5; % element spacing in wavelengths
c_bf2 = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi_steer*pi/180))/sqrt(M);

% The angles at which to evaluate the beampattern
phi = -90:0.2:90;
V = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi*pi/180))/sqrt(M);
C_bp2 = V'*c_bf2; % beampattern

% Compute beamformer from part a
M = 30; % number of elements
spacing = 1; % element spacing in wavelengths
c_bf3 = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi_steer*pi/180))/sqrt(M);

% The angles at which to evaluate the beampattern
phi = -90:0.2:90;
V = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi*pi/180))/sqrt(M);
C_bp3 = V'*c_bf3; % beampattern

% Plot out the beampattern in decibels
figure
plot(phi,20*log10(abs(C_bp1)), 'b', 'linewidth', 2.5);
xlabel('Angle (deg)', 'fontweight', 'bold', 'fontsize', 20);
```

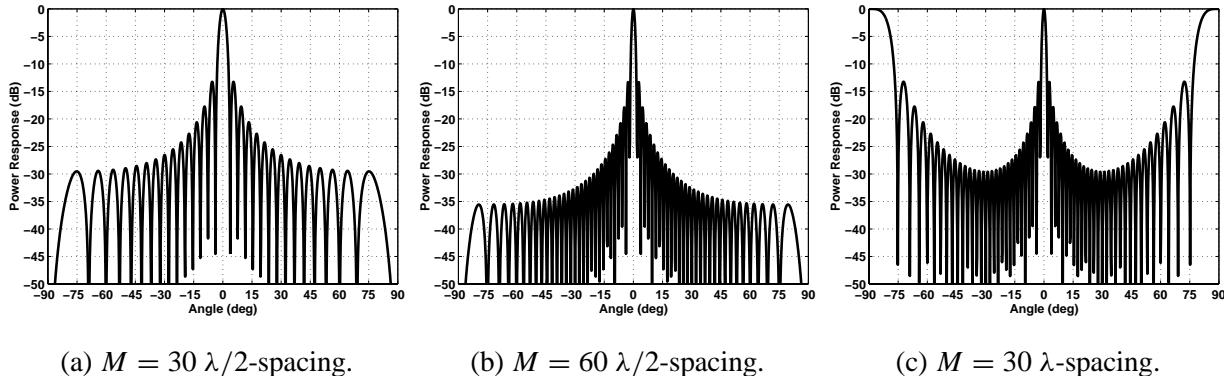


Figure 11.4d: Beampatterns for the spatial matched filters steered to $\phi = 0^\circ$ with (a) $M = 30 \lambda/2$ -spaced elements, (b) $M = 60 \lambda/2$ -spaced elements and (c) $M = 30 \lambda$ -spaced elements.

```

ylabel('Power Response (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca, 'fontweight', 'bold', 'fontsize', 18);
axis([-90 90 -50 0]);
grid
set(gca, 'xtick', -90:15:90);

% Plot out the beampattern in decibels
figure
plot(phi, 20*log10(abs(C_bp2)), 'b', 'linewidth', 2.5);
xlabel('Angle (deg)', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Power Response (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca, 'fontweight', 'bold', 'fontsize', 18);
axis([-90 90 -50 0]);
grid
set(gca, 'xtick', -90:15:90);

% Plot out the beampattern in decibels
figure
plot(phi, 20*log10(abs(C_bp3)), 'b', 'linewidth', 2.5);
xlabel('Angle (deg)', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Power Response (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca, 'fontweight', 'bold', 'fontsize', 18);
axis([-90 90 -50 0]);
grid
set(gca, 'xtick', -90:15:90);

```

- 11.5** This problem is a MATLAB based problem. Below is the MATLAB code and four output figures. Note that for this problem the signals should be at $\phi = 0^\circ$ and $\phi = 3^\circ$. If your text in Problem 11.4 does not have this correction, please pass this along to the students.

```

%
% Problem 11.5
%
clear

```

```

M0 = 60; % number of elements for a filled array
spacing = 0.5; % element spacing in wavelengths
N = 1000; % number of samples

% The angles in degrees of the two spatially propagating signals
phi1 = 0;
phi2 = 3;

% The power levels of the signals and the noise
P1_db = 20;
P2_db = 20;
Pn = 0;

% Generate the 45 randomly selected elements (75% thinning)
elements_used = 0.75;
M = elements_used*M0;
r = rand(M0,1);
[dummy,index] = sort(r);
elements = sort(index(1:(elements_used*M0)))';

% Compute the array response vectors of the two signals
% (subtract one from elements to have first element be reference (0))
v1 = exp(-j*2*pi*spacing*[elements-1]'*sin(phi1*pi/180))/sqrt(M);
v2 = exp(-j*2*pi*spacing*[elements-1]'*sin(phi2*pi/180))/sqrt(M);

% Generate the signals 1 and 2 and the noise
s1 = (10^(P1_db/20)) * v1 * ((randn(1,N) + j*randn(1,N))/sqrt(2));
s2 = (10^(P2_db/20)) * v2 * ((randn(1,N) + j*randn(1,N))/sqrt(2));
w = (10^(Pn/20)) * (randn(M,N) + j*randn(M,N))/sqrt(2); % Noise

% Add signals and noise to get the total array signal
x = s1 + s2 + w;

% Generate the spatial matched filters to compute the steered
% response for angles from -90 to 90 degrees
phi = -90:0.2:90;
C_smf = exp(-j*2*pi*spacing*[elements-1]'*sin(phi*pi/180))/sqrt(M);

% Compute the steered response
R_steer_est = mean(abs(C_smf' * x).^2,2);

% Plot out the steered response in decibels
figure
plot(phi,10*log10(R_steer_est),'b','linewidth',2.5);
xlabel('Angle (deg)', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Steered Response (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca, 'fontweight', 'bold', 'fontsize', 18);
axis([-90 90 -5 25]);

```

```

grid
set(gca,'xtick',-90:15:90);

%%%%%%%%%%%%%
%
% Repeat the same process for 50% array thinning
%
%%%%%%%%%%%%%
clear

M0 = 60;                      % number of elements for a filled array
spacing = 0.5;                  % element spacing in wavelengths
N = 1000;                      % number of samples

% The angles in degrees of the two spatially propagating signals
phi1 = 0;
phi2 = 3;

% The power levels of the signals and the noise
P1_db = 20;
P2_db = 20;
Pn = 0;

% Generate the 30 randomly selected elements (50% thinning)
elements_used = 0.50;
M = elements_used*M0;
r = rand(M0,1);
[dummy,index] = sort(r);
elements = sort(index(1:(elements_used*M0)))';

% Compute the array response vectors of the two signals
% (subtract one from elements to have first element be reference (0))
v1 = exp(-j*2*pi*spacing*[elements-1]'*sin(phi1*pi/180))/sqrt(M);
v2 = exp(-j*2*pi*spacing*[elements-1]'*sin(phi2*pi/180))/sqrt(M);

% Generate the signals 1 and 2 and the noise
s1 = (10^(P1_db/20)) * v1 * ((randn(1,N) + j*randn(1,N))/sqrt(2));
s2 = (10^(P2_db/20)) * v2 * ((randn(1,N) + j*randn(1,N))/sqrt(2));
w = (10^(Pn/20)) * (randn(M,N) + j*randn(M,N))/sqrt(2); % Noise

% Add signals and noise to get the total array signal
x = s1 + s2 + w;

% Generate the spatial matched filters to compute the steered
% response for angles from -90 to 90 degrees
phi = -90:0.2:90;
C_smf = exp(-j*2*pi*spacing*[elements-1]'*sin(phi*pi/180))/sqrt(M);

% Compute the steered response

```

```

R_steer_est = mean(abs(C_smf' * x).^2,2);

% Plot out the steered response in decibels
figure
plot(phi,10*log10(R_steer_est),'b','linewidth',2.5);
xlabel('Angle (deg)', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Steered Response (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca,'fontweight','bold','fontsize',18);
axis([-90 90 -5 25]);
grid
set(gca,'xtick',-90:15:90);

%%%%%%%%%%%%%
%
% Generate 3 randomly thinned arrays both at 50% and 75% and compute
% the beampatterns for the spatial matched filter at 0 degrees
%
%%%%%%%%%%%%%
clear

M0 = 60; % number of elements for a filled array
spacing = 0.5; % element spacing in wavelengths
phi_steer = 0; % steering direction for beamformer
phi = -90:0.2:90; % angles to compute the beampattern

% Generate the 30 randomly selected elements (75% thinning)
elements_used = 0.75;
M = elements_used*M0;

% Beamformer 1 with 75% thinning and its beampattern
r = rand(M0,1);
[dummy,index] = sort(r);
elements = sort(index(1:(elements_used*M0)))';
c1_75 = exp(-j*2*pi*spacing*[elements-1]'*sin(phi_steer*pi/180))/sqrt(M);
V1_75 = exp(-j*2*pi*spacing*[elements-1]'*sin(phi*pi/180))/sqrt(M);
C1_75_bp = V1_75'*c1_75;

% Beamformer 2 with 75% thinning and its beampattern
r = rand(M0,1);
[dummy,index] = sort(r);
elements = sort(index(1:(elements_used*M0)))';
c2_75 = exp(-j*2*pi*spacing*[elements-1]'*sin(phi_steer*pi/180))/sqrt(M);
V2_75 = exp(-j*2*pi*spacing*[elements-1]'*sin(phi*pi/180))/sqrt(M);
C2_75_bp = V2_75'*c2_75;

% Beamformer 3 with 75% thinning and its beampattern
r = rand(M0,1);
[dummy,index] = sort(r);
elements = sort(index(1:(elements_used*M0)))';

```

```

c3_75 = exp(-j*2*pi*spacing*[elements-1]>*sin(phi_steer*pi/180))/sqrt(M);
V3_75 = exp(-j*2*pi*spacing*[elements-1]>*sin(phi*pi/180))/sqrt(M);
C3_75_bp = V3_75'*c3_75;

% Plot out the beampatterns in decibels
figure
plot(phi,10*log10(abs(C1_75_bp).^2),'b','linewidth',2.5);
xlabel('Angle (deg)', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Power Response (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca,'fontweight','bold','fontsize',18);
axis([-90 90 -50 0]);
grid
set(gca,'xtick',-90:15:90);

figure
plot(phi,10*log10(abs(C2_75_bp).^2),'b','linewidth',2.5);
xlabel('Angle (deg)', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Power Response (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca,'fontweight','bold','fontsize',18);
axis([-90 90 -50 0]);
grid
set(gca,'xtick',-90:15:90);

figure
plot(phi,10*log10(abs(C3_75_bp).^2),'b','linewidth',2.5);
xlabel('Angle (deg)', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Power Response (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca,'fontweight','bold','fontsize',18);
axis([-90 90 -50 0]);
grid
set(gca,'xtick',-90:15:90);

% Generate the 30 randomly selected elements (50% thinning)
elements_used = 0.50;
M = elements_used*M0;

% Beamformer 1 with 50% thinning and its beampattern
r = rand(M0,1);
[dummy,index] = sort(r);
elements = sort(index(1:(elements_used*M0)))';
c1_50 = exp(-j*2*pi*spacing*[elements-1]*sin(phi_steer*pi/180))/sqrt(M);
V1_50 = exp(-j*2*pi*spacing*[elements-1]*sin(phi*pi/180))/sqrt(M);
C1_50_bp = V1_50'*c1_50;

% Beamformer 2 with 50% thinning and its beampattern
r = rand(M0,1);
[dummy,index] = sort(r);
elements = sort(index(1:(elements_used*M0)))';
c2_50 = exp(-j*2*pi*spacing*[elements-1]*sin(phi_steer*pi/180))/sqrt(M);

```

```

V2_50 = exp(-j*2*pi*spacing*[elements-1] *sin(phi*pi/180))/sqrt(M);
C2_50_bp = V2_50'*c2_50;

% Beamformer 3 with 50% thinning and its beampattern
r = rand(M0,1);
[dummy,index] = sort(r);
elements = sort(index(1:(elements_used*M0)))';
c3_50 = exp(-j*2*pi*spacing*[elements-1] *sin(phi_steer*pi/180))/sqrt(M);
V3_50 = exp(-j*2*pi*spacing*[elements-1] *sin(phi*pi/180))/sqrt(M);
C3_50_bp = V3_50'*c3_50;

% Plot out the beampatterns in decibels
figure
plot(phi,10*log10(abs(C1_50_bp).^2),'b','linewidth',2.5);
xlabel('Angle (deg)', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Power Response (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca,'fontweight','bold','fontsize',18);
axis([-90 90 -50 0]);
grid
set(gca,'xtick',-90:15:90);

figure
plot(phi,10*log10(abs(C2_50_bp).^2),'b','linewidth',2.5);
xlabel('Angle (deg)', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Power Response (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca,'fontweight','bold','fontsize',18);
axis([-90 90 -50 0]);
grid
set(gca,'xtick',-90:15:90);

figure
plot(phi,10*log10(abs(C3_50_bp).^2),'b','linewidth',2.5);
xlabel('Angle (deg)', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Power Response (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca,'fontweight','bold','fontsize',18);
axis([-90 90 -50 0]);
grid
set(gca,'xtick',-90:15:90);

```

The steered responses for the case of 75% (45 element) and 50% (30 element) thinned arrays are shown in Figures 11.5a and 11.5b. In both cases we are able to resolve the two signals at $\phi = 0^\circ$ and $\phi = 3^\circ$ since the aperture is equal to the $M = 60 \lambda/2$ -spaced array. Due to the reduced number of elements in the random spacing the background levels in the steered response are raised. The beampatterns for the three different random spacings for the 75% and 50% thinning are shown in Figures 11.5c and 11.5d. Here, we see that the increase in the background level was due to the elevated sidelobes that get worse the more thinning is done. In addition, we see that the sidelobes exhibit random behavior caused by the random thinning of the arrays.

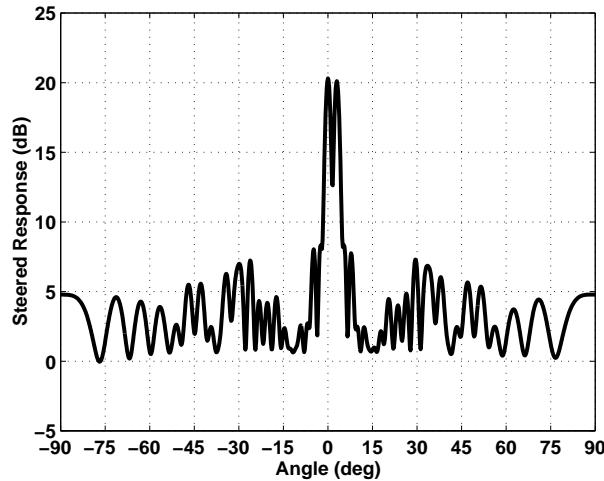


Figure 11.5a: Steered responses for scenario in Problem 11.4(a) for an array with $M = 60 \lambda/2$ -spaced element array thinned by 75% with signals at $\phi = 0^\circ$ and $\phi = 3^\circ$ both with powers of 20 dB.

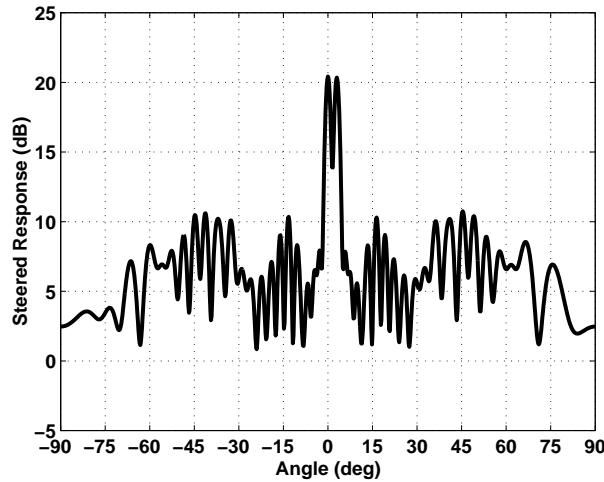
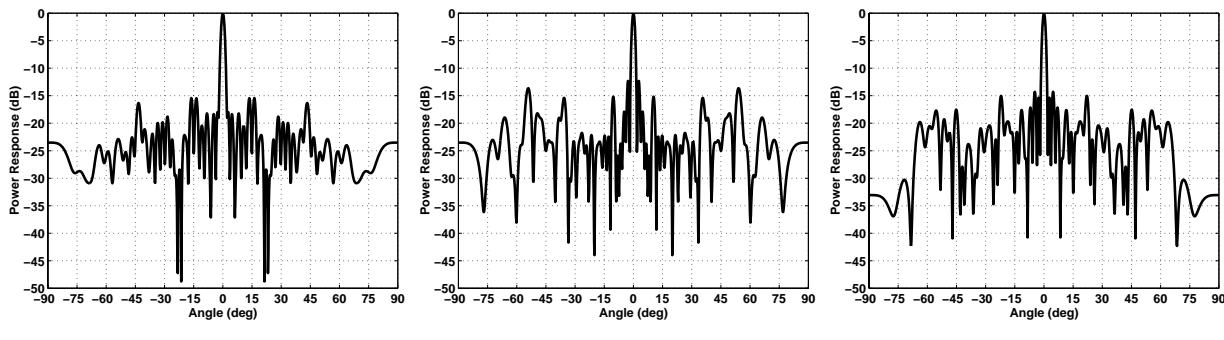


Figure 11.5b: Steered responses for scenario in Problem 11.4(a) for an array with $M = 60 \lambda/2$ -spaced element array thinned by 50% with signals at $\phi = 0^\circ$ and $\phi = 3^\circ$ both with powers of 20 dB.



(a) 1st thinned array.

(b) 2nd thinned array.

(c) 3rd thinned array.

Figure 11.5c: Beampatterns for three different randomly thinned $M = 60 \lambda/2$ -spaced element array thinned by 75% ($M = 45$ elements).

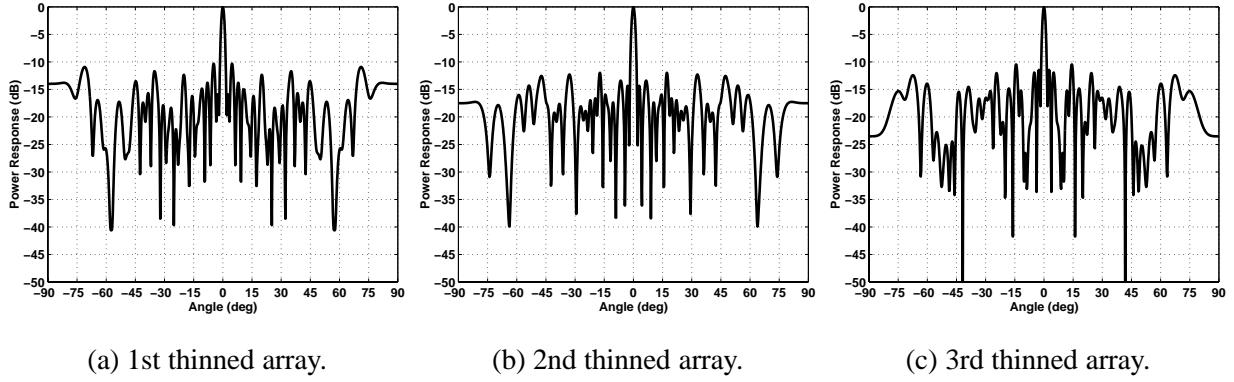


Figure 11.5d: Beampatterns for three different randomly thinned $M = 60 \lambda/2$ -spaced element array thinned by 50% ($M = 30$ elements).

- 11.6** We have an $M = 20$ element array whose even-numbered elements have a gain of 1 and whose odd-numbered elements have a gain of $3/4$. For an array whose elements all have equal gain, the beamforming gain is simply the ratio of the array and element SNR

$$G_{\text{bf}} = \frac{\text{SNR}_{\text{array}}}{\text{SNR}_{\text{element}}}$$

Since the signal and noise powers at the array output are

$$P_s = (M\sigma_s)^2 = M^2\sigma_s^2$$

$$P_n = E \left\{ \left| \sum_{m=1}^M w_m(n) \right|^2 \right\} = M\sigma_w^2$$

Then the SNR at the array output is

$$\text{SNR}_{\text{array}} = \frac{P_s}{P_n} = M \frac{\sigma_s^2}{\sigma_w^2}$$

while the element level SNR is

$$\text{SNR}_{\text{element}} = \frac{\sigma_s^2}{\sigma_w^2}$$

so that the beamforming gain is given by

$$G_{\text{bf}} = M$$

Now for the scenario described above with unequal gain on the elements of the array with half the elements having a gain of 1 and half having a gain of $3/4$. The signal power at the output of the array is

$$\begin{aligned} P_s &= \left(\frac{M}{2}\sigma_s + \frac{M}{2}\frac{3}{4}\sigma_s \right)^2 \\ &= \left(\frac{7}{8}M\sigma_s \right)^2 \\ &= \frac{49}{64}M^2 \end{aligned}$$

The noise power is as before

$$P_n = M\sigma_w^2$$

Therefore, the SNR at the array output is

$$\text{SNR}_{\text{array}} = \frac{P_s}{P_n} = \frac{49}{64}M \frac{\sigma_s^2}{\sigma_w^2} = \frac{49}{64}M \cdot \text{SNR}_{\text{element}}$$

and the array gain is

$$G_{\text{bf}} = \frac{49}{64}M$$

11.7 From (11.3.15), the optimum beamformer weight vector is given by

$$\mathbf{c}_o = \frac{\mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)}{\mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)}$$

As was pointed out in the text the optimum beamformer can also be formulated as the constrained minimization problem in (11.3.17)

$$\min \mathbf{c}^H \mathbf{R}_{i+n} \mathbf{c} \quad \text{subject to} \quad \mathbf{c}^H \mathbf{v}(\phi_s) = 1$$

we use Lagrange multipliers from Appendix B.2. First, we define the two functions

$$f(\mathbf{c}) = \mathbf{c}^H \mathbf{R}_{i+n} \mathbf{c} \quad g(\mathbf{c}) = \mathbf{c}^H \mathbf{v}(\phi_s) - 1$$

The Lagrangian function is then

$$\begin{aligned} \mathcal{L}(\mathbf{c}, \lambda) &= f(\mathbf{c}) + \lambda g(\mathbf{c}) \\ &= \mathbf{c}^H \mathbf{R}_{i+n} \mathbf{c} + \lambda [\mathbf{c}^H \mathbf{v}(\phi_s) - 1] \end{aligned}$$

where λ is the Lagrange multiplier. We then want to minimize the Lagrangian function with respect to both \mathbf{c} and λ . Taking the gradient of $\mathcal{L}(\mathbf{c}, \lambda)$ with respect to \mathbf{c} and setting it to zero, we have

$$\nabla_{\mathbf{c}} \mathcal{L}(\mathbf{c}, \lambda) = \mathbf{c}^H \mathbf{R}_{i+n} + \lambda \mathbf{v}^H(\phi_s) = 0$$

$$\mathbf{c} = -\lambda \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s) \tag{1}$$

Similarly, taking the gradient of $\mathcal{L}(\mathbf{c}, \lambda)$ with respect to λ and setting it to zero

$$\nabla_{\lambda} \mathcal{L}(\mathbf{c}, \lambda) = \mathbf{c}^H \mathbf{v}(\phi_s) - 1 = 0 \tag{2}$$

Substituting for (1) into (2), yields

$$-\lambda \mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s) - 1 = 0$$

and solving for λ

$$\lambda = \frac{-1}{\mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)} \tag{3}$$

Substituting (3) back into (1), we have the minimum variance or optimum beamformer weight vector for the look-direction ϕ_s

$$\mathbf{c}_o = \frac{\mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)}{\mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)}$$

11.8 This problem is a MATLAB based problem. Below is the MATLAB code and one output figure. Use a total of $M = 30 \lambda/2$ -spaced elements if not stated in your text. If your text does not have this correction, please pass this along to the students.

```

%
% Problem 11.8
%
clear

M = 30;                      % number of elements
spacing = 0.5;                % element spacing in wavelengths

% The angles in degrees of the two spatial interference sources
phi_i1 = 45;
phi_i2 = 20;

% The power levels of the signals and the noise
P_i1_db = 30;                 % power of interference source 1 (dB)
P_i2_db = 15;                  % power of interference source 2 (dB)
sigma_w = 1;                   % standard deviation of noise

% Compute the array response vectors of the two signals
v_i1 = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi_i1*pi/180))/sqrt(M);
v_i2 = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi_i2*pi/180))/sqrt(M);

% Compute the covariance matrix of the interference-plus-noise
R_in = (10^(P_i1_db/10))*v_i1*v_i1' + (10^(P_i2_db/10))*v_i2*v_i2' + ...
        (sigma_w^2) * eye(M);

% Generate the spatial matched filters to compute the steered
% response for angles from -90 to 90 degrees
phi = -90:90;

for n = 1:length(phi)
    v0 = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi(n)*pi/180))/sqrt(M);
    c_unnorm = R_in\v0;
    norm_mvdr = 1/(c_unnorm'*v0);
    norm_amf = 1/sqrt(c_unnorm'*v0);
    norm_noise = 1/sqrt(v0'*inv(R_in).^2*v0);
    c_mvdr = norm_mvdr * c_unnorm;
    c_amf = norm_amf * c_unnorm;
    c_noise = norm_noise * c_unnorm;
    R_steer_mvdr(n) = c_mvdr' * R_in * c_mvdr;
    R_steer_amf(n) = c_amf' * R_in * c_amf;
    R_steer_noise(n) = c_noise' * R_in * c_noise;
end

% Plot out the steered responses in decibels
figure

```

```

plot(phi,10*log10(abs(R_steer_mvdr)), 'b', 'linewidth', 2.5);
xlabel('Angle (deg)', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Steered Response (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca, 'fontweight', 'bold', 'fontsize', 18);
axis([-90 90 -32 32]);
set(gca, 'xtick', -90:30:90);
set(gca, 'ytick', -30:10:30);
grid
set(gca, 'xtick', -90:15:90);

figure
plot(phi,10*log10(abs(R_steer_amf)), 'b', 'linewidth', 2.5);
xlabel('Angle (deg)', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Steered Response (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca, 'fontweight', 'bold', 'fontsize', 18);
axis([-90 90 -32 32]);
set(gca, 'xtick', -90:30:90);
set(gca, 'ytick', -30:10:30);
grid
set(gca, 'xtick', -90:15:90);

figure
plot(phi,10*log10(abs(R_steer_noise)), 'b', 'linewidth', 2.5);
xlabel('Angle (deg)', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Steered Response (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca, 'fontweight', 'bold', 'fontsize', 18);
axis([-90 90 -32 32]);
set(gca, 'xtick', -90:30:90);
set(gca, 'ytick', -30:10:30);
grid
set(gca, 'xtick', -90:15:90);

```

In this problem, the optimum beamformers with the three different normalizations are computed for the scenario with two interference sources at $\phi = 45^\circ$ and $\phi = 20^\circ$ with powers of 30 and 15 dB, respectively. The steered response is computed for the three different normalizations and shown in Figure 11.8a. The steered response for the MVDR-normed optimum beamformer has the correct power estimates for both of the interference sources as well as the noise background at 0 dB. Thus, the MVDR normalization generates a spatial power spectrum. On the other hand, the output level of the AMF-normed optimum beamformer is constant by design (unit-gain on interference-plus-noise). This normalization is particularly useful if we want to set a detection threshold independent of angle to generate a constant probability of detection. The steered response for the optimum beamformer with unit-gain on noise normalization produces the reciprocal of the MVDR-normed steered response.

11.9

11.10 Starting with the correlation matrix with the signal at ϕ_s with amplitude σ_s present

$$\mathbf{R}_x = \mathbf{R}_{i+n} + \sigma_s^2 \mathbf{v}(\phi_s) \mathbf{v}^H(\phi_s) \quad (1)$$

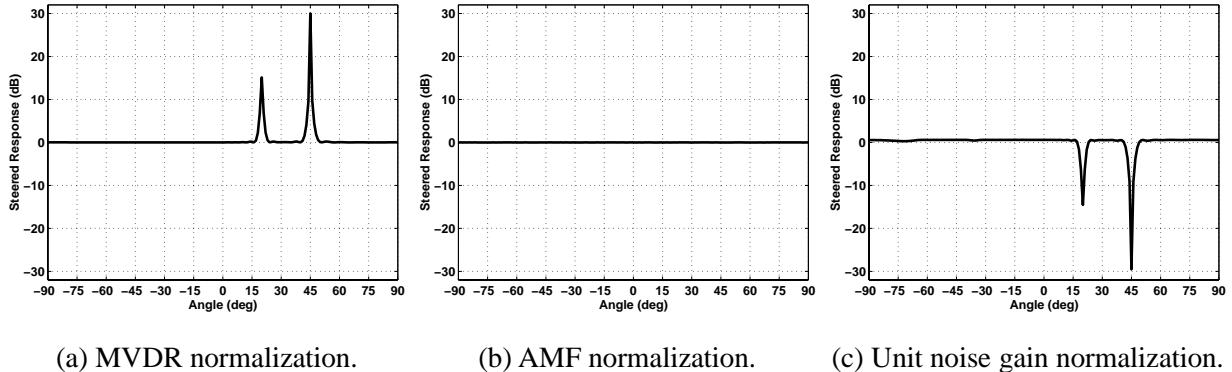


Figure 11.8: Steered responses for scenario in Problem 11.8 for optimum beamformers with (a) MVDR , (b) AMF and (c) unit-gain on noise normalizations.

we want to show that the optimum beamformer weights do not change when the signal is present in the case of no mismatch, i.e.,

$$\mathbf{c}_o = \frac{\mathbf{R}_x^{-1}\mathbf{v}(\phi_s)}{\mathbf{v}^H(\phi_s)\mathbf{R}_x^{-1}\mathbf{v}(\phi_s)} = \frac{\mathbf{R}_{i+n}^{-1}\mathbf{v}(\phi_s)}{\mathbf{v}^H(\phi_s)\mathbf{R}_{i+n}^{-1}\mathbf{v}(\phi_s)}$$

Using the matrix inversion lemma from Appendix A, the inverse correlation matrix from (1) is

$$\mathbf{R}_x^{-1} = \mathbf{R}_{i+n}^{-1} - \frac{\sigma_s^2 \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s) \mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1}}{1 + \sigma_s^2 \mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)}$$

Multiplying by $\mathbf{v}(\phi_s)$, we have

$$\begin{aligned} \mathbf{R}_x^{-1}\mathbf{v}(\phi_s) &= \mathbf{R}_{i+n}^{-1}\mathbf{v}(\phi_s) - \frac{\sigma_s^2 \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s) \mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)}{1 + \sigma_s^2 \mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)} \\ &= \mathbf{R}_{i+n}^{-1}\mathbf{v}(\phi_s) \left[1 - \frac{\sigma_s^2 \mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)}{1 + \sigma_s^2 \mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)} \right] \\ &= \mathbf{R}_{i+n}^{-1}\mathbf{v}(\phi_s) \frac{1}{1 + \sigma_s^2 \mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)} \end{aligned} \quad (2)$$

Similarly, we can compute $\mathbf{v}^H(\phi_s)\mathbf{R}_x^{-1}\mathbf{v}(\phi_s)$, by multiplying (1) by $\mathbf{v}(\phi_s)$

$$\mathbf{v}^H(\phi_s)\mathbf{R}_x^{-1}\mathbf{v}(\phi_s) = \frac{\mathbf{v}^H(\phi_s)\mathbf{R}_{i+n}^{-1}\mathbf{v}(\phi_s)}{1 + \sigma_s^2 \mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)} \quad (3)$$

Substituting (1) and (2) for the numerator and denominator of the optimum beamformer, we have

$$\begin{aligned} \mathbf{c}_o &= \frac{\mathbf{R}_x^{-1}\mathbf{v}(\phi_s)}{\mathbf{v}^H(\phi_s)\mathbf{R}_x^{-1}\mathbf{v}(\phi_s)} \\ &= \frac{\mathbf{R}_{i+n}^{-1}\mathbf{v}(\phi_s)}{1 + \sigma_s^2 \mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)} \cdot \frac{1 + \sigma_s^2 \mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)}{\mathbf{v}^H(\phi_s)\mathbf{R}_{i+n}^{-1}\mathbf{v}(\phi_s)} \\ &= \frac{\mathbf{R}_{i+n}^{-1}\mathbf{v}(\phi_s)}{\mathbf{v}^H(\phi_s)\mathbf{R}_{i+n}^{-1}\mathbf{v}(\phi_s)} \end{aligned}$$

Therefore, in the case of no signal mismatch, the optimum beamformer is not affected by the presence of the signal of interest in the correlation matrix. Note that in practice it may not be possible to perfectly match the signal of interest which can have significant performance implications.

- 11.11** This problem is a MATLAB based problem. Below is the MATLAB code and two output figures. The code is broken up into parts (a) and (b) and presented sequentially along with the accompanying figures.

(a) In this part of the problem, we examine the effect of mismatch when the signal is not in the correlation matrix. The output powers for the cases of no mismatch (signal at $\phi = 0^\circ$) and mismatch in angle (signal at $\phi = -1^\circ$) shown by the dashed and solid lines in Figure 11.11a. The mismatch loss is simply the difference between these two lines in decibels (or their ratio). From the figure, we observe an approximate loss of -3 dB for all angles independent of the signal-to-noise ratio of the signal since the signal is not included in the correlation matrix. This result can be compared to that predicted by (11.4.12).

```
%  
% Problem 11.11  
%  
clear  
  
M = 50; % number of elements  
spacing = 0.5; % element spacing in wavelengths  
  
% The angles in degrees of the three spatially interference sources  
phi_i1 = 5;  
phi_i2 = 20;  
phi_i3 = -30;  
  
% The power levels of the signals and the noise  
P_i1_db = 25; % power of interference source 1 (dB)  
P_i2_db = 30; % power of interference source 2 (dB)  
P_i3_db = 50; % power of interference source 3 (dB)  
sigma_w = 1; % standard deviation of noise  
  
% Compute the array response vectors of the three signals  
v_i1 = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi_i1*pi/180))/sqrt(M);  
v_i2 = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi_i2*pi/180))/sqrt(M);  
v_i3 = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi_i3*pi/180))/sqrt(M);  
  
% Compute the covariance matrix of the interference-plus-noise  
R_in = (10^(P_i1_db/10))*v_i1*v_i1' + (10^(P_i2_db/10))*v_i2*v_i2' + ...  
       (10^(P_i3_db/10))*v_i3*v_i3' + (sigma_w^2) * eye(M);  
  
% Specify the signal of interest  
phi_s = -1; % angle of signal of interest (deg)  
SNR_db = 0:30; % signal-to-noise ratios (dB)  
  
%%%%%%%%%%%%%  
%  
% Part A  
%
```

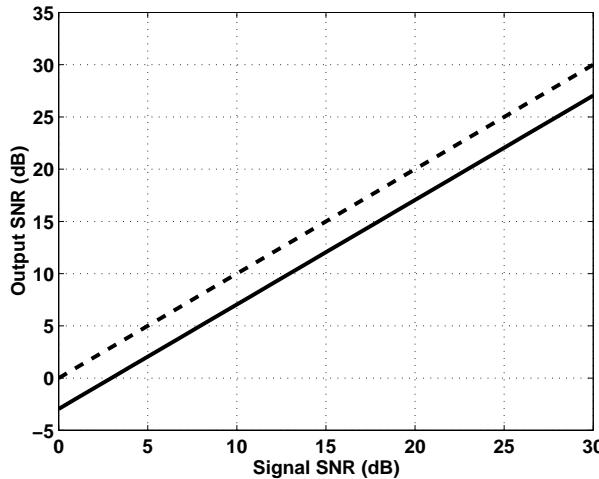


Figure 11.11a: Output power as a function of signal-to-noise ratio for the case of no mismatch ($\phi = 0^\circ$, dashed line) and angle mismatch ($\phi = -1^\circ$, solid line) when the signal is not contained in the correlation matrix.

```
%%%%%
% Compute the optimum beamformer to the desired angle
phi0 = 0; % angle to which we are steering (deg)
v0 = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi0*pi/180))/sqrt(M);
c_unnorm = R_in\v0; % unnormalized optimum beamformer
c_opt = c_unnorm/(v0'*c_unnorm); % MVDR norm on optimum beamformer

% Loop on SNR to compute the signal mismatch losses
for n = 1:length(SNR_db)
    v_s = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi_s*pi/180))/sqrt(M);
    x_s = (10^(SNR_db(n)/20))*v_s; % actual array signal at SNR
    x0 = (10^(SNR_db(n)/20))*v0; % array signal at the steering angle
    P_s(n) = abs(c_opt'*x_s)^2; % output power of actual signal
    P0(n) = abs(c_opt'*x0)^2; % output power for signal at phi0
    L_sm(n) = P_s(n)/P0(n); % mismatch loss
end

% Plot out the output SNR in decibels vs. the signal SNR
figure
plot(SNR_db,10*log10(abs(P_s)), 'b',...
    SNR_db,10*log10(abs(P0)), 'r--', 'linewidth', 2.5);
xlabel('Signal SNR (dB)', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Output SNR (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca, 'fontweight', 'bold', 'fontsize', 18);
axis([0 30 -5 35]);
grid
```

- (b) In this part of the problem, we examine the effect of mismatch when the signal is in the correlation matrix. Here, we observe that, as predicted, the loss is strongly dependent on the signal-to-noise ratio. The stronger the signal, the larger the loss incurred is. This loss should be compared to the loss predicted by (11.4.17). Note that the loss computed and plotted in this problem is combination of the mismatch losses L_{sm} and L_{sp} .

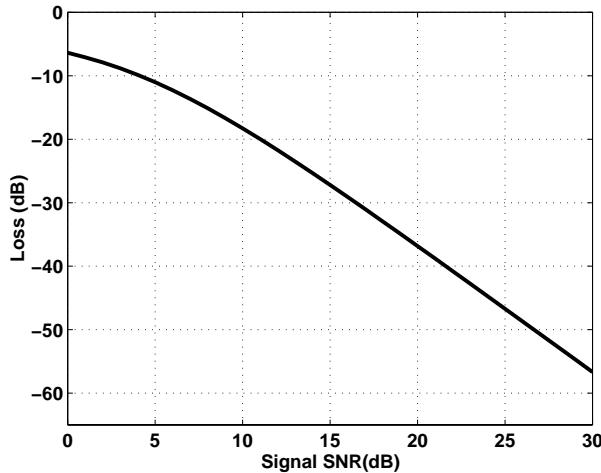


Figure 11.11b: Mismatch loss versus signal-to-noise ratio for a mismatched steering direction ($\phi = -1^\circ$) when the signal is contained in the correlation matrix .

```
%%%%%%%
%
% Part B
%
%%%%%%%
%
% Loop on SNR to compute the signal mismatch losses
for n = 1:length(SNR_db)
    v_s = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi_s*pi/180))/sqrt(M);
    x_s = (10^(SNR_db(n)/20))*v_s; % actual array signal at SNR
    x0 = (10^(SNR_db(n)/20))*v0; % array signal at the steering angle
    R_x = R_in + x_s*x_s'; % correlation matrix containing signal
    c_unnorm = R_x\v0; % unnormalized optimum beamformer
    c = c_unnorm/(v0'*c_unnorm); % MVDR norm on optimum beamformer
    P_s(n) = abs(c'*x_s)^2; % output power of actual signal
    P0(n) = abs(c'*x0)^2; % output power for signal at phi0
    L_mismatch(n) = P_s(n)/P0(n); % mismatch loss (losses due to
                                    % SINR loss, signal present in
                                    % correlation matrix, and mismatch
end

% Plot out the mismatch loss versus the signal SNR in decibels
figure
plot(SNR_db,10*log10(L_mismatch),'b','linewidth',2.5);
xlabel('Signal SNR(dB)', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Loss (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca,'fontweight','bold','fontsize',18);
axis([0 30 -65 0]);
grid
```

- 11.12** This problem is a MATLAB based problem. Below is the MATLAB code and two output figures. The code is broken up into parts (a) and (b) and presented sequentially along with the accompanying figures. The signal

level should be $M\sigma_s^2 = 20$ dB instead of 30 dB. In both parts (a) and (b), the problem should read: "Compute and plot the mismatch loss for diagonal loading levels of 0 to 20 dB". If your text does not have this correction, please pass this along to the students.

(a) In this part of the problem, we examine the effect of diagonal loading on the mismatch loss when the signal is not in the correlation matrix. The output powers for the cases of no mismatch (signal at $\phi = 0^\circ$, dashed line) and mismatch in angle (signal at $\phi = -1^\circ$, solid line) are plotted in Figure 11.12a. The mismatch loss is simply the difference between these two lines in decibels (or their ratio). From the figure, we observe an approximate loss of -3 dB for all diagonal loading levels since the signal is not included in the correlation matrix.

```

%
% Problem 11.12
%
clear

M = 50; % number of elements
spacing = 0.5; % element spacing in wavelengths

% The angles in degrees of the three spatially interference sources
phi_i1 = 5;
phi_i2 = 20;
phi_i3 = -30;

% The power levels of the signals and the noise
P_i1_db = 25; % power of interference source 1 (dB)
P_i2_db = 30; % power of interference source 2 (dB)
P_i3_db = 50; % power of interference source 3 (dB)
sigma_w = 1; % standard deviation of noise

% Compute the array response vectors of the three signals
v_i1 = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi_i1*pi/180))/sqrt(M);
v_i2 = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi_i2*pi/180))/sqrt(M);
v_i3 = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi_i3*pi/180))/sqrt(M);

% Compute the covariance matrix of the interference-plus-noise
R_in = (10^(P_i1_db/10))*v_i1*v_i1' + (10^(P_i2_db/10))*v_i2*v_i2' + ...
        (10^(P_i3_db/10))*v_i3*v_i3' + (sigma_w^2) * eye(M);

% Specify the signal of interest
phi_s = -1; % angle of signal of interest (deg)
SNR_db = 20; % signal-to-noise ratios (dB)

%%%%%%%%%%%%%
%
% Part A
%
%%%%%%%%%%%%%
% Compute the optimum beamformer to the desired angle
phi0 = 0; % angle to which we are steering (deg)
```

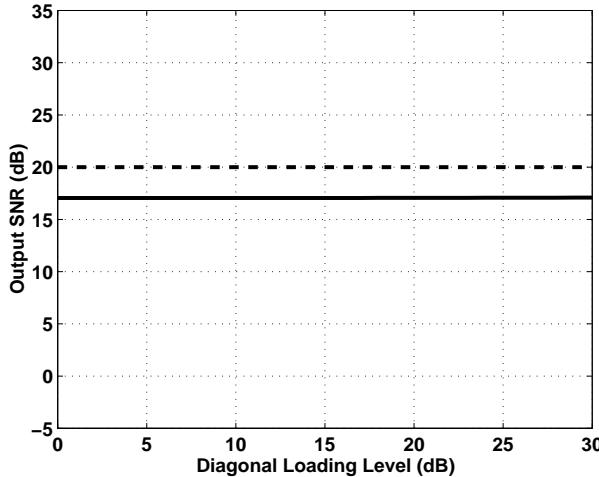


Figure 11.12a: Output power as a function of diagonal loading level for the case of no mismatch ($\phi = 0^\circ$, dashed line) and angle mismatch ($\phi = -1^\circ$, solid line) when the signal is not contained in the correlation matrix.

```

v0 = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi0*pi/180))/sqrt(M);

% Diagonal loading levels
DLL_db = 0:30; % diagonal loading levels (dB)

% Loop on SNR to compute the signal mismatch losses
for n = 1:length(DLL_db)
    v_s = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi_s*pi/180))/sqrt(M);
    x_s = (10^(SNR_db/20))*v_s; % actual array signal at SNR
    x0 = (10^(SNR_db/20))*v0; % array signal at the steering angle
    c_dl_unnorm = (R_in + (10^(DLL_db(n)/10))*eye(M))\v0;
    c_dl = c_dl_unnorm/(c_dl_unnorm'*v0);
    P_s(n) = abs(c_dl*x_s)^2; % output power of actual signal
    P0(n) = abs(c_dl*x0)^2; % output power for signal at phi0
    L_sm(n) = P_s(n)/P0(n); % mismatch loss
end

% Plot out the output powers versus diagonal loading level in decibels
figure
plot(DLL_db,10*log10(abs(P_s)), 'b',...
    DLL_db,10*log10(abs(P0)), 'r--', 'linewidth', 2.5);
xlabel('Diagonal Loading Level (dB)', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Output SNR (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca, 'fontweight', 'bold', 'fontsize', 18);
axis([0 30 -5 35]);
grid

```

(b) In this part of the problem, we examine the effect of diagonal loading on mismatch loss when the signal is in the correlation matrix. We observe that the more heavily we diagonally load the more signal we are able to protect from mismatch loss. This protection from mismatch loss when the signal is in the correlation matrix is achieved at the expense of interference cancellation performance in the case of diagonal loading. Note that the loss computed and plotted in this problem is combination of the mismatch losses L_{sm} and L_{sp} . Therefore, the

mismatch loss can never exceed the -3-dB level due to the steering direction mismatch loss that was observed in part (a).

```
%%%%%%%%%%%%%
%
% Part B
%
%%%%%%%%%%%%%
%
% Loop on SNR to compute the signal mismatch losses
for n = 1:length(DLL_db)
    v_s = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi_s*pi/180))/sqrt(M);
    x_s = (10^(SNR_db/20))*v_s;      % actual array signal at SNR
    x0 = (10^(SNR_db/20))*v0;        % array signal at the steering angle
    R_x = R_in + x_s*x_s';          % correlation matrix containing signal
    c_dl_unnorm = (R_x + (10^(DLL_db(n)/10))*eye(M))\v0;
    c_dl = c_dl_unnorm/(c_dl_unnorm'*v0);
    P_s(n) = abs(c_dl'*x_s)^2;       % output power of actual signal
    P0(n) = abs(c_dl'*x0)^2;         % output power for signal at phi0
    L_mismatch(n) = P_s(n)/P0(n);   % mismatch loss (losses due to
                                    % SINR loss, signal present in
                                    % correlation matrix, and mismatch
end

% Plot out the mismatch loss versus diagonal loading level in decibels
figure
plot(DLL_db,10*log10(L_mismatch),'b','linewidth',2.5);
xlabel('Diagonal Loading Level (dB)', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Loss (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca,'fontweight','bold','fontsize',18);
axis([0 30 -65 0]);
grid
```

- 11.13** The linearly constrained minimum variance (LCMV) beamformer weights \mathbf{c}_{lcmv} are found by solving the constrained minimization from (11.6.5)

$$\min \mathbf{c}^H \mathbf{R}_x \mathbf{c} \quad \text{subject to} \quad \mathbf{C}^H \mathbf{c} = \delta$$

where \mathbf{C} is the constraint matrix and δ is the vector of corresponding responses to the constraints. Using Lagrange multipliers from Appendix B.2, the LCMV beamformer weights are found to be

$$\mathbf{c}_{\text{lcmv}} = \mathbf{R}_x^{-1} \mathbf{C} (\mathbf{C}^H \mathbf{R}_x^{-1} \mathbf{C})^{-1} \delta$$

which minimizes the Lagrangian function. Using Lagrange multipliers from Appendix B.2, we define the following two functions

$$f(\mathbf{c}) = \mathbf{c}^H \mathbf{R}_x \mathbf{c} \quad g(\mathbf{c}) = \mathbf{C}^H \mathbf{c} - \delta$$

The Lagrangian function is then given by

$$\begin{aligned} \mathcal{L}(\mathbf{c}, \lambda) &= f(\mathbf{c}) + \lambda^H g(\mathbf{c}) \\ &= \mathbf{c}^H \mathbf{R}_x \mathbf{c} + \lambda^H [\mathbf{C}^H \mathbf{c} - \delta] \end{aligned}$$

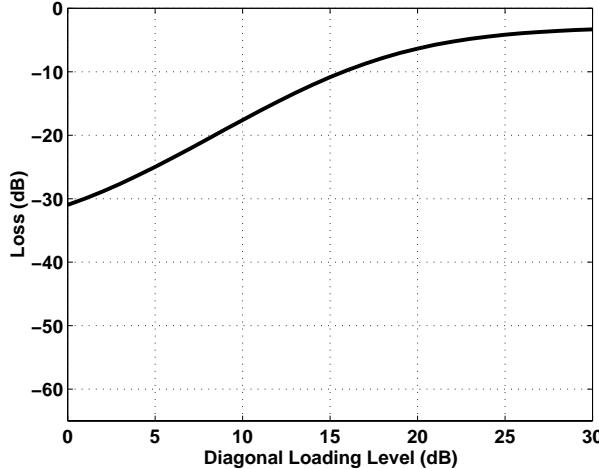


Figure 11.12b: Mismatch loss versus diagonal loading level for a mismatched steering direction ($\phi = -1^\circ$) when the signal is contained in the correlation matrix.

where λ is the vector of Lagrange multipliers whose length is equal to the number of constraints (same length as the constraint response vector δ). Taking the gradient of the Lagrangian function with respect to \mathbf{c} , we have

$$\nabla_{\mathbf{c}} \mathcal{L}(\mathbf{c}, \lambda) = \mathbf{R}_x \mathbf{c} + \mathbf{C} \lambda$$

For the method of steepest descent (Frost's algorithm), the adaptive weights at time $n+1$ are found by modifying the weights at time n by the gradient of the Lagrangian function scaled by the update factor μ

$$\begin{aligned} \mathbf{c}(n+1) &= \mathbf{c}(n) - \mu \nabla_{\mathbf{c}} \mathcal{L}(\mathbf{c}, \lambda) \\ &= \mathbf{c}(n) - \mu [\mathbf{R}_x \mathbf{c}(n) + \mathbf{C} \lambda(n)] \end{aligned} \quad (1)$$

To satisfy the constraint at time $n+1$, we require that

$$\delta = \mathbf{C}^H \mathbf{c}(n+1) = \mathbf{C}^H \mathbf{c}(n) - \mu \mathbf{C}^H \mathbf{R}_x \mathbf{c}(n) - \mu \mathbf{C}^H \mathbf{C} \lambda(n)$$

Solving for $\lambda(n)$, we have

$$\lambda(n) = \frac{1}{\mu} (\mathbf{C}^H \mathbf{C})^{-1} \mathbf{C}^H \mathbf{c}(n) - (\mathbf{C}^H \mathbf{C})^{-1} \mathbf{C}^H \mathbf{R}_x \mathbf{c}(n) - \frac{1}{\mu} (\mathbf{C}^H \mathbf{C})^{-1} \delta \quad (2)$$

Substituting the solution for $\lambda(n)$ from (2) back into the weight update equation in (1), we have

$$\begin{aligned} \mathbf{c}(n+1) &= \mathbf{c}(n) - \mu \mathbf{R}_x \mathbf{c}(n) \\ &\quad - \mu \mathbf{C} \left[\frac{1}{\mu} (\mathbf{C}^H \mathbf{C})^{-1} \mathbf{C}^H \mathbf{c}(n) - (\mathbf{C}^H \mathbf{C})^{-1} \mathbf{C}^H \mathbf{R}_x \mathbf{c}(n) - \frac{1}{\mu} (\mathbf{C}^H \mathbf{C})^{-1} \delta \right] \\ &= \mathbf{c}(n) - \mu \mathbf{R}_x \mathbf{c}(n) + \mu \mathbf{C} (\mathbf{C}^H \mathbf{C})^{-1} \mathbf{C}^H \mathbf{R}_x \mathbf{c}(n) \\ &\quad - \mathbf{C} (\mathbf{C}^H \mathbf{C})^{-1} \mathbf{C}^H \mathbf{c}(n) + \mathbf{C} (\mathbf{C}^H \mathbf{C})^{-1} \delta \\ &= \mathbf{c}(n) - \mu \left[\mathbf{I} - \mathbf{C} (\mathbf{C}^H \mathbf{C})^{-1} \mathbf{C}^H \right] \mathbf{R}_x \mathbf{c}(n) + \mathbf{C} (\mathbf{C}^H \mathbf{C})^{-1} [\delta - \mathbf{C}^H \mathbf{c}(n)] \end{aligned} \quad (3)$$

Let us now define the non-adaptive portion of the weight update equation

$$\mathbf{c}_{na} = \mathbf{C} (\mathbf{C}^H \mathbf{C})^{-1} \delta$$

and the projection matrix

$$\mathbf{P} = \mathbf{I} - \mathbf{C} (\mathbf{C}^H \mathbf{C})^{-1} \mathbf{C}^H$$

The weight update equation from (3) can then be written as

$$\begin{aligned}\mathbf{c}(n+1) &= \mathbf{c}(n) - \mu \mathbf{P} \mathbf{R}_x \mathbf{c}(n) - \mathbf{C} (\mathbf{C}^H \mathbf{C})^{-1} \mathbf{C}^H \mathbf{c}(n) + \mathbf{c}_{na} \\ &= \mathbf{c}_{na} + \mathbf{P} [\mathbf{c}(n) - \mu \mathbf{R}_x \mathbf{c}(n)]\end{aligned}\quad (4)$$

Equation (4) is a deterministic constrained gradient descent algorithm requiring knowledge of the correlation matrix \mathbf{R}_x . Since \mathbf{R}_x is unknown for any given time n , we can substitute the estimate

$$\widehat{\mathbf{R}}_x = \mathbf{x}(n) \mathbf{x}^H(n)$$

into (4) to obtain the weight update equation for Frost's algorithm with multiple linear constraints

$$\begin{aligned}\mathbf{c}(n+1) &= \mathbf{c}_{na} + \mathbf{P} [\mathbf{c}(n) - \mu \mathbf{x}(n) \mathbf{x}^H(n) \mathbf{c}(n)] \\ &= \mathbf{c}_{na} + \mathbf{P} [\mathbf{c}(n) - \mu \mathbf{x}(n) \mathbf{y}^*(n)]\end{aligned}\quad (5)$$

since $\mathbf{y}(n) = \mathbf{c}^H(n) \mathbf{x}(n)$.

- 11.14** The linearly constrained minimum variance (LCMV) beamformer weights \mathbf{c}_{lcmv} are found by solving the constrained minimization from (11.6.5)

$$\min \mathbf{c}^H \mathbf{R}_{i+n} \mathbf{c} \quad \text{subject to} \quad \mathbf{C}^H \mathbf{c} = \boldsymbol{\delta}$$

where \mathbf{C} is the constraint matrix and $\boldsymbol{\delta}$ is the vector of corresponding responses to the constraints. Using Lagrange multipliers from Appendix B.2, we define the following two functions

$$f(\mathbf{c}) = \mathbf{c}^H \mathbf{R}_{i+n} \mathbf{c} \quad g(\mathbf{c}) = \mathbf{C}^H \mathbf{c} - \boldsymbol{\delta}$$

The Lagrangian function is then given by

$$\begin{aligned}\mathcal{L}(\mathbf{c}, \boldsymbol{\lambda}) &= f(\mathbf{c}) + \boldsymbol{\lambda}^H g(\mathbf{c}) \\ &= \mathbf{c}^H \mathbf{R}_{i+n} \mathbf{c} + \boldsymbol{\lambda}^H [\mathbf{C}^H \mathbf{c} - \boldsymbol{\delta}]\end{aligned}$$

where $\boldsymbol{\lambda}$ is the vector of Lagrange multipliers whose length is equal to the number of constraints (same length as the constraint response vector $\boldsymbol{\delta}$). Taking the gradient of the Lagrangian function with respect to \mathbf{c} , we have

$$\nabla_{\mathbf{c}} \mathcal{L}(\mathbf{c}, \boldsymbol{\lambda}) = \mathbf{R}_{i+n} \mathbf{c} + \mathbf{C} \boldsymbol{\lambda}$$

Setting the gradient to zero in order to minimize the Lagrangian

$$\mathbf{c} = -\mathbf{R}_{i+n}^{-1} \mathbf{C} \boldsymbol{\lambda} \quad (1)$$

Now, we simply need to solve for the vector of Lagrange multipliers. Taking the gradient of the Lagrangian function with respect to $\boldsymbol{\lambda}$

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L}(\mathbf{c}, \boldsymbol{\lambda}) = \mathbf{C}^H \mathbf{c} - \boldsymbol{\delta}$$

Setting this equal to zero, we have

$$\mathbf{C}^H \mathbf{c} = \boldsymbol{\delta} \quad (2)$$

and substituting for \mathbf{c} from (1) into (2) and solving for $\boldsymbol{\lambda}$

$$\boldsymbol{\lambda} = -(\mathbf{C}^H \mathbf{R}_{i+n}^{-1} \mathbf{C})^{-1} \boldsymbol{\delta} \quad (3)$$

Substituting (3) back into (1), we have the LCMV beamformer weight vector

$$\mathbf{c}_{lcmv} = \mathbf{R}_{i+n}^{-1} \mathbf{C} (\mathbf{C}^H \mathbf{R}_{i+n}^{-1} \mathbf{C})^{-1} \boldsymbol{\delta}$$

11.15 In this problem, we study the performance of the sidelobe canceler. Let us start by defining the main channel signal as

$$x_{mc} = g_s s(n) + i_{mc}(n) + w_{mc}(n)$$

where $i_{mc}(n) = g_i i(n)$ is the interference signal $i(n)$ with a main channel gain g_i . The interference signal is assumed to be temporally uncorrelated with unit variance. The main channel thermal noise signal $w_{mc}(n)$ is also temporally uncorrelated and has a variance of σ_0^2 . Likewise, the signal in the auxiliary channel, consisting of an M -element $\lambda/2$ -spaced ULA, is given by

$$\mathbf{x}_a(n) = s(n) \mathbf{v}(\phi_s) + \sigma_i i(n) \mathbf{v}(\phi_i) + \mathbf{w}(n)$$

where $\mathbf{w}(n)$ is the the thermal noise signal in the array with a variance of σ_w^2 in each channel (element).

(a) The auxiliary channel correlation matrix is given by

$$\begin{aligned} \mathbf{R}_a &= E\{\mathbf{x}_a(n) \mathbf{x}_a^H(n)\} \\ &= P_s \mathbf{v}(\phi_s) \mathbf{v}^H(\phi_s) + \sigma_i^2 \mathbf{v}(\phi_i) \mathbf{v}^H(\phi_i) + \sigma_w^2 \mathbf{I} \end{aligned}$$

where $P_s = |s(n)|^2$. The cross-correlation vector between the auxiliary channels and the main channel is

$$\begin{aligned} \mathbf{r}_{ma} &= E\{\mathbf{x}_a(n) x_{mc}^*(n)\} \\ &= P_s g_s^* \mathbf{v}(\phi_s) + \sigma_i g_i^* \mathbf{v}(\phi_i) \end{aligned}$$

(b) The output signal from the sidelobe canceler is given by

$$y(n) = x_{mc}(n) - \mathbf{c}_a^H \mathbf{x}_a(n)$$

The interference-plus-noise portion of this signal, however, is given by

$$y_{i+n}(n) = i_{i+n}(n) + w_{i+n}(n) - \sigma_i i(n) \mathbf{c}_a^H \mathbf{v}(\phi_i) - \mathbf{c}_a^H \mathbf{w}(n)$$

The output power of the interference-plus-noise is thus

$$\begin{aligned} P_{i+n} &= E\{|y_{i+n}(n)|^2\} \\ &= g_i^2 + \sigma_0^2 + \sigma_i^2 |\mathbf{c}_a^H \mathbf{v}(\phi_i)|^2 + \sigma_w^2 \mathbf{c}_a^H \mathbf{c}_a - 2 \cdot \text{Real}\{\sigma_i g_i^* \mathbf{c}_a^H \mathbf{v}(\phi_i)\} \end{aligned} \quad (1)$$

Recall that

$$\mathbf{R}_a = \mathbf{R}_{i+n} + P_s \mathbf{v}(\phi_s) \mathbf{v}^H(\phi_s)$$

Here, we make the assumption that $P_s \ll \sigma_w^2 < \sigma_i^2$ so that the correlation matrix and cross-correlation vector from part (a) are

$$\mathbf{R}_a \approx \mathbf{R}_{i+n}$$

$$\mathbf{r}_{ma} \approx \sigma_i g_i^* \mathbf{v}(\phi_i)$$

Using the matrix inversion lemma from Appendix A, we have

$$\begin{aligned} \mathbf{R}_a^{-1} &= \frac{1}{\sigma_w^2} \mathbf{I} - \frac{(1/\sigma_w^2) \mathbf{I} \cdot \sigma_i^2 \mathbf{v}(\phi_i) \mathbf{v}^H(\phi_i) (1/\sigma_w^2) \mathbf{I}}{1 + \sigma_i^2 \mathbf{v}^H(\phi_i) (1/\sigma_w^2) \mathbf{I} \cdot \mathbf{v}(\phi_i)} \\ &= \frac{1}{\sigma_w^2} \left[\mathbf{I} - \frac{\sigma_i^2 \mathbf{v}(\phi_i) \mathbf{v}^H(\phi_i)}{\sigma_w^2 + \sigma_i^2} \right] \end{aligned} \quad (2)$$

Therefore, the adaptive weights are given by

$$\mathbf{c}_a \approx \mathbf{R}_{i+n}^{-1} \sigma_i g_i^* \mathbf{v}(\phi_i) \quad (3)$$

Substituting (2) and (3) back into (1), we find the interference-plus-noise output power is

$$\begin{aligned} P_{i+n} &= g_i^2 + \sigma_0^2 + |\sigma_i^2 g_i^*|^2 \mathbf{v}^H(\phi_i) \mathbf{R}_{i+n}^{-1} \mathbf{R}_{i+n} \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_i) \\ &\quad - 2 \cdot \text{Real} \left\{ |\sigma_i g_i^*|^2 \mathbf{v}^H(\phi_i) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_i) \right\} \\ &= g_i^2 + \sigma_0^2 + |\sigma_i^2 g_i^*|^2 \mathbf{v}^H(\phi_i) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_i) \\ &= \sigma_0^2 + \left(1 - \frac{\sigma_i^2}{\sigma_i^2 + \sigma_w^2} \right) g_i^2 \end{aligned} \quad (4)$$

using some algebraic manipulation and the fact that $\mathbf{v}^H(\phi_i) \mathbf{v}(\phi_i) = 1$.

(c) The output power of the signal from the sidelobe canceler is

$$y_s(n) = g_s s(n) - \mathbf{c}_a^H \mathbf{v}(\phi_s) s(n)$$

where

$$\mathbf{c}_a = \mathbf{R}_a^{-1} \mathbf{r}_{ma} \quad (5)$$

The output signal power is then

$$P_s = \sigma_s^2 \cdot |g_s - \mathbf{c}_a^H \mathbf{v}(\phi_s)|^2 \quad (6)$$

Recall the expressions for the correlation matrix \mathbf{R}_a and the cross-correlation vector \mathbf{r}_{ma} from part (a). Using the matrix inversion lemma from Appendix A, we have

$$\mathbf{R}_a^{-1} = \mathbf{R}_{i+n}^{-1} + \frac{\mathbf{R}_{i+n}^{-1} \sigma_s^2 \mathbf{v}(\phi_s) \mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1}}{1 + \sigma_s^2 \mathbf{v}^H(\phi_s) \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)} \quad (7)$$

Also using the matrix inversion lemma on the interference-plus-noise correlation matrix as was done for part (b), we have

$$\mathbf{R}_{i+n}^{-1} = \frac{1}{\sigma_w^2} \left[\mathbf{I} - \frac{\sigma_i^2 \mathbf{v}(\phi_i) \mathbf{v}^H(\phi_i)}{\sigma_w^2 + \sigma_i^2} \right] \quad (8)$$

Substituting (8) into (7), and (7) into (5), and finally (5) into (6), along with some algebraic manipulation, we have the expression for the output signal power

$$\begin{aligned} P_s &= \frac{\sigma_i^2 + \sigma_w^2}{\sigma_i^2 + \sigma_w^2 + \sigma_s^2 \left(1 + \frac{\sigma_i^2}{\sigma_w^2} [1 - \mathbf{v}^H(\phi_i) \mathbf{v}^H(\phi_s)] \right)} \cdot \\ &\quad \left[|g_s|^2 - 2 \frac{\sigma_i^2}{\sigma_i^2 + \sigma_w^2} \text{Real} \{ g_i g_s^* \mathbf{v}^H(\phi_i) \mathbf{v}(\phi_s) \} + \frac{\sigma_i^2}{\sigma_i^2 + \sigma_w^2} |g_i|^2 |\mathbf{v}^H(\phi_i) \mathbf{v}(\phi_s)|^2 \right] \end{aligned} \quad (9)$$

The first term in (8) represents the signal cancellation due to its inclusion in the auxiliary channels. One can see that this term has a maximum value of unit when $\sigma_s^2 = 0$. The second term in the expression for the output signal power represents the signal gain due to interference cancellation of the sidelobe canceler.

- 11.16** (a) From (11.6.7), the LCMV beamformer weight vector is

$$\mathbf{c}_{\text{lcmv}} = \mathbf{R}_{i+n}^{-1} \mathbf{C} (\mathbf{C}^H \mathbf{R}_{i+n}^{-1} \mathbf{C})^{-1} \boldsymbol{\delta}$$

Now for the minimum variance distortionless response (MVDR) beamformer, the constraint matrix is the single look-direction constraint $\mathbf{C} = \mathbf{v}(\phi_s)$ and the constraint response vector is the scalar $\boldsymbol{\delta} = 1$. Substituting these values into the LCMV beamformer weights, we have

$$\begin{aligned}\mathbf{c}_{\text{mvdr}} &= \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s) \left(\mathbf{v}(\phi_s)^H \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s) \right)^{-1} \\ &= \frac{\mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)}{\mathbf{v}(\phi_s)^H \mathbf{R}_{i+n}^{-1} \mathbf{v}(\phi_s)} \\ &= \mathbf{c}_o\end{aligned}$$

- (b) The output interference-plus-noise power for any beamformer \mathbf{c} is simply

$$P_{i+n} = \mathbf{c}^H \mathbf{R}_{i+n} \mathbf{c}$$

Using the LCMV beamformer, we have

$$\begin{aligned}P_{i+n} &= \mathbf{c}_{\text{lcmv}}^H \mathbf{R}_{i+n} \mathbf{c}_{\text{lcmv}} \\ &= \boldsymbol{\delta}^H (\mathbf{C}^H \mathbf{R}_{i+n}^{-1} \mathbf{C})^{-1} \mathbf{C}^H \mathbf{R}_{i+n}^{-1} \mathbf{R}_{i+n} \mathbf{R}_{i+n}^{-1} \mathbf{C} (\mathbf{C}^H \mathbf{R}_{i+n}^{-1} \mathbf{C})^{-1} \boldsymbol{\delta} \\ &= \boldsymbol{\delta}^H (\mathbf{C}^H \mathbf{R}_{i+n}^{-1} \mathbf{C})^{-1} \boldsymbol{\delta}\end{aligned}$$

- 11.17** As with the minimum variance distortionless response (MVDR) beamformer, the linearly constrained minimum variance (LCMV) beamformer can also be implemented using a generalized sidelobe canceler (GSC). Recall the constrained minimization for the LCMV beamformer from (11.6.1)

$$\min \mathbf{c}^H \mathbf{R}_{i+n} \mathbf{c} \quad \text{subject to} \quad \mathbf{C}^H \mathbf{c} = \boldsymbol{\delta}$$

where \mathbf{C} is the $M \times K$ matrix consisting of K constraint vectors and $\boldsymbol{\delta}$ is the $K \times 1$ vector of constraint responses. Recall that for the single constraint MVDR beamformer $[\mathbf{v}(\phi_s) \ \mathbf{B}]$ formed a basis for the M -dimensional space in which the beamforming weight vector lies. \mathbf{B} is the $M \times (M - 1)$ blocking matrix that is orthogonal to the look-direction constraint vector, i.e., $\mathbf{B}^H \mathbf{v}(\phi_s) = \mathbf{0}$. In the case of the multiple constraints in the LCMV beamformer with K constraints, $[\mathbf{C} \ \mathbf{B}]$ must span the M -dimensional space for the LCMV beamformer weight vector and, thus, the blocking matrix \mathbf{B} is a $M \times (M - K)$ matrix. Again, we require that $\mathbf{B}^H \mathbf{C} = \mathbf{0}$. As with the GSC for the MVDR beamformer, the GSC for the LCMV beamformer consists of an upper branch that satisfies the constraint using a quiescent or non-adaptive weight vector \mathbf{c}_q and a lower branch with the blocking matrix \mathbf{B} and the unconstrained adaptive weight vector \mathbf{c}_B . The overall weight vector can then be written as

$$\mathbf{c} = \mathbf{c}_q - \mathbf{B} \mathbf{c}_B \tag{1}$$

where the quiescent weight vector

$$\mathbf{c}_q = \mathbf{C} \tilde{\mathbf{c}}$$

is a linear combination of the K constraints, i.e. it must lie in the constraint subspace. Requiring (1) to satisfy the LCMV constraint $\mathbf{C}^H \mathbf{c} = \boldsymbol{\delta}$, we find that

$$\mathbf{C}^H \mathbf{c}_q = \mathbf{C}^H \mathbf{C} \tilde{\mathbf{c}} = \boldsymbol{\delta}$$

since \mathbf{B} is orthogonal to \mathbf{C} . Solving for $\tilde{\mathbf{c}}$

$$\tilde{\mathbf{c}} = (\mathbf{C}^H \mathbf{C})^{-1} \boldsymbol{\delta} \quad (2)$$

and the quiescent weight vector or the non-adaptive portion in the upper branch of the GSC is

$$\mathbf{c}_q = \mathbf{C}\tilde{\mathbf{c}} = \mathbf{C}(\mathbf{C}^H \mathbf{C})^{-1} \boldsymbol{\delta}$$

Now, looking at the unconstrained minimization that computes an adaptive weight vector \mathbf{c}_B in the lower branch of the GSC that minimizes the overall output power P_{i+n} given by

$$\begin{aligned} P_{i+n} &= (\mathbf{c}_q - \mathbf{B}\mathbf{c}_B)^H \mathbf{R}_{i+n} (\mathbf{c}_q - \mathbf{B}\mathbf{c}_B) \\ &= \mathbf{c}_q^H \mathbf{R}_{i+n} \mathbf{c}_q - \mathbf{c}_q^H \mathbf{R}_{i+n} \mathbf{B} \mathbf{c}_B - \mathbf{c}_B^H \mathbf{B}^H \mathbf{R}_{i+n} \mathbf{c}_q + \mathbf{c}_B^H \mathbf{B}^H \mathbf{R}_{i+n} \mathbf{B} \mathbf{c}_B \end{aligned}$$

If we complete the square by adding and subtracting the term $\mathbf{c}_q^H \mathbf{R}_{i+n} \mathbf{B} (\mathbf{B}^H \mathbf{R}_{i+n} \mathbf{B})^{-1} \mathbf{B}^H \mathbf{R}_{i+n} \mathbf{c}_q$ from P_{i+n} , we have

$$\begin{aligned} P_{i+n} &= \left[\mathbf{c}_B - (\mathbf{B}^H \mathbf{R}_{i+n} \mathbf{B})^{-1} \mathbf{B}^H \mathbf{R}_{i+n} \mathbf{B} \mathbf{c}_q \right]^H \mathbf{B}^H \mathbf{R}_{i+n} \mathbf{B} \left[\mathbf{c}_B - (\mathbf{B}^H \mathbf{R}_{i+n} \mathbf{B})^{-1} \mathbf{B}^H \mathbf{R}_{i+n} \mathbf{B} \mathbf{c}_q \right] \\ &\quad + \mathbf{c}_q^H \mathbf{R}_{i+n} \mathbf{c}_q - \mathbf{c}_q^H \mathbf{R}_{i+n} \mathbf{B} (\mathbf{B}^H \mathbf{R}_{i+n} \mathbf{B})^{-1} \mathbf{B}^H \mathbf{R}_{i+n} \mathbf{c}_q \end{aligned} \quad (3)$$

If we minimize P_{i+n} with respect to \mathbf{c}_B by taking the derivative with respect to \mathbf{c}_B and setting it to zero, we see that the last two terms in (3) drop out since they are independent of \mathbf{c}_B . Thus, the adaptive weight vector in the lower branch of the GSC is

$$\mathbf{c}_B = (\mathbf{B}^H \mathbf{R}_{i+n} \mathbf{B})^{-1} \mathbf{B}^H \mathbf{R}_{i+n} \mathbf{c}_q \quad (4)$$

Note that correlation matrix in the lower branch following the blocking matrix is

$$\mathbf{R}_B = E\{\mathbf{x}_B(n) \mathbf{x}_B^H(n)\} = \mathbf{B}^H \mathbf{R}_{i+n} \mathbf{B}$$

and the cross-correlation vector between the lower branch of the GSC and the upper branch is

$$\begin{aligned} \mathbf{r}_B &= E\{\mathbf{x}_B(n) y_0^*(n)\} \\ &= E\{\mathbf{B}^H \mathbf{x}(n) \mathbf{x}^H(n) \mathbf{c}_q\} \\ &= \mathbf{B}^H \mathbf{R}_{i+n} \mathbf{c}_q \end{aligned}$$

Using these expressions, we can put the adaptive weight vector for the GSC of the LCMV beamformer from (4) in the familiar form from (11.3.42)

$$\mathbf{c}_B = \mathbf{R}_B^{-1} \mathbf{r}_B$$

11.18 This problem is a MATLAB based problem. Below is the MATLAB code and three output figures. The code is broken up into parts (a), (b) and (c) and presented sequentially along with the accompanying figures. Note that the interference scenario is not stressing enough to illustrate the fact that too few degrees of freedom (number of subarrays) will compromise interference nulling capability. See the new problem 11.22 that is not included in your textbook but instead found in this solution manual. This problem, however, is useful for students to learn how to beamform using subarrays.

(a) The signal is clearly visible at $\phi = 0^\circ$ with a SNR = 20 dB. The $M = 20$ element array has sufficient degrees of freedom to cancel the interferer at $\phi = 30^\circ$.

```

%
% Problem 11.18
%
clear

M = 20;                      % number of elements
spacing = 0.5;                % element spacing in wavelengths
N = 1000;                     % number of samples in the realization

% The angle, power, and sample number of the signal of interest
phi_s = 0;                    % angle (deg)
P_s_db = 20;                  % signal power level (dB)
N_s = 100;                    % sample number of signal

% The angle in degrees of the single interference source
phi_i = 30;

% The power levels of the signals and the noise
P_i_db = 50;                  % power of interference source (dB)
sigma_w = 1;                   % standard deviation of noise

% Compute the array response vectors for the signal of interest and
% the interference
v_s = exp(-j*2*pi*spacing*[0:(M-1)']*sin(phi_s*pi/180))/sqrt(M);
v_i = exp(-j*2*pi*spacing*[0:(M-1)']*sin(phi_i*pi/180))/sqrt(M);

% Generate one realization of the signal of interest, the interference,
% and the noise
x_s = zeros(M,N);
x_s(:,N_s) = (10^(P_s_db/20))*v_s;
x_i = (10^(P_i_db/20))*v_i*(randn(1,N) + i*randn(1,N))/sqrt(2);
x_n = (randn(M,N) + i*randn(M,N))/sqrt(2);

% Combine to get overall signal and interference-plus-noise signal
x = x_s + x_i + x_n;
x_in = x_i + x_n;

%%%%%%%%%%%%%
%
% Part A
%
%%%%%%%%%%%%%

% Estimate the covariance matrix of the interference-plus-noise
R_in = (1/N)*x_in*x_in';

% Compute SMI beamformer and normalize to MVDR
c_opt_unnorm = R_in\w_s;
c_opt = c_opt_unnorm/(c_opt_unnorm'*w_s);

```

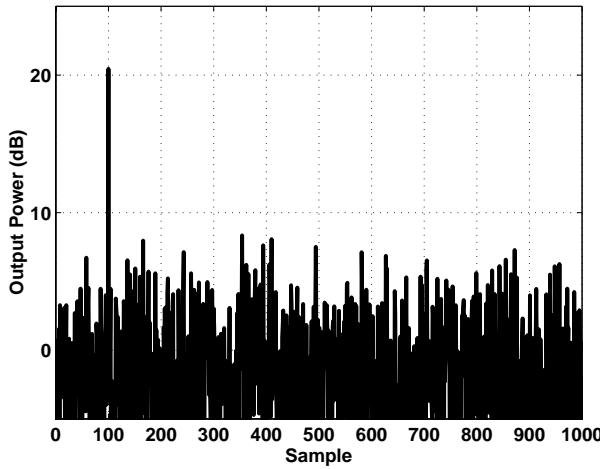


Figure 11.18a: Output signal for the full array SMI adaptive beamformer with $M = 20$ elements.

```
% Compute the output signal
y_out = c_opt'*x;

% Plot out the output signal in decibels
figure
plot(1:N,20*log10(abs(y_out)), 'b', 'linewidth', 2.5);
xlabel('Sample', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Output Power (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca, 'fontweight', 'bold', 'fontsize', 18);
axis([0 1000 -5 25]);
set(gca, 'xtick', 0:100:1000);
set(gca, 'ytick', 0:10:40);
grid
```

(b) As in part (a), the signal is clearly visible at $\phi = 0^\circ$ with a SNR = 20 dB. The $Q = 4$ subarrays or degrees of freedom are sufficient to cancel the single interferer at $\phi = 30^\circ$.

```
%%%%%%%%%%%%%
%
% Part B
%
%%%%%%%%%%%%%
Q = 4; % number of non-overlapping subarrays
M_sub = 5; % number of elements per subarray

% Compute the subarray transformation matrix
v_sub = exp(-j*2*pi*spacing*[0:(M_sub-1)]'*sin(phi_s*pi/180))/sqrt(M_sub);
T_sub = zeros(M,Q);
for q = 1:Q
    index = (1:M_sub) + (q-1)*M_sub;
    T_sub(index,q) = v_sub;
end
```

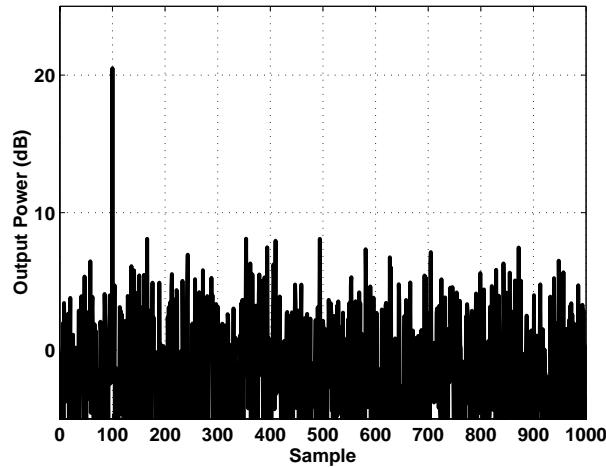


Figure 11.18b: Output signal for the SMI partially adaptive beamformer with $Q = 4$ nonoverlapping subarrays with $\tilde{M} = 5$ elements.

```
% Compute the subarray output signals
x_sub = T_sub'*x; % overall signal
x_in_sub = T_sub'*x_in; % interference-plus-noise

% Compute the subarray correlation matrix
R_in_sub = (1/N)*x_in_sub*x_in_sub';

% Compute subarray SMI beamformer and normalize to MVDR
v_s_sub = T_sub'*v_s;
c_opt_sub_unnorm = R_in_sub\w_s_sub;
c_opt_sub = c_opt_sub_unnorm/(c_opt_sub_unnorm'*v_s_sub);

% Compute the output signal
y_out_sub = c_opt_sub'*x_sub;

% Plot out the output signal in decibels
figure
plot(1:N,20*log10(abs(y_out_sub)),'b','linewidth',2.5);
xlabel('Sample','fontweight','bold','fontsize',20);
ylabel('Output Power (dB)','fontweight','bold','fontsize',20);
set(gca,'fontweight','bold','fontsize',18);
axis([0 1000 -5 25]);
set(gca,'xtick',0:100:1000);
set(gca,'ytick',0:10:40);
grid
```

(c) As in parts (a) and (b), the signal is clearly visible at $\phi = 0^\circ$ with a SNR = 20 dB. The $Q = 2$ subarrays or degrees of freedom are sufficient to cancel the single interferer at $\phi = 30^\circ$. The interference scenario is not stressing enough to illustrate the fact that too few degrees of freedom will limit the ability to cancel interferers. See problem 11.22 in this solutions manual for a more stressing case.

```
%%%%%%%%%%%%%%%
%
% Part C
%
%%%%%%%%%%%%%%%
Q = 2; % number of non-overlapping subarrays
M_sub = 10; % number of elements per subarray

% Compute the subarray transformation matrix
v_sub = exp(-j*2*pi*spacing*[0:(M_sub-1)]'*sin(phi_s*pi/180))/sqrt(M_sub);
T_sub = zeros(M,Q);
for q = 1:Q
    index = (1:M_sub) + (q-1)*M_sub;
    T_sub(index,q) = v_sub;
end

% Compute the subarray output signals
x_sub = T_sub'*x; % overall signal
x_in_sub = T_sub'*x_in; % interference-plus-noise

% Compute the subarray correlation matrix
R_in_sub = (1/N)*x_in_sub*x_in_sub';

% Compute subarray SMI beamformer and normalize to MVDR
v_s_sub = T_sub'*v_s;
c_opt_sub_unnorm = R_in_sub\w_s_sub;
c_opt_sub = c_opt_sub_unnorm/(c_opt_sub_unnorm'*v_s_sub);

% Compute the output signal
y_out_sub = c_opt_sub'*x_sub;

% Plot out the output signal in decibels
figure
plot(1:N,20*log10(abs(y_out_sub)),'b','linewidth',2.5);
xlabel('Sample','fontweight','bold','fontsize',20);
ylabel('Output Power (dB)','fontweight','bold','fontsize',20);
set(gca,'fontweight','bold','fontsize',18);
axis([0 1000 -5 25]);
set(gca,'xtick',0:100:1000);
set(gca,'ytick',0:10:40);
grid
```

- 11.19** This problem is a MATLAB based problem. Below is the MATLAB code and three output figures. The code is broken up into parts (a), (b) and (c) and presented sequentially along with the accompanying figures. Note that the interference scenario is not stressing enough to illustrate the fact that too few degrees of freedom (number of beams) will compromise interference nulling capability. See the new problem 11.23 that is not included in your textbook but instead found in this solution manual. This problem, however, is useful for students to learn how to implement an adaptive beamformer in beamspace which is commonly done in practical applications.

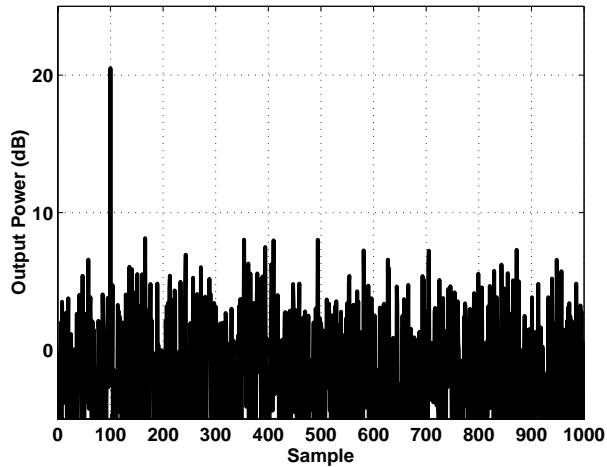


Figure 11.18c: Output signal for the SMI partially adaptive beamformer with $Q = 2$ nonoverlapping subarrays with $\tilde{M} = 10$ elements.

(a) The signal is clearly visible at $\phi = 0^\circ$ with a SNR = 20 dB. The $M = 40$ element array has sufficient degrees of freedom to cancel the interferer at $\phi = 20^\circ$.

```

%
% Problem 11.19
%
clear

M = 40; % number of elements
spacing = 0.5; % element spacing in wavelengths
N = 1000; % number of samples in the realization

% The angle, power, and sample number of the signal of interest
phi_s = 0; % angle (deg)
P_s_db = 20; % signal power level (dB)
N_s = 100; % sample number of signal

% The angle in degrees of the interference source
phi_i = 20;

% The power levels of the signals and the noise
P_i_db = 50; % power of interference source (dB)
sigma_w = 1; % standard deviation of noise

% Compute the array response vectors for the signal of interest and
% the interference
v_s = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi_s*pi/180))/sqrt(M);
v_i = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi_i*pi/180))/sqrt(M);

% Generate one realization of the signal of interest, the interference,
% and the noise
x_s = zeros(M,N);

```

```

x_s(:,N_s) = (10^(P_s_db/20))*v_s;
x_i = (10^(P_i_db/20))*v_i*(randn(1,N) + i*randn(1,N))/sqrt(2);
x_n = (randn(M,N) + i*randn(M,N))/sqrt(2);

% Combine to get overall signal and interference-plus-noise signal
x = x_s + x_i + x_n;
x_in = x_i + x_n;

%%%%%%%%%%%%%%%
%
% Part A
%
%%%%%%%%%%%%%%%

% Estimate the covariance matrix of the interference-plus-noise
R_in = (1/N)*x_in*x_in';

% Compute SMI beamformer and normalize to MVDR
c_opt_unnorm = R_in\v_s;
c_opt = c_opt_unnorm/(c_opt_unnorm'*v_s);

% Compute the output signal
y_out = c_opt'*x;

% Plot out the output signal in decibels
figure
plot(1:N,20*log10(abs(y_out)),'b','linewidth',2.5);
xlabel('Sample','fontweight','bold','fontsize',20);
ylabel('Output Power (dB)','fontweight','bold','fontsize',20);
set(gca,'fontweight','bold','fontsize',18);
axis([0 1000 -5 25]);
set(gca,'xtick',0:100:1000);
set(gca,'ytick',0:10:40);
grid

```

(b) As in part (a), the signal is clearly visible at $\phi = 0^\circ$ with a SNR = 20 dB. The 11 beams or degrees of freedom are sufficient to cancel the single interferer at $\phi = 20^\circ$.

```

%%%%%%%%%%%%%%%
%
% Part B
%
%%%%%%%%%%%%%%%
%
% Angles of beam in beamspace
phi_bs = -5:5;
B = length(phi_bs);

% Compute the beamspace transformation matrix
T_bs = zeros(M,B);

```

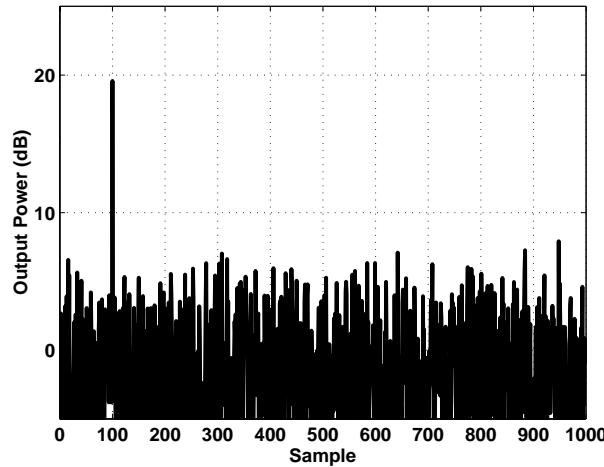


Figure 11.19a: Output signal for the full array SMI adaptive beamformer with $M = 40$ elements.

```

for b = 1:B
    v_bs = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi_bs(b)*pi/180))/sqrt(M);
    T_bs(1:M,b) = v_bs;
end

% Compute the beamspace output signals
x_bs = T_bs'*x; % overall signal
x_in_bs = T_bs'*x_in; % interference-plus-noise

% Compute the subarray correlation matrix
R_in_bs = (1/N)*x_in_bs*x_in_bs';

% Compute subarray SMI beamformer and normalize to MVDR
v_s_bs = T_bs'*v_s;
c_opt_bs_unnorm = R_in_bs\w_s_bs;
c_opt_bs = c_opt_bs_unnorm/(c_opt_bs_unnorm'*v_s_bs);

% Compute the output signal
y_out_bs = c_opt_bs'*x_bs;

% Plot out the output signal in decibels
figure
plot(1:N,20*log10(abs(y_out_bs)), 'b', 'linewidth', 2.5);
xlabel('Sample', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Output Power (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca, 'fontweight', 'bold', 'fontsize', 18);
axis([0 1000 -5 25]);
set(gca, 'xtick', 0:100:1000);
set(gca, 'ytick', 0:10:40);
grid

```

(c) As in parts (a) and (b), the signal is clearly visible at $\phi = 0^\circ$ with a SNR = 20 dB. The 3 beams or degrees of freedom are sufficient to cancel the single interferer at $\phi = 20^\circ$. The interference scenario is not stressing

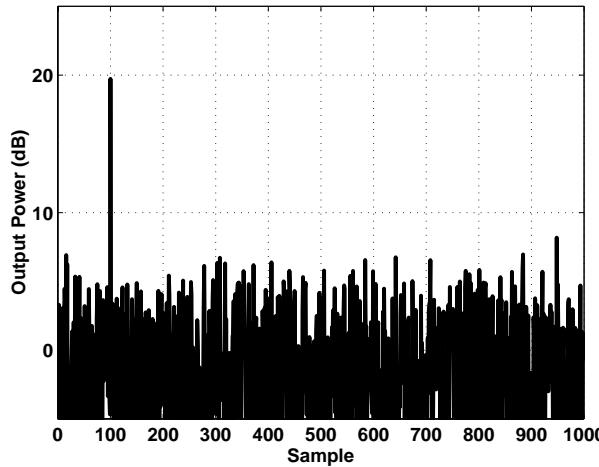


Figure 11.19b: Output signal for the SMI partially adaptive beamformer with 11 beams between $-5^\circ \leq \phi \leq 5^\circ$ at 1° increments.

enough to illustrate the fact that too few degrees of freedom will limit the ability to cancel interferers. See problem 11.23 in this solutions manual for a more stressing case.

```
%%%%%%%
%
% Part C
%
%%%%%%%
clear T_bs

% Angles of beam in beamspace
phi_bs = -1:1;
B = length(phi_bs);

% Compute the beamspace transformation matrix
T_bs = zeros(M,B);
for b = 1:B
    v_bs = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi_bs(b)*pi/180))/sqrt(M);
    T_bs(1:M,b) = v_bs;
end

% Compute the beamspace output signals
x_bs = T_bs'*x; % overall signal
x_in_bs = T_bs'*x_in; % interference-plus-noise

% Compute the subarray correlation matrix
R_in_bs = (1/N)*x_in_bs*x_in_bs';

% Compute subarray SMI beamformer and normalize to MVDR
v_s_bs = T_bs'*v_s;
c_opt_bs_unnorm = R_in_bs\w_s_bs;
c_opt_bs = c_opt_bs_unnorm/(c_opt_bs_unnorm'*v_s_bs);
```

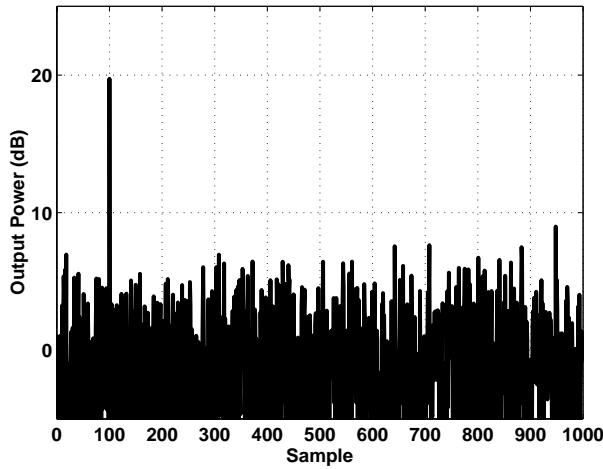


Figure 11.19c: Output signal for the SMI partially adaptive beamformer with 3 beams at $\phi = -1^\circ, 0^\circ, \text{ and } 1^\circ$.

```
% Compute the output signal
y_out_bs = c_opt_bs'*x_bs;

% Plot out the output signal in decibels
figure
plot(1:N,20*log10(abs(y_out_bs)), 'b', 'linewidth', 2.5);
xlabel('Sample', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Output Power (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca, 'fontweight', 'bold', 'fontsize', 18);
axis([0 1000 -5 25]);
set(gca, 'xtick', 0:100:1000);
set(gca, 'ytick', 0:10:40);
grid
```

11.20 Starting with the SINR loss for a beamformer steered to ϕ_s from (11.3.18)

$$L_{\text{sinc}} = \frac{\text{SINR}_{\text{out}}(\phi_s)}{\text{SNR}_0} \quad (1)$$

where $\text{SINR}_{\text{out}}(\phi_s)$ is the output SINR at ϕ_s and SNR_0 is the output signal-to-noise ratio in the absence of interference (noise-only) given by

$$\text{SNR}_0 = M \frac{\sigma_s^2}{\sigma_w^2} \quad (2)$$

The weight vector for the partially adaptive beamformer with a rank-reducing transformation \mathbf{T} in a Q -dimensional subspace is given by

$$\tilde{\mathbf{c}} = \alpha \tilde{\mathbf{R}}_{i+n}^{-1} \tilde{\mathbf{v}}(\phi_s)$$

where α is the normalization constant and

$$\tilde{\mathbf{R}}_{i+n} = \mathbf{T}^H \mathbf{R}_{i+n} \mathbf{T}$$

$$\tilde{\mathbf{v}}(\phi_s) = \mathbf{T}^H \mathbf{v}(\phi_s)$$

are the interference-plus-noise correlation matrix and steering vector projected into the Q -dimensional subspace. If we include the rank-reducing transformation, the resulting $M \times 1$ beamforming weight vector is

$$\mathbf{c}_{\text{pa}} = \mathbf{T}\tilde{\mathbf{c}}$$

Therefore, the output SINR of the beamformer is

$$\begin{aligned} \text{SINR}_{\text{out}} &= \frac{\left| \mathbf{c}_{\text{pa}}^H \sigma_s \mathbf{v}(\phi_s) \right|^2}{\mathbf{c}_{\text{pa}}^H \mathbf{R}_{i+n} \mathbf{c}_{\text{pa}}} \\ &= \frac{M \sigma_s^2}{\tilde{\mathbf{c}}^H \mathbf{T}^H \mathbf{R}_{i+n} \mathbf{T} \tilde{\mathbf{c}}} \\ &= \frac{M \sigma_s^2}{\tilde{\mathbf{c}}^H \tilde{\mathbf{R}}_{i+n} \tilde{\mathbf{c}}} \\ &= \frac{M \sigma_s^2}{\tilde{\mathbf{v}}^H(\phi_s) \tilde{\mathbf{R}}_{i+n}^{-1} \tilde{\mathbf{R}}_{i+n} \tilde{\mathbf{R}}_{i+n}^{-1} \tilde{\mathbf{v}}(\phi_s)} \cdot \left(\tilde{\mathbf{v}}^H(\phi_s) \tilde{\mathbf{R}}_{i+n}^{-1} \tilde{\mathbf{v}}(\phi_s) \right)^2 \\ &= M \sigma_s^2 \tilde{\mathbf{v}}^H(\phi_s) \tilde{\mathbf{R}}_{i+n}^{-1} \tilde{\mathbf{v}}(\phi_s) \end{aligned} \quad (3)$$

Substituting (2) and (3) back into the SINR loss equation in (1), we have the SINR loss for a partially adaptive beamformer

$$\begin{aligned} L_{\text{sinr}} &= \frac{M \sigma_s^2 \tilde{\mathbf{v}}^H(\phi_s) \tilde{\mathbf{R}}_{i+n}^{-1} \tilde{\mathbf{v}}(\phi_s)}{M \sigma_s^2 / \sigma_w^2} \\ &= \sigma_w^2 \tilde{\mathbf{v}}^H(\phi_s) \tilde{\mathbf{R}}_{i+n}^{-1} \tilde{\mathbf{v}}(\phi_s) \end{aligned}$$

11.21 This problem is a MATLAB based problem. Below is the MATLAB code and three output figures. The code is broken up into parts (a) and (c) [no code for part(b)] and presented sequentially along with the accompanying figures.

(a) The SINR loss for the scenario with a single interferer is plotted in Figure 11.21a. Note that the SINR loss is computed by finding the optimum beamformer at each angle $-90 \leq \phi < 90$. The SINR loss has a minimum value of the interference power at the angle of the interference. Otherwise, the SINR loss is near 0 dB since the interference has been effectively canceled at these other angles.

```
%  
% Problem 11.21  
%  
clear  
  
M = 20; % number of elements  
spacing = 0.5; % element spacing in wavelengths  
N = 1000; % number of samples in the realization  
  
% The angle in degrees of the interference source  
phi_i = 30;  
  
% The power levels of the signals and the noise
```

```

P_i_db = 40; % power of interference source (dB)
sigma_i = 10^(P_i_db/20); % std. deviation of interference
sigma_w = 1; % standard deviation of noise

% Compute the array response vector for the interference
u_i = spacing * sin(phi_i*pi/180); % spatial frequency
v_i = exp(-j*2*pi*[0:(M-1)]'*u_i)/sqrt(M); % array response

% Compute the full array interference-plus-noise correlation matrix
R_n = (sigma_w^2) * eye(M); % noise correlation matrix
R_in = (sigma_i^2)*v_i*v_i' + R_n;

%%%%%%%%%%%%%%%
%
% Part A
%
%%%%%%%%%%%%%%%
% Set the angles at which the SINR loss will be computed
phi = -90:0.5:90;

for n = 1:length(phi)
    % steering vector to current angle
    u_s = spacing*sin(phi(n)*pi/180); % spatial frequency
    v = exp(-j*2*pi*[0:(M-1)]'*u_s)/sqrt(M); % steering vector

    L_sinr1(n) = (sigma_w^2) * v'* (R_in\ v); % SINR loss from (11.3.18)
    c_o = R_in\ v; % unnormalized optimum beamformer

    % Compute the array output SINR for the optimum beamformer
    SINR = (abs(c_o'*v)^2)/(c_o'*R_in*c_o);

    % Compute the array output SNR in the absence of interference
    % (best we could do)
    SNR_0 = (abs(v'*v)^2)/(v'*R_n*v);

    % Definition of SINR loss
    L_sinr(n) = SINR/SNR_0;
end

% Plot out the SINR loss of the optimum beamformer in decibels
figure
plot(phi,10*log10(abs(L_sinr)),'b','linewidth',2.5);
xlabel('Angle (deg)', 'fontweight', 'bold', 'fontsize', 20);
ylabel('SINR Loss (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca, 'fontweight', 'bold', 'fontsize', 18);
axis([-90 90 -45 2]);
set(gca, 'xtick', -90:30:90);
set(gca, 'ytick', -40:10:0);

```

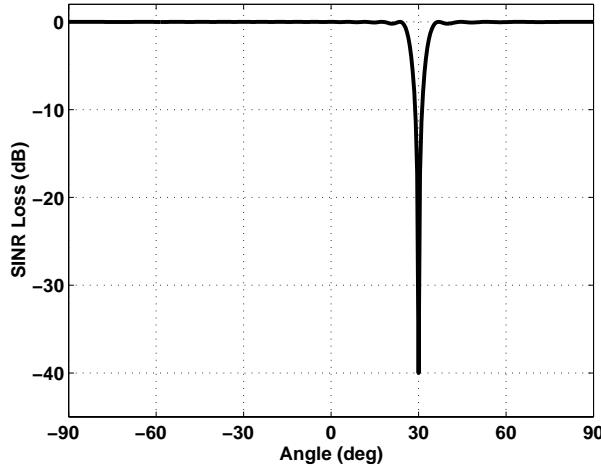


Figure 11.21a: SINR loss for the optimum beamformer.

grid

(b) From (11.3.39), the spatial frequencies of the spatial matched filters that are orthogonal to the spatial matched filter in the upper branch of the generalized sidelobe canceller (GSC) with spatial frequency u_s are

$$u_m = u_s + \frac{m}{M}$$

for $m = 1, 2, \dots, M - 1$. For $\phi_s = 0^\circ$ the spatial frequency of its spatial matched filter is $u_s = 0$. Therefore, the spatial frequencies of the spatial matched filters for the blocking matrix \mathbf{B} for a beamspace GSC are

$$u_m = \frac{m}{M}$$

(c) In this part of the problem, we first compute the SINR loss for the full-rank GSC which is equivalent to the optimum beamformer. This should serve as a check to make sure the the GSC has been implemented correctly. Again, the SINR loss is found by computing a different GSC at each angle ϕ and computing the SINR loss of that beamformer to the given interference scenario. Note that since the lower branch signal is

$$\mathbf{x}_B(n) = \mathbf{B}^H \mathbf{x}(n)$$

and the upper branch signal is

$$y_0(n) = \mathbf{v}^H(\phi_s) \mathbf{x}(n)$$

then the theoretical value of the lower branch correlation matrix is

$$\mathbf{R}_B = E\{\mathbf{x}_B(n) \mathbf{x}_B^H(n)\} = \mathbf{B}^H \mathbf{R}_{i+n} \mathbf{B}$$

and the theoretical value of the cross-correlation vector between the lower branch and the upper branch is given by

$$\mathbf{r}_B = E\{\mathbf{x}_B(n) y_0^*(n)\} = \mathbf{B}^H \mathbf{R}_{i+n} \mathbf{v}(\phi_s)$$

as are used in the accompanying MATLAB code.

We see that the full-rank SINR loss matches the SINR loss of the optimum beamformer from Figure 11.2a for part (a). The SINR loss of the partially adaptive GSC, however, also matches the SINR loss of the optimum beamformer. That is, the same performance is achieved with 2 degrees of freedom as with the full-rank $M = 20$ array since only one interferer is present requiring only one degree of freedom to effectively cancel it.

```
%%%%%%%%%%%%%%%
%
% Part C
%
% - first compute the SINR loss for the full-rank GSC which is
%   equivalent to the optimum beamformer
% - then compute the partially adaptive GSC only using 2 beams in
%   the lower branch
%
%%%%%%%%%%%%%%%
for n = 1:length(phi)
    % steering vector to current angle
    u_s = spacing*sin(phi(n)*pi/180);           % spatial frequency
    v = exp(-j*2*pi*[0:(M-1)]'*u_s)/sqrt(M);  % array response

    % Compute the spatial frequencies for beamspace blocking matrix
    % making sure that they lie between -0.5 and 0.5
    u_bs = u_s + (1:(M-1))/M;
    u_bs(find(u_bs >= 0.5)) = u_bs(find(u_bs >= 0.5)) - 1;
    u_bs(find(u_bs < -0.5)) = u_bs(find(u_bs < -0.5)) + 1;
    u_bs = sort(u_bs);

    % Compute the full-dimensional beamspace blocking matrix
    B = exp(-j*2*pi*[0:(M-1)]'*u_bs)/sqrt(M);

    % Compute the full-rank GSC adaptive weights
    R_B = B'*R_in*B;                           % correlation matrix
    r_B = (sigma_i^2)*(B'*v_i)*(v'*v_i)';    % cross-correlation vector
    c_B = inv(R_B)*r_B;                       % GSC adaptive weights
    c_total = v - B*c_B;                      % effective end-to-end weights

    % Compute the array output SINR for the full-rank GSC
    SINR_full = (abs(c_total'*v)^2)/(c_total'*R_in*c_total);

    % Compute the array output SNR in the absence of interference
    % (best we could do)
    SNR_0 = (abs(v'*v)^2)/(v'*R_n*v);

    % Definition of SINR loss
    L_sinr_full(n) = SINR_full/SNR_0;

    % Find the 2 spatial frequencies closest to the interference
    [dummy,index] = sort(abs(u_bs - u_i));
    u_pa = u_bs(index(1:2));

    % Form the partially adaptive beamspace blocking matrix
    B_pa = exp(-j*2*pi*[0:(M-1)]'*u_pa)/sqrt(M);

```

```

% Compute the partially adaptive GSC adaptive weights
R_B_pa = B_pa'*R_in*B_pa; % correlation matrix
r_B_pa = (sigma_i^2)*(B_pa'*v_i)*(v'*v_i)'; % cross-correlation vector
c_B_pa = inv(R_B_pa)*r_B_pa; % GSC adaptive weights
c_total_pa = v - B_pa*c_B_pa; % effective end-to-end weights

% Compute the array output SINR for the partially adaptive GSC
SINR_pa = (abs(c_total_pa'*v)^2)/(c_total_pa'*R_in*c_total_pa);

% Compute the array output SNR in the absence of interference
% (best we could do)
SNR_0 = (abs(v'*v)^2)/(v'*R_n*v);

% Definition of SINR loss
L_sinr_pa(n) = SINR_pa/SNR_0;
end

% Plot out the SINR loss in decibels for the full-rank GSC
figure
plot(phi,10*log10(abs(L_sinr_full)), 'b', 'linewidth', 2.5);
xlabel('Angle (deg)', 'fontweight', 'bold', 'fontsize', 20);
ylabel('SINR Loss (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca, 'fontweight', 'bold', 'fontsize', 18);
axis([-90 90 -45 2]);
set(gca, 'xtick', -90:30:90);
set(gca, 'ytick', -40:10:0);
grid

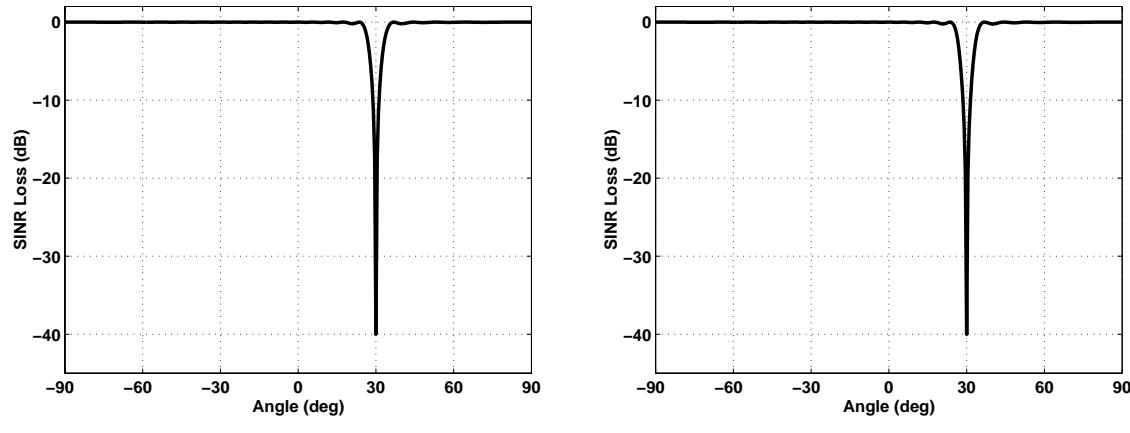
% Plot out the SINR loss in decibels for the partially adaptive GSC
figure
plot(phi,10*log10(abs(L_sinr_pa)), 'b', 'linewidth', 2.5);
xlabel('Angle (deg)', 'fontweight', 'bold', 'fontsize', 20);
ylabel('SINR Loss (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca, 'fontweight', 'bold', 'fontsize', 18);
axis([-90 90 -45 2]);
set(gca, 'xtick', -90:30:90);
set(gca, 'ytick', -40:10:0);
grid

```

11.22 NEW PROBLEM

PROBLEM STATEMENT

Consider the case of a $M = 20$ element array with $d = \lambda/2$ inter-element spacing and thermal noise power $\sigma_w^2 = 1$. Two interference sources are present at $\phi = 3^\circ$ and $\phi = -5^\circ$ both with powers of 50 dB. Generate one realization of 1000 samples of this interferer. In addition a signal of interest is present at $\phi_s = 0^\circ$ with a power of $\sigma_s = 100$ (20 dB) in the $n = 100$ th sample only.



(a) SINR loss for full-rank GSC.

(b) SINR loss for partially adaptive GSC.

Figure 11.21b: SINR loss for full-rank and partially adaptive GSC with 2 spatial matched filters in the lower branch.

- (a) Using an SMI adaptive beamformer for the full array compute the output signal. Is the signal of interest visible?
- (b) Using a partially adaptive beamformer with $Q = 4$ non-overlapping sub-arrays with $\tilde{M} = 5$ elements, compute the output of an SMI adaptive beamformer. What can you say about the signal of interest now?
- (c) Repeat part (b) with $Q = 2$ and $\tilde{M} = 10$. What are your observations now?

SOLUTION

This problem is a MATLAB based problem. Below is the MATLAB code and three output figures. The code is broken up into parts (a), (b) and (c) and presented sequentially along with the accompanying figures.

- (a) The signal is clearly visible at $\phi = 0^\circ$ with a SNR = 20 dB. The $M = 20$ element array has sufficient degrees of freedom to cancel the two interferers at $\phi = 3^\circ$ and $\phi = -5^\circ$.

```
%  
% Problem 11.22 -- new problem that is like 11.18 but with  
% a more challenging interference scenario  
%  
clear  
  
M = 20; % number of elements  
spacing = 0.5; % element spacing in wavelengths  
N = 1000; % number of samples in the realization  
  
% The angle, power, and sample number of the signal of interest  
phi_s = 0; % angle (deg)  
P_s_db = 20; % signal power level (dB)  
N_s = 100; % sample number of signal  
  
% The angle in degrees of the two interference sources  
phi_i1 = 3;
```

```

phi_i2 = -5;

% The power levels of the signals and the noise
P_i1_db = 50; % power of interference source 1 (dB)
P_i2_db = 50; % power of interference source 2 (dB)
sigma_w = 1; % standard deviation of noise

% Compute the array response vectors for the signal of interest and
% the interference
v_s = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi_s*pi/180))/sqrt(M);
v_i1 = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi_i1*pi/180))/sqrt(M);
v_i2 = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi_i2*pi/180))/sqrt(M);

% Generate one realization of the signal of interest, the interference,
% and the noise
x_s = zeros(M,N);
x_s(:,N_s) = (10^(P_s_db/20))*v_s;
x_i1 = (10^(P_s_db/20))*v_i1*(randn(1,N) + i*randn(1,N))/sqrt(2);
x_i2 = (10^(P_s_db/20))*v_i2*(randn(1,N) + i*randn(1,N))/sqrt(2);
x_n = (randn(M,N) + i*randn(M,N))/sqrt(2);

% Combine to get overall signal and interference-plus-noise signal
x = x_s + x_i1 + x_i2 + x_n;
x_in = x_i1 + x_i2 + x_n;

%%%%%%%%%%%%%
%
% Part A
%
%%%%%%%%%%%%%

% Estimate the covariance matrix of the interference-plus-noise
R_in = (1/N)*x_in*x_in';

% Compute SMI beamformer and normalize to MVDR
c_opt_unnorm = R_in\y_s;
c_opt = c_opt_unnorm/(c_opt_unnorm'*v_s);

% Compute the output signal
y_out = c_opt'*x;

% Plot out the output signal in decibels
figure
plot(1:N,20*log10(abs(y_out)), 'b', 'linewidth', 2.5);
xlabel('Sample', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Output Power (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca, 'fontweight', 'bold', 'fontsize', 18);
axis([0 1000 -5 25]);
set(gca, 'xtick', 0:100:1000);

```

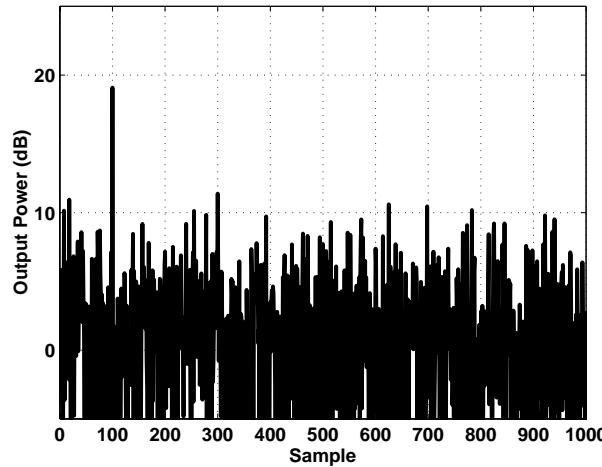


Figure 11.22a: Output signal for the full array SMI adaptive beamformer with $M = 20$ elements.

```
set(gca, 'ytick', 0:10:40);
grid
```

(b) As in part (a), the signal is clearly visible at $\phi = 0^\circ$ with a SNR = 20 dB. The $Q = 4$ subarrays or degrees of freedom are sufficient to cancel the two interferers at $\phi = 3^\circ$ and $\phi = -5^\circ$.

```
%%%%%%%
%
% Part B
%
%%%%%%%
Q = 4; % number of non-overlapping subarrays
M_sub = 5; % number of elements per subarray

% Compute the subarray transformation matrix
v_sub = exp(-j*2*pi*spacing*[0:(M_sub-1)]'*sin(phi_s*pi/180))/sqrt(M_sub);
T_sub = zeros(M,Q);
for q = 1:Q
    index = (1:M_sub) + (q-1)*M_sub;
    T_sub(index,q) = v_sub;
end

% Compute the subarray output signals
x_sub = T_sub'*x; % overall signal
x_in_sub = T_sub'*x_in; % interference-plus-noise

% Compute the subarray correlation matrix
R_in_sub = (1/N)*x_in_sub*x_in_sub';

% Compute subarray SMI beamformer and normalize to MVDR
v_s_sub = T_sub'*v_s;
c_opt_sub_unnorm = R_in_sub\c_opt_sub;
```

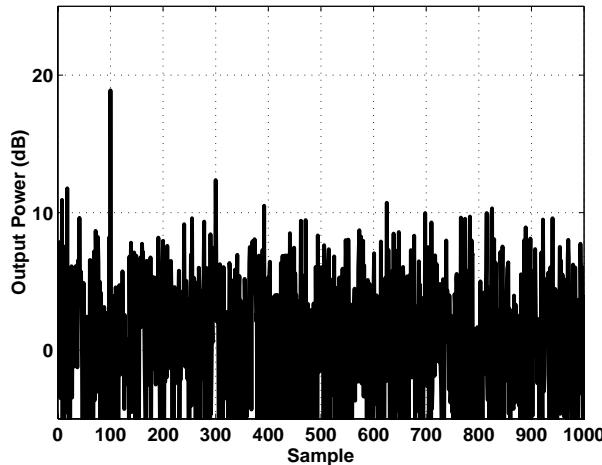


Figure 11.22b: Output signal for the SMI partially adaptive beamformer with $Q = 4$ nonoverlapping subarrays with $\tilde{M} = 5$ elements.

```
c_opt_sub = c_opt_sub_unnorm/(c_opt_sub_unnorm'*v_s_sub);

% Compute the output signal
y_out_sub = c_opt_sub'*x_sub;

% Plot out the output signal in decibels
figure
plot(1:N,20*log10(abs(y_out_sub)), 'b', 'linewidth', 2.5);
xlabel('Sample', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Output Power (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca, 'fontweight', 'bold', 'fontsize', 18);
axis([0 1000 -5 25]);
set(gca, 'xtick', 0:100:1000);
set(gca, 'ytick', 0:10:40);
grid
```

(c) Unlike parts (a) and (b), the signal is no longer visible at $\phi = 0^\circ$. The $Q = 2$ subarrays or degrees of freedom are not enough to cancel the two interferers at $\phi = 3^\circ$ and $\phi = -5^\circ$. Note that one degree of freedom is used by the constraint at $\phi = 0^\circ$ of the optimum beamformer.

```
%%%%%%%%%%%%%
%
% Part C
%
%%%%%%%%%%%%%
Q = 2; % number of non-overlapping subarrays
M_sub = 10; % number of elements per subarray

% Compute the subarray transformation matrix
v_sub = exp(-j*2*pi*spacing*[0:(M_sub-1)]'*sin(phi_s*pi/180))/sqrt(M_sub);
T_sub = zeros(M,Q);
```

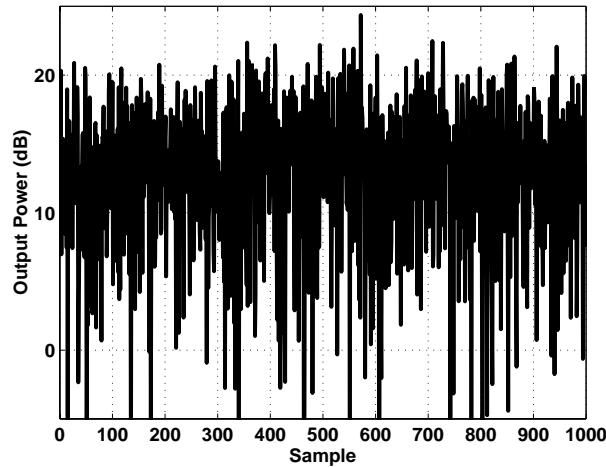


Figure 11.22c: Output signal for the SMI partially adaptive beamformer with $Q = 2$ nonoverlapping subarrays with $\tilde{M} = 10$ elements.

```

for q = 1:Q
    index = (1:M_sub) + (q-1)*M_sub;
    T_sub(index,q) = v_sub;
end

% Compute the subarray output signals
x_sub = T_sub'*x;                      % overall signal
x_in_sub = T_sub'*x_in;                  % interference-plus-noise

% Compute the subarray correlation matrix
R_in_sub = (1/N)*x_in_sub*x_in_sub';

% Compute subarray SMI beamformer and normalize to MVDR
v_s_sub = T_sub'*v_s;
c_opt_sub_unnorm = R_in_sub\ v_s_sub;
c_opt_sub = c_opt_sub_unnorm/(c_opt_sub_unnorm'*v_s_sub);

% Compute the output signal
y_out_sub = c_opt_sub'*x_sub;

% Plot out the output signal in decibels
figure
plot(1:N,20*log10(abs(y_out_sub)),'b','linewidth',2.5);
xlabel('Sample','fontweight','bold','fontsize',20);
ylabel('Output Power (dB)','fontweight','bold','fontsize',20);
set(gca,'fontweight','bold','fontsize',18);
axis([0 1000 -5 25]);
set(gca,'xtick',0:100:1000);
set(gca,'ytick',0:10:40);
grid

```

11.23 NEW PROBLEM

PROBLEM STATEMENT

Consider the case of a $M = 40$ element array with $d = \lambda/2$ inter-element spacing and thermal noise power $\sigma_w^2 = 1$. Three interference sources are present at $\phi = -5^\circ, -10^\circ$ and 5° with powers of 50, 40 and 45 dB, respectively. Generate one realization of 1000 samples of this interferer. In addition a signal of interest is present at $\phi_s = 0^\circ$ with a power of $\sigma_s = 100$ (20 dB) in the $n = 100$ th sample only.

- (a) Using an SMI adaptive beamformer for the full array compute the output signal. Is the signal of interest visible?
- (b) Using a beamspace partially adaptive beamformer 11 beams at the angles $-5^\circ \leq \phi \leq 5^\circ$ at 1° increments, compute the output of an SMI adaptive beamformer. What can you say about the signal of interest now?
- (c) Repeat part (b) with beams only at $\phi = -1^\circ, 0^\circ, 1^\circ$. What are your observations now?

SOLUTION

This problem is a MATLAB based problem. Below is the MATLAB code and three output figures. The code is broken up into parts (a), (b) and (c) and presented sequentially along with the accompanying figures.

- (a) The signal is clearly visible at $\phi = 0^\circ$ with a SNR = 20 dB. The $M = 40$ element array has sufficient degrees of freedom to cancel the three interferers at $\phi = -5^\circ, -10^\circ$, and 5° .

```
%  
% Problem 11.23 -- new problem that is like 11.19 but with  
%                   a more challenging interference scenario  
%  
clear  
  
M = 40;                      % number of elements  
spacing = 0.5;                % element spacing in wavelengths  
N = 1000;                     % number of samples in the realization  
  
% The angle, power, and sample number of the signal of interest  
phi_s = 0;                     % angle (deg)  
P_s_db = 20;                   % signal power level (dB)  
N_s = 100;                     % sample number of signal  
  
% The angle in degrees of the interference sources  
phi_i1 = -5;  
phi_i2 = -10;  
phi_i3 = 5;  
  
% The power levels of the signals and the noise  
P_i1_db = 50;                  % power of interference source 1 (dB)  
P_i2_db = 40;                  % power of interference source 2 (dB)  
P_i3_db = 45;                  % power of interference source 3 (dB)  
sigma_w = 1;                    % standard deviation of noise
```

```

% Compute the array response vectors for the signal of interest and
% the interference
v_s = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi_s*pi/180))/sqrt(M);
v_i1 = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi_i1*pi/180))/sqrt(M);
v_i2 = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi_i2*pi/180))/sqrt(M);
v_i3 = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi_i3*pi/180))/sqrt(M);

% Generate one realization of the signal of interest, the interference,
% and the noise
x_s = zeros(M,N);
x_s(:,N_s) = (10^(P_s_db/20))*v_s;
x_i1 = (10^(P_i1_db/20))*v_i1*(randn(1,N) + j*randn(1,N))/sqrt(2);
x_i2 = (10^(P_i2_db/20))*v_i2*(randn(1,N) + j*randn(1,N))/sqrt(2);
x_i3 = (10^(P_i3_db/20))*v_i3*(randn(1,N) + j*randn(1,N))/sqrt(2);
x_n = (randn(M,N) + j*randn(M,N))/sqrt(2);

% Combine to get overall signal and interference-plus-noise signal
x = x_s + x_i1 + x_i2 + x_i3 + x_n;
x_in = x_i1 + x_i2 + x_i3 + x_n;

%%%%%%%%%%%%%
%
% Part A
%
%%%%%%%%%%%%%

% Estimate the covariance matrix of the interference-plus-noise
R_in = (1/N)*x_in*x_in';

% Compute SMI beamformer and normalize to MVDR
c_opt_unnorm = R_in\y_s;
c_opt = c_opt_unnorm/(c_opt_unnorm'*v_s);

% Compute the output signal
y_out = c_opt'*x;

% Plot out the output signal in decibels
figure
plot(1:N,20*log10(abs(y_out)), 'b', 'linewidth', 2.5);
xlabel('Sample', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Output Power (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca, 'fontweight', 'bold', 'fontsize', 18);
axis([0 1000 -5 25]);
set(gca, 'xtick', 0:100:1000);
set(gca, 'ytick', 0:10:40);
grid

```

(b) As in part (a), the signal is clearly visible at $\phi = 0^\circ$ with a SNR = 20 dB. The 11 beams or degrees of

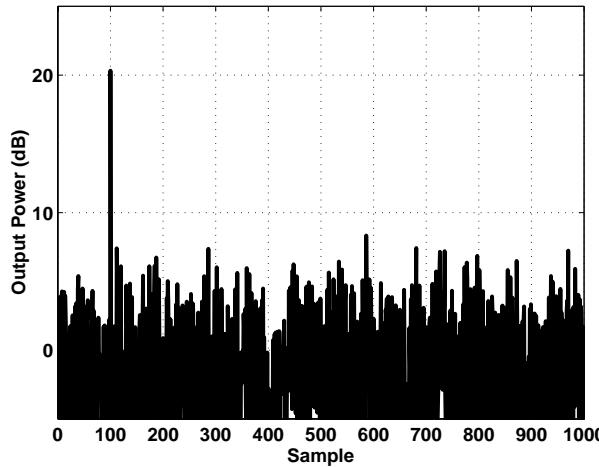


Figure 11.23a: Output signal for the full array SMI adaptive beamformer with $M = 40$ elements.

freedom are sufficient to cancel the three interferers at $\phi = -5^\circ$, -10° , and 5° .

```
%%%%%%%
%
% Part B
%
%%%%%%%%%%%%%
%
% Angles of beam in beamspace
phi_bs = -5:5;
B = length(phi_bs);

% Compute the beamspace transformation matrix
T_bs = zeros(M,B);
for b = 1:B
    v_bs = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi_bs(b)*pi/180))/sqrt(M);
    T_bs(1:M,b) = v_bs;
end

% Compute the beamspace output signals
x_bs = T_bs'*x; % overall signal
x_in_bs = T_bs'*x_in; % interference-plus-noise

% Compute the subarray correlation matrix
R_in_bs = (1/N)*x_in_bs*x_in_bs';

% Compute subarray SMI beamformer and normalize to MVDR
v_s_bs = T_bs'*v_s;
c_opt_bs_unnorm = R_in_bs\w_s_bs;
c_opt_bs = c_opt_bs_unnorm/(c_opt_bs_unnorm'*v_s_bs);

% Compute the output signal
y_out_bs = c_opt_bs'*x_bs;
```

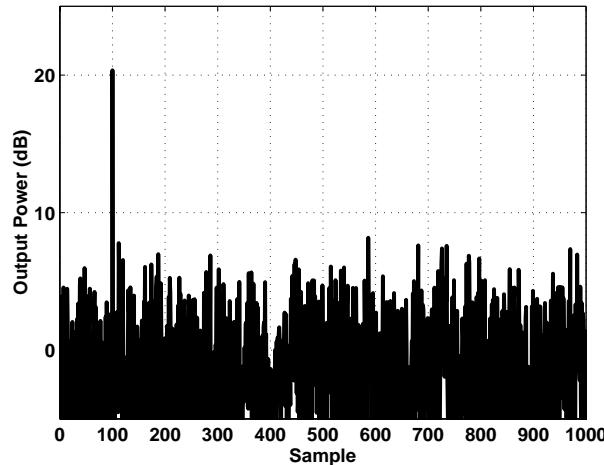


Figure 11.23b: Output signal for the SMI partially adaptive beamformer with 11 beams between $-5^\circ \leq \phi \leq 5^\circ$ at 1° increments.

```
% Plot out the output signal in decibels
figure
plot(1:N,20*log10(abs(y_out_bs)), 'b', 'linewidth', 2.5);
xlabel('Sample', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Output Power (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca, 'fontweight', 'bold', 'fontsize', 18);
axis([0 1000 -5 25]);
set(gca, 'xtick', 0:100:1000);
set(gca, 'ytick', 0:10:40);
grid
```

(c) Unlike parts (a) and (b), the signal is no longer visible at $\phi = 0^\circ$. The 3 beams or degrees of freedom are not enough to cancel the three interferers at $\phi = -5^\circ$, -10° , and 5° . Note that one degree of freedom is used by the constraint at $\phi = 0^\circ$ of the optimum beamformer.

```
%%%%%%%%%%%%%
%
% Part C
%
%%%%%%%%%%%%%
clear T_bs

% Angles of beam in beampspace
phi_bs = -1:1;
B = length(phi_bs);

% Compute the beampspace transformation matrix
T_bs = zeros(M,B);
for b = 1:B
    v_bs = exp(-j*2*pi*spacing*[0:(M-1)]'*sin(phi_bs(b)*pi/180))/sqrt(M);
    T_bs(1:M,b) = v_bs;
end
```

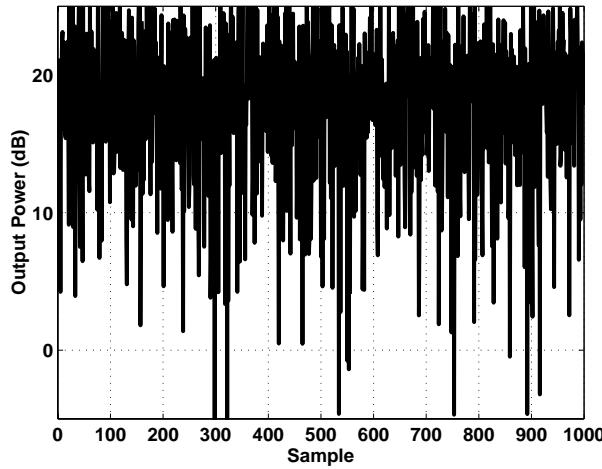


Figure 11.23c: Output signal for the SMI partially adaptive beamformer with 3 beams at $\phi = -1^\circ, 0^\circ, \text{ and } 1^\circ$.

```
% Compute the beamspace output signals
x_bs = T_bs'*x; % overall signal
x_in_bs = T_bs'*x_in; % interference-plus-noise

% Compute the subarray correlation matrix
R_in_bs = (1/N)*x_in_bs*x_in_bs';

% Compute subarray SMI beamformer and normalize to MVDR
v_s_bs = T_bs'*v_s;
c_opt_bs_unnorm = R_in_bs\w_s_bs;
c_opt_bs = c_opt_bs_unnorm/(c_opt_bs_unnorm'*v_s_bs);

% Compute the output signal
y_out_bs = c_opt_bs'*x_bs;

% Plot out the output signal in decibels
figure
plot(1:N,20*log10(abs(y_out_bs)), 'b', 'linewidth', 2.5);
xlabel('Sample', 'fontweight', 'bold', 'fontsize', 20);
ylabel('Output Power (dB)', 'fontweight', 'bold', 'fontsize', 20);
set(gca, 'fontweight', 'bold', 'fontsize', 18);
axis([0 1000 -5 25]);
set(gca, 'xtick', 0:100:1000);
set(gca, 'ytick', 0:10:40);
grid
```

Chapter 12

Further Topics

12.1 Prove (12.1.27), which relates the output and input fourth-order cumulants of a linear, time-invariant system.

We have

$$\begin{aligned}
 r_y^{(4)}(l_1, l_2, l_3) &= E \left\{ y^*(n)y^*(n+l_1)y(n+l_2) + y(n+l_3) \right\} \\
 &= E \left\{ \sum_{k_0} h^*(k_0)x^*(n-k_0) \sum_{k_1} h^*(k_1)x^*(n-k_1) \sum_{k_2} h(k_2)x(n-k_2) \sum_{k_3} h(k_3)x(n-k_3) \right\} \\
 &= \sum_{k_0} \sum_{k_1} \sum_{k_2} \sum_{k_3} h^*(k_0)h^*(k_1)h(k_2)h(k_3) E \left\{ x^*(n-k_0)x^*(n-k_1)x(n-k_2)x(n-k_3) \right\} \\
 &= \sum_{k_0} \sum_{k_1} \sum_{k_2} \sum_{k_3} h^*(k_0)h^*(k_1)h(k_2)h(k_3) r_x^{(4)}(l_1 - k_1 + k_0, l_2 - k_2 + k_0, l_3 - k_3 + k_0)
 \end{aligned}$$

using the linearity property of mathematical expectation and the definitions of higher order moments.

12.2 Derive (12.1.35) and (12.1.36), using the formulas for the cumulant of the sum of IID random variables, developed in Section 3.2.4.

From (12.1.28) and (12.1.34), we have

$$\kappa_x^{(4)}(l_1, l_2, l_3) = \gamma_w^{(4)} \sum_{n=0}^{\infty} h(n)h(n+l_1)h(n+l_2)h(n+l_3)$$

or for $l_1 = l_2 = l_3 = 0$

$$\kappa_x^{(4)} = \gamma_w^{(4)} \sum_{n=0}^{\infty} h^4(n)$$

which also follows from (3.2.75).

12.3 If $x(n)$ is a stationary Gaussian process, show that $E\{x(n)x^2(n-l)\} = \rho_x^2(l)$ and explain how it can be used to investigate the presence of nonlinearities.

From (3.2.53) we obtain

$$\begin{aligned}
 E\{x(n)x(n)x(n+l)x(n+l)\} &= E\{x^2(n)\} E\{x^2(n+l)\} \\
 &\quad + E\{x(n)x(n+l)\}^2 + E\{x(n)x(n+l)\}^2 \\
 &= r_x^2(0) + 2r_x^2(l)
 \end{aligned}$$

We can check for non-Gaussianity or nonlinearities by exciting a system with a Gaussian input $w(n)$, computing the output $x(n)$, and then computing and comparing the autocorrelation of $x(n)$ with the autocorrelation of $x^2(n)$.

12.4 In this problem we use an MA(2) model to explore some properties of cumulants and bispectra.

- (a) Write a MATLAB function `k=cuma(b)` that computes the cumulant $\kappa_x^{(3)}(l_1, l_2)$ of the MA(2) model $H(z) = b_0 + b_1z^{-1} + b_2z^{-2}$ for $-L \leq l_1, l_2 \leq L$.

- (b) Use the functions $k=cuma(b)$, $X=fft(x)$, and $X=shiftfft(X)$ to compute the bispectra of the three MA(2) models in Table 12.1. Plot your results and compare with those in Figure 12.2.
- (c) Compute the bispectra of the models using the formula

$$R_x^{(3)}(e^{j\omega_1}, e^{j\omega_2}) = \kappa_w^{(3)} H(e^{j\omega_1}) H(e^{j\omega_2}) H^*(e^{j(\omega_1+\omega_2)})$$

for $\omega_1 = \omega_2 = 2\pi k/N$, $0 \leq k \leq N - 1$. Compare with the results in part b and Figure 12.2.

- (d) Show that the bispectrum can be computed in MATLAB using the following segment of code:

```
H=freqz(h,1,N,'whole');
Hc=conj(H);
R3x=(H.*H').*hankel(Hc,Hc([N,1:N-1]));
R3x=shiftfft(R3x);
```

The Matlab script is given below and plots are shown in Figures 12.4a through 12.4d.

```
% Problem 12.04
```

```
clear
close all
j=sqrt(-1);
z1=0.8*exp(j*0.6*pi);
z2=0.8*exp(j*0.9*pi);
hmin=poly([z1 conj(z1) z2 conj(z2)]);
hmax=poly([1/z1 1/conj(z1) 1/z2 1/conj(z2)]);

a=0.5;b=0.9;
hmin=[1 -(a+b) a*b];
hmax=fliplr(hmin); %hmax=hmax/hmax(1);

[Hmin,om]=freqz(hmin,1,1000);
[Hmax,om]=freqz(hmax,1,1000);
f=om/(2*pi);

subplot(2,2,1),plot(f,abs(Hmin));axis([0 0.5 -Inf Inf]);
ylabel('|H(e^{j\omega})|');
xlabel('\omega/2\pi');
title('Minimum phase system')
subplot(2,2,3),plot(f,angle(Hmin));axis([0 0.5 -Inf Inf]);
ylabel('\angle H(e^{j\omega})');
xlabel('\omega/2\pi');
title('Minimum phase system')

subplot(2,2,2),plot(f,abs(Hmax));axis([0 0.5 -Inf Inf]);
ylabel('|H(e^{j\omega})|');
xlabel('\omega/2\pi');
title('Maximum phase system')

subplot(2,2,4),plot(f,angle(Hmax));axis([0 0.5 -Inf Inf]);
```

```

ylabel('\angle H(e^{j\omega}))';
xlabel('\omega/2\pi');
title('Maximum phase system')

NFFT=128;
H=freqz(hmin,1,NFFT,'whole');
Hc=conj(H);
for i=1:NFFT
    for k=1:NFFT
        ik=mod(i+k,NFFT);
        if ik==0 ik=1; end
        R3min(i,k)=H(i)*H(k)*Hc(ik);
    end
end
R3min=fftshift(R3min);

H=freqz(hmax,1,NFFT,'whole');
Hc=conj(H);
for i=1:NFFT
    for k=1:NFFT
        ik=mod(i+k,NFFT);
        if ik==0 ik=1; end
        R3max(i,k)=H(i)*H(k)*Hc(ik);
    end
end
R3max=fftshift(R3max);

exportfig(gcf,'p1204a.eps','fontmode','scaled',...
    'linemode','scaled','color','cmyk');

figure
om = [-NFFT/2:NFFT/2-1]/NFFT;

subplot(2,2,1),contour(om,om,abs(R3min))
title('Bispectrum magnitude: Min phase')
xlabel('f_1');
ylabel('f_2');

subplot(2,2,3),contour(om,om,angle(R3min))
title('Bispectrum phase: Min phase')
xlabel('f_1');
ylabel('f_2');

subplot(2,2,2),contour(om,om,abs(R3max))
title('Bispectrum magnitude: Max phase')
xlabel('f_1');
ylabel('f_2');

subplot(2,2,4),contour(om,om,angle(R3max))

```

```

title('Bispectrum phase: Max phase')
xlabel('f_1');
ylabel('f_2');

exportfig(gcf,'p1204b.eps','fontmode','scaled',...
'linemode','scaled','color','cmyk');

figure

[R3min,om]=bispectrum(hmin,NFFT);
[R3max,om]=bispectrum(hmax,NFFT);

exportfig(gcf,'p1204c.eps','fontmode','scaled',...
'linemode','scaled','color','cmyk');

figure
subplot(2,2,1),contour(om,om,abs(R3min))
title('Bispectrum magnitude: Min phase')
xlabel('f_1');
ylabel('f_2');

subplot(2,2,3),contour(om,om,angle(R3min))
title('Bispectrum phase: Min phase')
xlabel('f_1');
ylabel('f_2');

subplot(2,2,2),contour(om,om,abs(R3max))
title('Bispectrum magnitude: Max phase')
xlabel('f_1');
ylabel('f_2');

subplot(2,2,4),contour(om,om,angle(R3max))
title('Bispectrum phase: Max phase')
xlabel('f_1');
ylabel('f_2');

exportfig(gcf,'p1204d.eps','fontmode','scaled',...
'linemode','scaled','color','cmyk');

```

- 12.5** Using the minimum-, maximum-, and mixed-phase systems discussed in Example 12.1.1, write a MATLAB program to reproduce the results shown in Figures 12.3 and 12.4. Use $a = 0.4$, $b = 0.8$, and $N = 300$ samples.

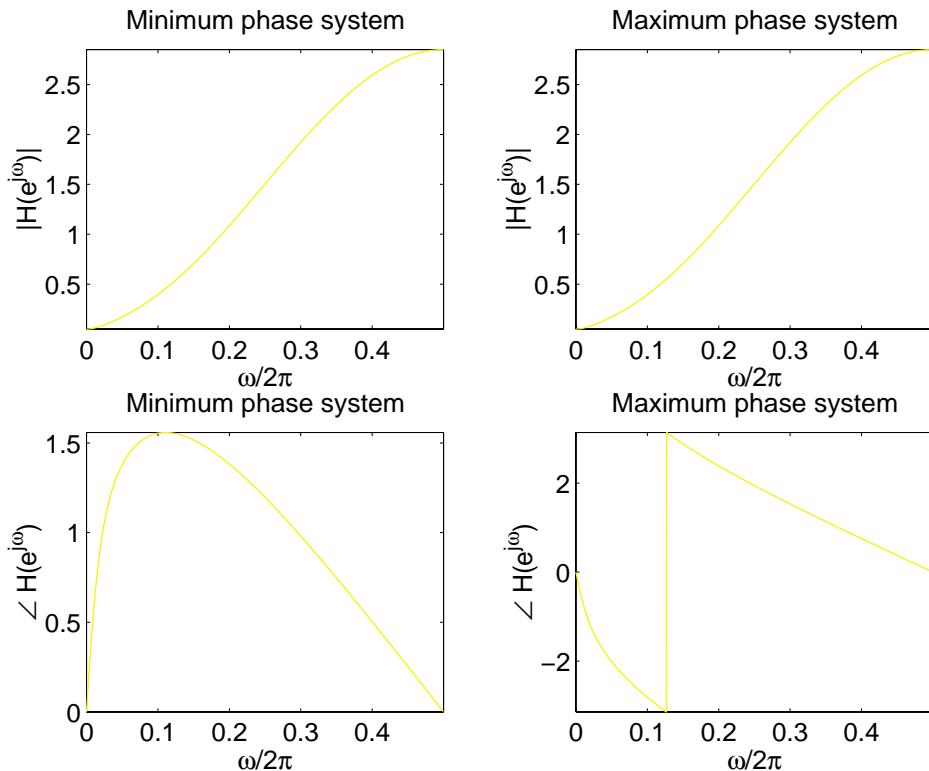


Figure 12.4a: Plots of system responses in 12.4a

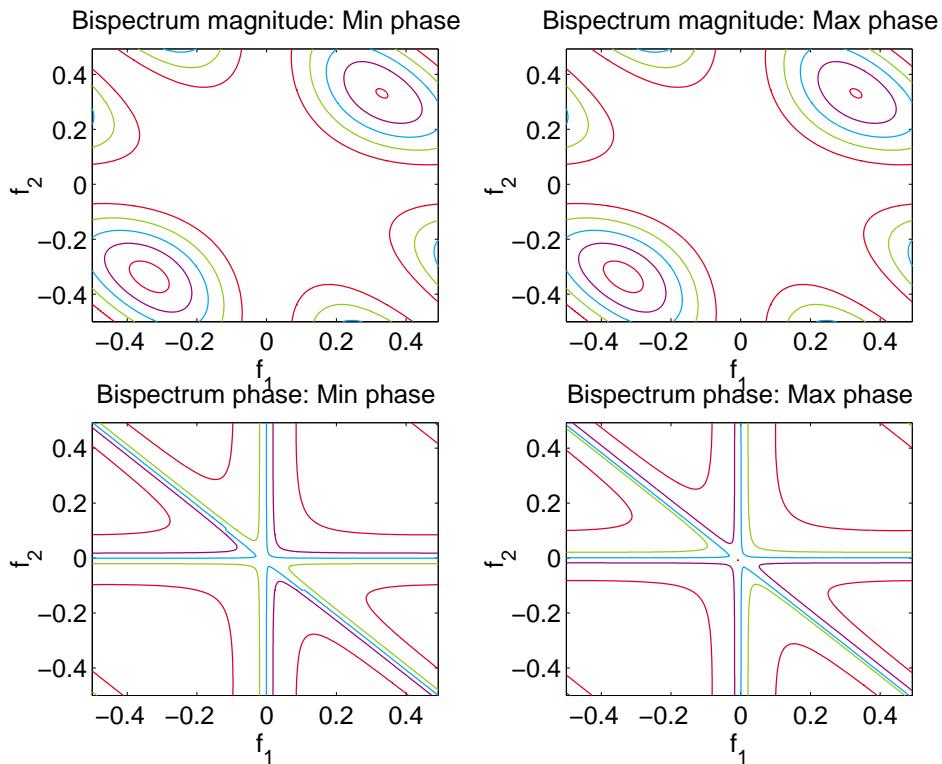
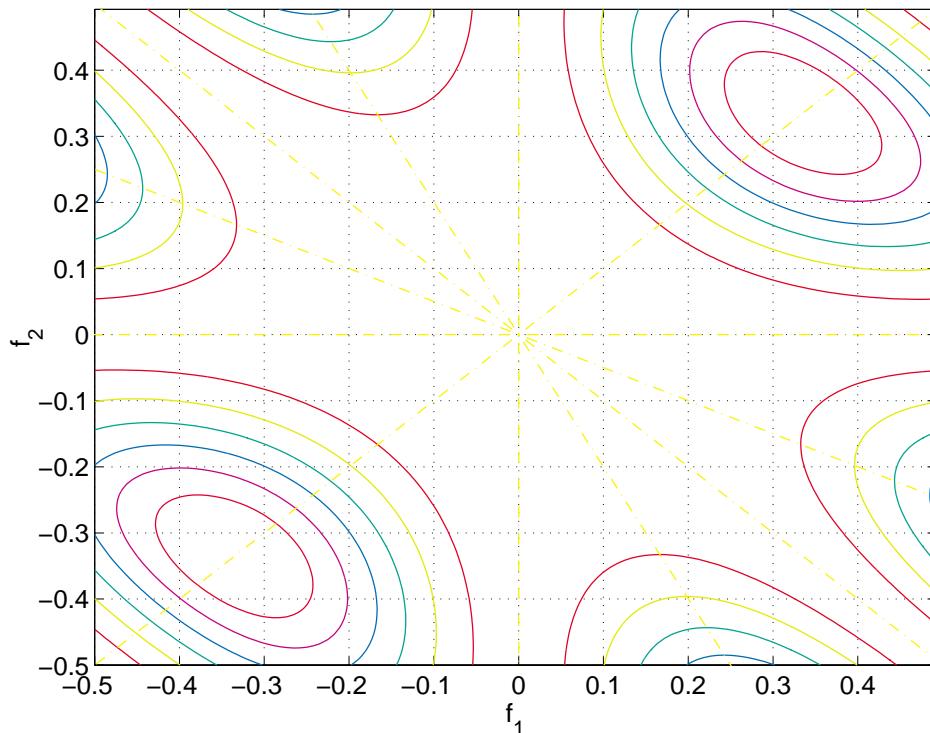
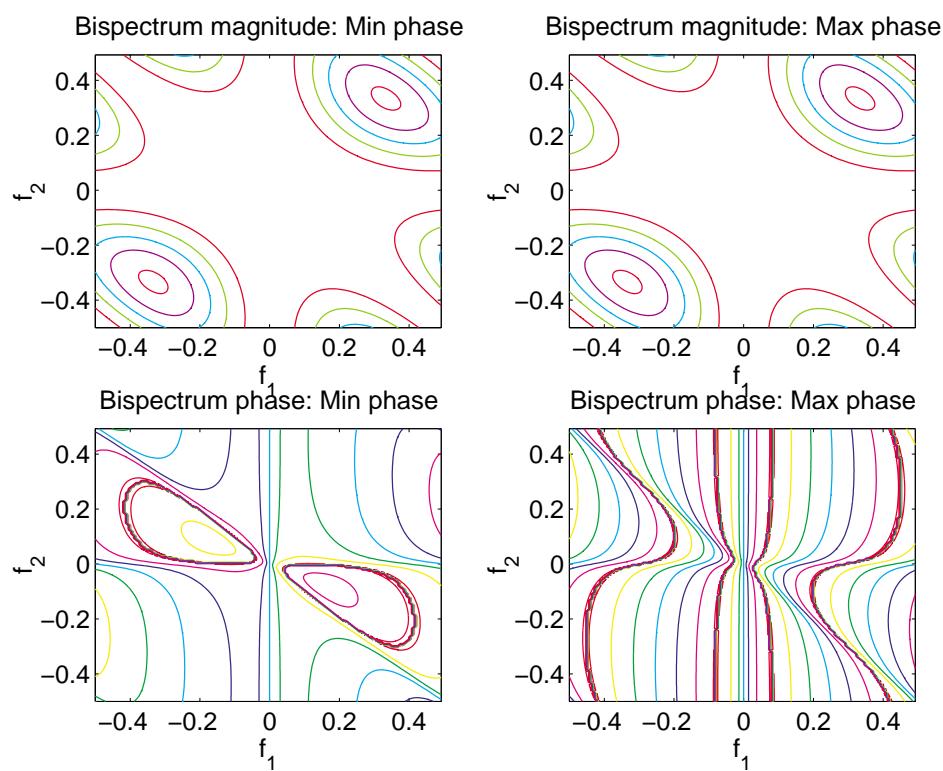


Figure 12.4b: Plots of bispectra in 12.4b

**Figure 12.4c:** Plot of bispectrum in 12.4c**Figure 12.4d:** Plots of bispectra in 12.4d

The Matlab script is given below and plots are shown in Figures 12.5a and 12.5b.

```
% Min, max, and mix phase deconvolution of Gaussian
% and non-Gaussian random signals.

close all
a=0.4; b=0.8;
N=300;

hmin=[1 -(a+b) a*b]';
hmax=flipud(hmin);
hmix=[-a (1+a*b) -b]';

w=randn(N,1);title('Gaussian');
x=filter(1,hmin,w);
wmin=filter(hmin,1,x);
wmax=filter(hmax,1,x);
wmix=filter(hmix,1,x);
subplot(5,1,1),plot(w); ylabel('w(n)');title('Gaussian');
subplot(5,1,2),plot(x); ylabel('x(n)');
subplot(5,1,3),plot(wmin); ylabel('w_{min}(n)');
subplot(5,1,4),plot(wmax); ylabel('w_{max}(n)');
subplot(5,1,5),plot(wmix); ylabel('w_{mix}(n)');
xlabel('Sample index (n)');

exportfig(gcf,'p1205a.eps','fontmode','scaled',...
    'linemode','scaled','color','cmyk');

figure
w=-log(rand(N,1));
x=filter(1,hmin,w);
wmin=filter(hmin,1,x);
wmax=filter(hmax,1,x);
wmix=filter(hmix,1,x);
subplot(5,1,1),plot(w); ylabel('w(n)');title('Non-Gaussian');
subplot(5,1,2),plot(x); ylabel('x(n)');
subplot(5,1,3),plot(wmin); ylabel('w_{min}(n)');
subplot(5,1,4),plot(wmax); ylabel('w_{max}(n)');
subplot(5,1,5),plot(wmix); ylabel('w_{mix}(n)');
xlabel('Sample index (n)');
exportfig(gcf,'p1205b.eps','fontmode','scaled',...
    'linemode','scaled','color','cmyk');
```

- 12.6** Use the Levinson-Durbin algorithm, developed in Chapter 7, to derive expressions (12.5.20), direct-form coefficients, and (12.5.21) for the lattice parameters of the fractional pole model.

We first determine the partial correlation coefficients for $m = 1$ and $m = 2$. Then we go from order $m - 1$ to order m using induction.

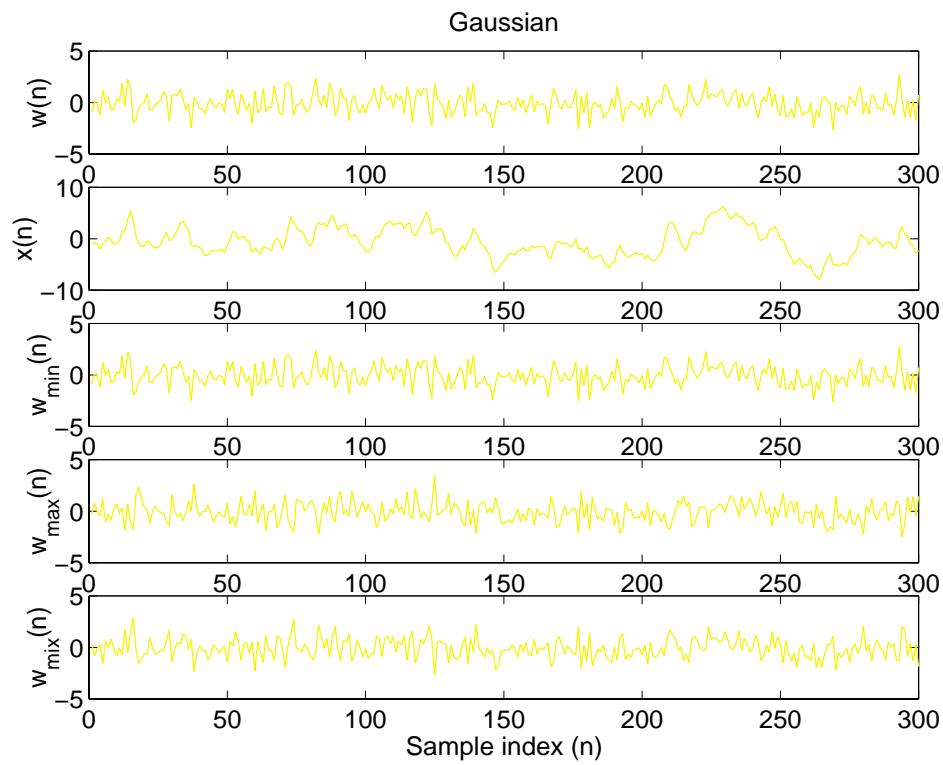


Figure 12.5a: Signal plots in 12.5a

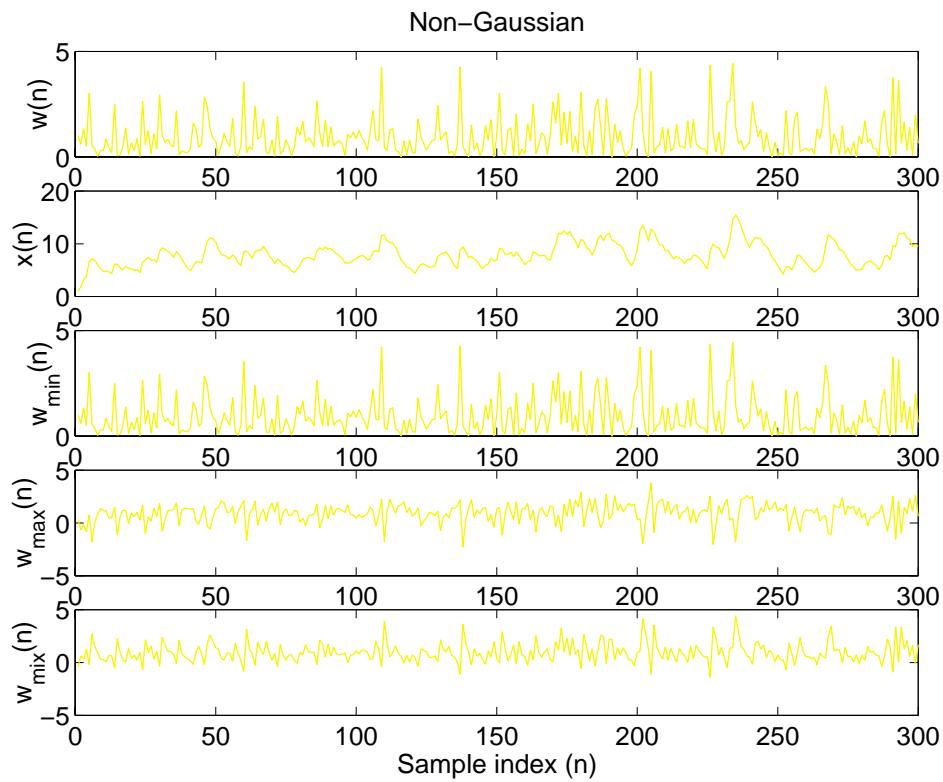


Figure 12.5b: Signal plots in 12.5b

For $m = 1$ we have

$$k_1 = a_1^{(1)} = -\rho(1) = -\frac{d}{1-d}$$

For $m = 2$ we use the Levinson-Durbin recursion. Indeed, we have

$$k_2 = a_2^{(2)} = -\frac{\rho(2) + a_1^{(1)}\rho(1)}{1 + a_1^{(1)}\rho(1)} = -\frac{d}{2-d}$$

and

$$a_1^{(2)} = a_1^{(1)} + k_2 a_1^{(1)} = \frac{-2d}{2-d}$$

which is identical to (12.5.20) for $m = 2$ and $k = 1$.

Let us assume now that formula (12.5.20) is true for $m - 1$. Using the Levinson-Durbin recursion, we have

$$a_j^{(m)} = a_j^{(m-1)} + k_m a_{m-j}^{(m-1)}$$

$$\begin{aligned} a_j^{(m-1)} &= \binom{m-1}{j} \frac{(j-d-1)!(m-1-d-j)!}{(-d-1)!(m-1-d)!} \\ &= \binom{m}{j} \frac{(j-d-1)!(m-d-j)!(m-j)(m-d)}{(-d-1)!(m-d)!m(m-d-j)} \end{aligned}$$

But

$$a_j^{(m)} a_{m-j}^{(m-1)} = \binom{m}{j} \frac{(j-d-1)!(m-d-j)!(-d)j}{(-d-1)!(m-d)!m(m-d-j)}$$

Using the previous relations we can show that

$$a_j^{(m-1)} + k_m a_{m-j}^{(m-1)} = \binom{m}{j} \frac{(j-d-1)!(m-d-j)!}{(-d-1)!(m-d)!} = a_j^{(m)}$$

which completes the proof by induction.

12.7 Consider the FPZ($1, d, 0$) model

$$H_{\text{fpz}}(z) = \frac{1}{(1-z^{-1})^d} \frac{1}{(1+az^{-1})}$$

where $-\frac{1}{2} < d < \frac{1}{2}$ and $-1 < a < 1$. Compute and plot the impulse response, autocorrelation, and spectrum for $a = \pm 0.9$ and $d = \pm 0.2, \pm 0.4$. Identify which models have long memory and which have short memory.

The Matlab script is given below and plots are shown in Figures 12.7a and 12.7b.

```
% Problem 12.7
```

```
close all;
clear;

ap=0.9; am=-0.9;
d=[-0.2 -0.4 0.2 0.4];
```

```

PSD1=zeros(1000,4);
PSD2=PSD1;
f=(1:1000)'/2000;
om=2*pi*f;

for i=1:length(d)
PSDfp=(2*sin(om/2)).^(-2*d(i));
PSDar1=(abs(1+ap*exp(j*om))).^2;
    PSD1(:,i)=PSDfp./PSDar1;
end

for i=1:length(d)
PSDfp=(2*sin(om/2)).^(-2*d(i));
PSDar1=(abs(1+am*exp(j*om))).^2;
    PSD2(:,i)=PSDfp./PSDar1;
end

loglog(f,PSD1);
xlabel('f');
ylabel('PSD')
title('a=0.9');

exportfig(gcf,'p1207a.eps','fontmode','scaled',...
    'linemode','scaled','color','cmyk');

figure
loglog(f,PSD2);
xlabel('f');
ylabel('PSD')
title('a=-0.9');

exportfig(gcf,'p1207b.eps','fontmode','scaled',...
    'linemode','scaled','color','cmyk');

```

12.8 Compute and plot the PSD of the FGN process, using the following approaches, and compare the results.

- (a) The definition $R_x(e^{j\omega}) = \sum_{l=-\infty}^{\infty} r_x(l)e^{-j\omega l}$ and formula (12.6.36) for the autocorrelation.
- (b) The theoretical formula (12.6.37).

The Matlab script is given below and plots are shown in Figures 12.8a and 12.8b.

```
% fBn auto and PSD
```

```
close all
clear
```

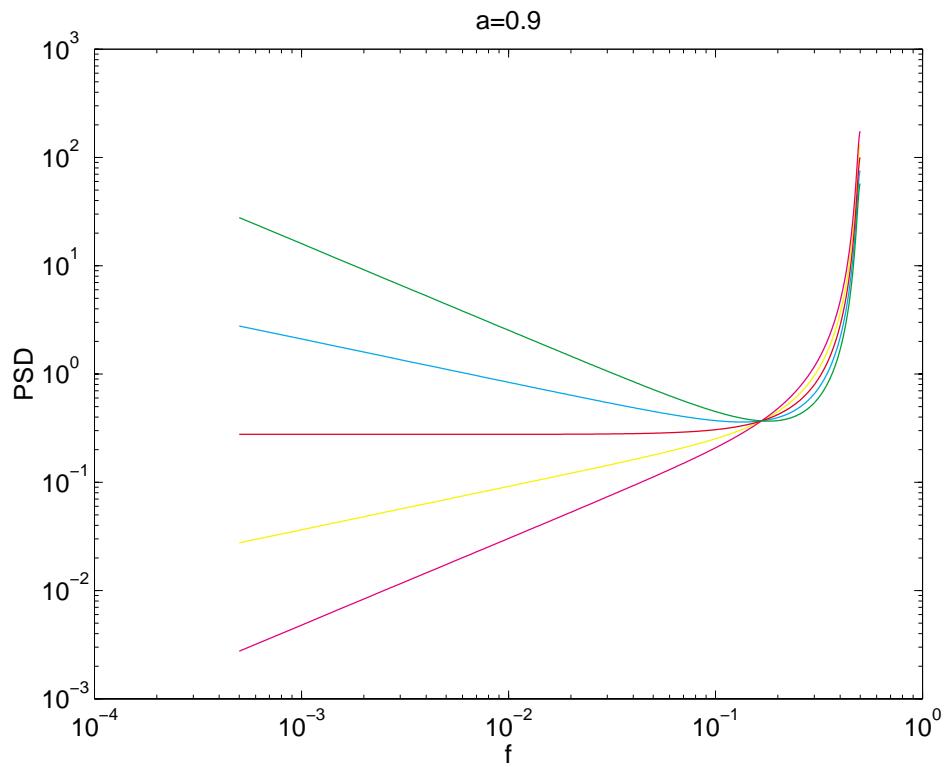


Figure 12.7a: PSD plot in 12.7a

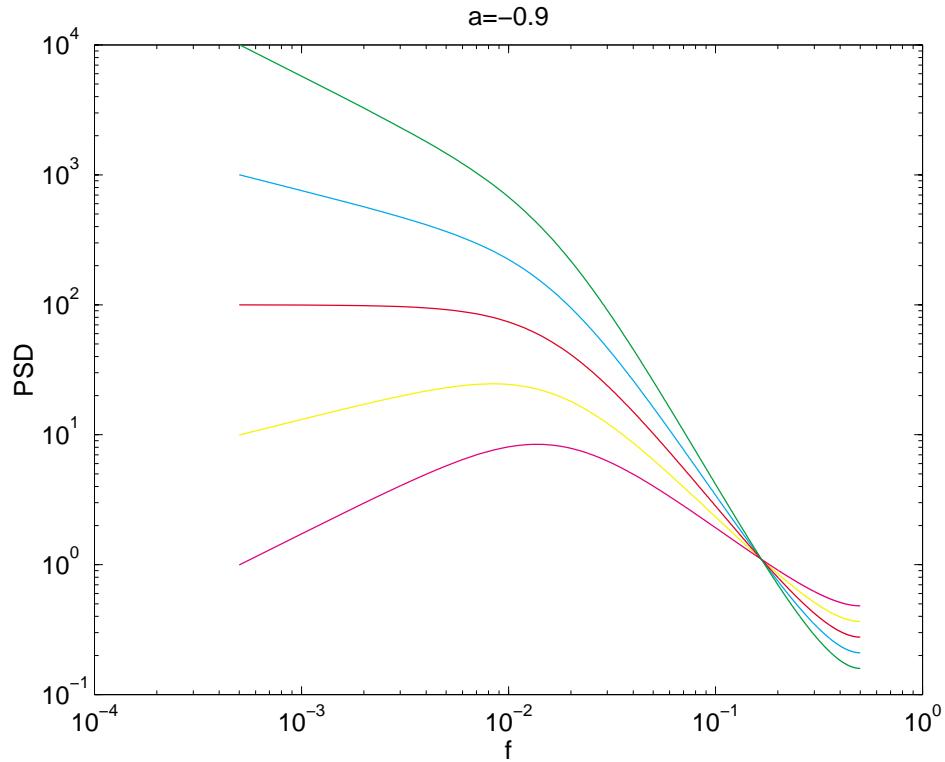


Figure 12.7b: PSD plot in 12.7b

```

H=0.3;
f=(0:0.001:0.5)';
om=2*pi*f;
K=50;
H1=(0.1:0.1:0.9)';
for i=1:length(H1)
    H=H1(i);
    al=zeros(size(om));
    for k=-K:1:K
        al=al+1./((abs(om+2*k*pi)).^(2*H+1));
    end
    Rx(:,i)=(abs(1-exp(-j*om))).^2;
    Rx1(:,i)=Rx(:,i).*al;
    end

loglog(f,Rx1); grid;
%xlabel('f');
%ylabel('R_x(e^{j2\pi f})')
xlabel('NORMALIZED FREQUENCY (F/Fs)')
ylabel('POWER SPECTRUM')

exportfig(gcf,'p1208a.eps','fontmode','scaled',...
    'linemode','scaled','color','cmyk');

figure
L=20;
l=(0:L)';
H1=(0.1:0.1:0.9)';
for i=1:length(H1)
    H=H1(i);
    rx(:,i)=(abs(l-1).^(2*H)-2*abs(l).^(2*H)+abs(l+1).^(2*H))/2;
end

plot(l,rx,'.-'); grid
%xlabel('\fontname{times}\it{l}');
%ylabel('\fontname{times}r_x(\it{l})');
xlabel('LAG')
ylabel('AUTOCORRELATION')

exportfig(gcf,'p1208b.eps','fontmode','scaled',...
    'linemode','scaled','color','cmyk');

```

- 12.9** Use the algorithm of Schür to develop a more efficient implementation of the fractional pole noise generation method described by Equations (12.5.24) to (12.5.28).

To be completed.

- 12.10** In this problem we study the properties of the harmonic fractional unit-pole model specified by the system

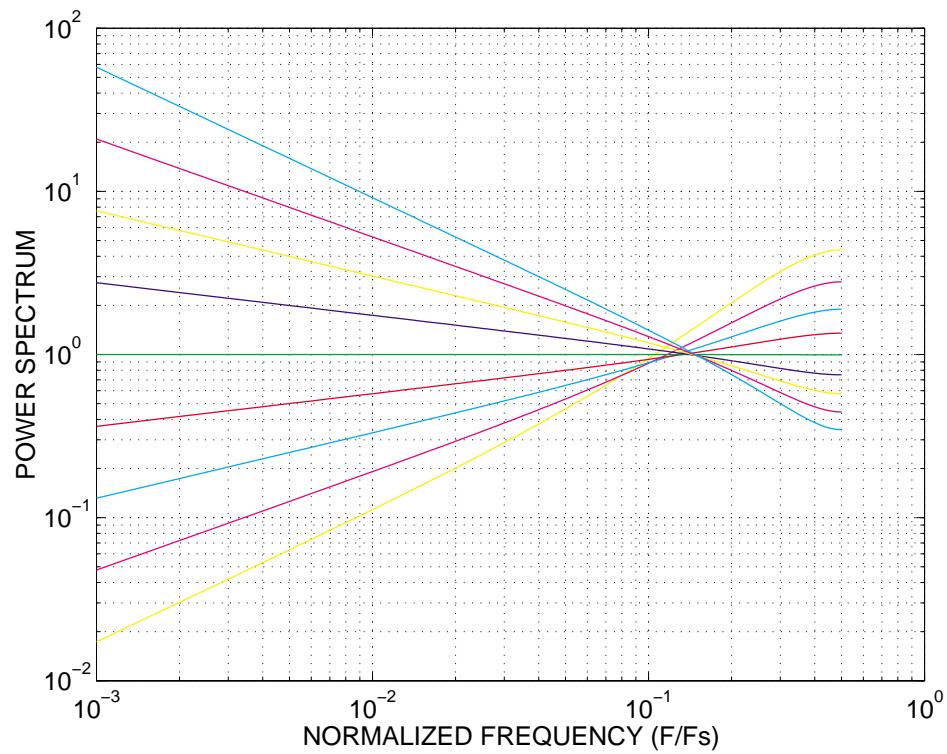


Figure 12.8a: Power spectrum plot in 12.8a

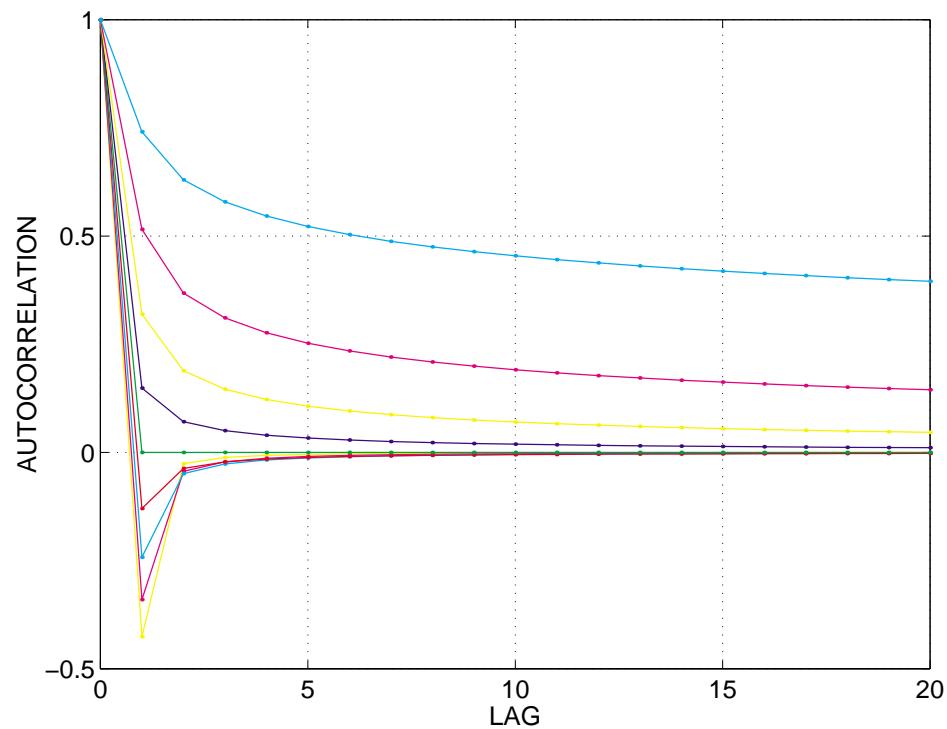


Figure 12.8b: Autocorrelation plot in 12.8b

function given by (12.5.32). The impulse response is given by

$$h_{\theta,d}(n) = \sum_{k=0}^{\lfloor n/2 \rfloor} \frac{(-1)^k \Gamma(d+n-k)(2 \cos \theta)^{n-2k}}{k!(n-2k)!\Gamma(d)}$$

where $\Gamma(\cdot)$ is the gamma function.

- (a) Compute and plot $h_{\theta,d}(n)$ for various values of θ and d .
- (b) Demonstrate the validity of the above formula by evaluating $h_{\theta,d}(n)$ from $H_{\theta,d}(z)$ for the same values of θ and d .
- (c) Illustrate that the model is minimum-phase if $|\cos \theta| < 1$ and $-\frac{1}{2} < d < \frac{1}{2}$ or $\cos \theta = \pm 1$ and $-\frac{1}{4} < d < \frac{1}{4}$.
- (d) Illustrate that the harmonic minimum-phase model, like the FPZ(0, d , 0) one, exhibits long-memory behavior only for positive values of d .
- (e) Show that for $0 < d < \frac{1}{4}$ and $\cos \theta = 1$, the autocorrelation equals that of the FPZ(0, $2d$, 0) model [multiplied by $(-1)^l$ if $\cos \theta = -1$]. When $|\cos \theta| < 1$ and $0 < d < \frac{1}{2}$, illustrate numerically that the autocorrelation can be approximated by $\rho(l) \sim -l^{2d-1} \sin(\theta l - \pi d)$ as $l \rightarrow \infty$.
- (f) Compute and plot the spectrum of the model for $\theta = \pi/3$ and various values of d .
- (g) Generate and plot realizations of Gaussian HFPZ noise for $\theta = \pi/6$ and $d = -0.3, 0.1$, and 0.4 .

To be completed.

12.11 Determine the variogram of the process $x(n)$ obtained by exciting the system

$$H(z) = \frac{1}{(1-z^{-1})(1-az^{-1})} \quad |a| < 1$$

with white noise $w(n) \sim \text{WGN}(0, \sigma_w^2)$.

The variogram of a process $x(n)$ is defined by

$$v_x(l) = \frac{1}{2} E\{[x(n+l) - x(n)]^2\}$$

If the process is stationary

$$v_x(l) = r_x(0) - r_x(l)$$

Unfortunately, the system $H(z)$ has a pole on the unit circle. Hence, the output process is nonstationary. To determine the variogram we note that $x(n)$ can be obtained by integrating an AR(1) process

$$\begin{aligned} s(n) &= as(n-1) + w(n) \\ x(n) &= x(n-1) + s(n) = \sum_{k=-\infty}^n s(k) \end{aligned}$$

Therefore

$$x(n+l) - x(n) = s(n+1) + \dots + s(n+l)$$

and

$$v_x(l) = \frac{1}{2}E\{[x(n+l) - x(n)]^2\} = \frac{1}{2}E\{[s(n+1) + \dots + s(n+l)]^2\}$$

The autocorrelation of $s(n)$ is

$$E\{s(n+l)s(n)\} = E\{s^2(n)\}a^l = \sigma^2 a^l$$

Hence

$$\begin{aligned} v_x(1) &= \frac{1}{2}E\{[s(n+1)]^2\} = \frac{1}{2}\sigma^2 \\ v_x(2) &= \frac{1}{2}E\{[s(n+1) + s(n+2)]^2\} = \frac{1}{2}(\sigma^2 + 2\sigma^2 a + \sigma^2) = \frac{1}{2}\sigma^2(2 + 2a) \\ v_x(3) &= \frac{1}{2}E\{[s(n+1) + s(n+2) + s(n+3)]^2\} = \frac{1}{2}\sigma^2(3 + 4a + 2a^2) \\ v_x(4) &= \frac{1}{2}\sigma^2(4 + 6a + 4a^2 + a^3) \end{aligned}$$

or in general

$$v_x(l) = \frac{1}{2}\sigma^2 \left[l + 2 \sum_{k=1}^{l-1} (l-k)a^k \right]$$

- 12.12** Following the steps leading to (12.6.26), show that the fractal (Haussdorff) dimension D is related to the Hurst exponent H by

$$D = 2 - H$$

Consider an arbitrary trace of a fractional Brownian motion with Hurst exponent H . Let us assume that this trace is enclosed by a box of length (time-axis) A and height (amplitude-axis) B . We will use the method of box counting. If we extract a piece of this trace of length rA where $0 \leq r \leq 1$ then to preserve self-similarity of the process, the *scaled* subbox needed to enclose the piece will have height $r^H B$. On the other hand, if we did not have this self-similarity property, then the height of the *unscaled* subbox would be rB . Hence the total number of such unscaled subboxes that we can fit in one scaled subbox is equal to

$$\frac{\text{area of scaled subbox}}{\text{area of unscaled subbox}} = \frac{(rA)(r^H B)}{(rA)(rB)} = \frac{1}{r^{1-H}}$$

However the total number of subtraces (of length rA) are $1/r$. Hence the total number of unscaled boxes (that is, the identical units carved out of the original one) are

$$N = \left(\frac{1}{r}\right) \left(\frac{1}{r^{1-H}}\right) = \frac{1}{r^{2-H}} = \frac{1}{r^D}$$

Hence

$$D = 2 - H$$

- 12.13** Develop a MATLAB function to generate the ordinary Brownian motion trace according to the steps given for the cumulative sum method in Section 12.6.3. The format of the function should be $x = \text{obm_cumsum}(N)$.

The MATLAB function $x = \text{obm_cumsum}(N)$ is given below.

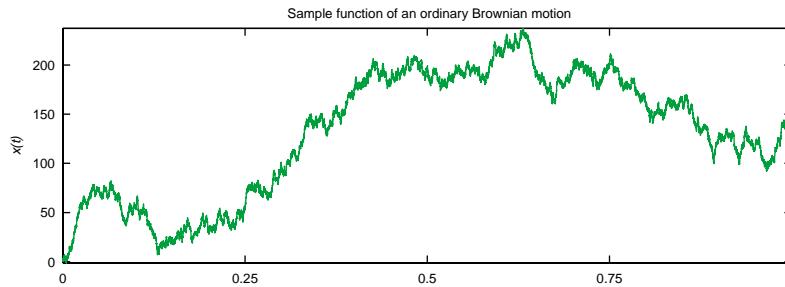


Figure 12.13a: Plot of a sample function of an ordinary Brownian motion

```
function x = obm_cumsum(N)
% Function to generate N samples of an ordinary Brownian motion trace x
% using the cumulative sum method
% x = obm_cumsum(N)

x = randn(N,1); x = cumsum(x);
```

- (a) Generate 16,384 samples of the Brownian motion $x(t)$ over $0 \leq t \leq 1$.

The MATLAB script is given below and the plot is shown in Figure 12.13a.

```
close all;
set(0,'defaultaxesfontsize',default_fontsize);

% (a) Generate 16,384 samples
N = 16384; n = [0:N-1]/N;
x = obm_cumsum(N);

Hf_1 = figure('Units',SCRUN,'position',SCRPOS,'color',[0,0,0],...
    'Paperunits',PAPUN,'PaperPosition',[0,0,4.9,1.5]);
set(Hf_1,'NumberTitle','off','Name','P1213a');

plot(n,x,'g'); axis([0,1,min(x),max(x)]);
xlabel('Time \it{t}', 'fontsize',label_fontsize);
ylabel('x(t)', 'fontsize',label_fontsize);
title('Sample function of an ordinary Brownian motion', 'fontsize',title_fontsize);
set(gca,'xtick',[0:0.25:1],'ytick',[floor(min(x)/50)*50:50:ceil(max(x)/50)*50]);
```

- (b) Investigate the self-affine property of $x(t)$ by reproducing a figure similar to Figure 12.23.

The MATLAB script is given below and the plots are shown in Figure 12.13b.

```
% (b) Self-affine property
Hf_2 = figure('Units',SCRUN,'position',SCRPOS,'color',[0,0,0],...
    'Paperunits',PAPUN,'PaperPosition',[0,0,4.9,4.5]);
set(Hf_2,'NumberTitle','off','Name','P1213b');

subplot(3,1,1); plot(n,x,'g'); axis([0,1,min(x),max(x)]); hold on;
plot([0.375,0.375],[min(x),max(x)],'w:',[0.625,0.625],[min(x),max(x)],'w:');
xlabel('Time \it{t}', 'fontsize',label_fontsize);
ylabel('x(t)', 'fontsize',label_fontsize);
```

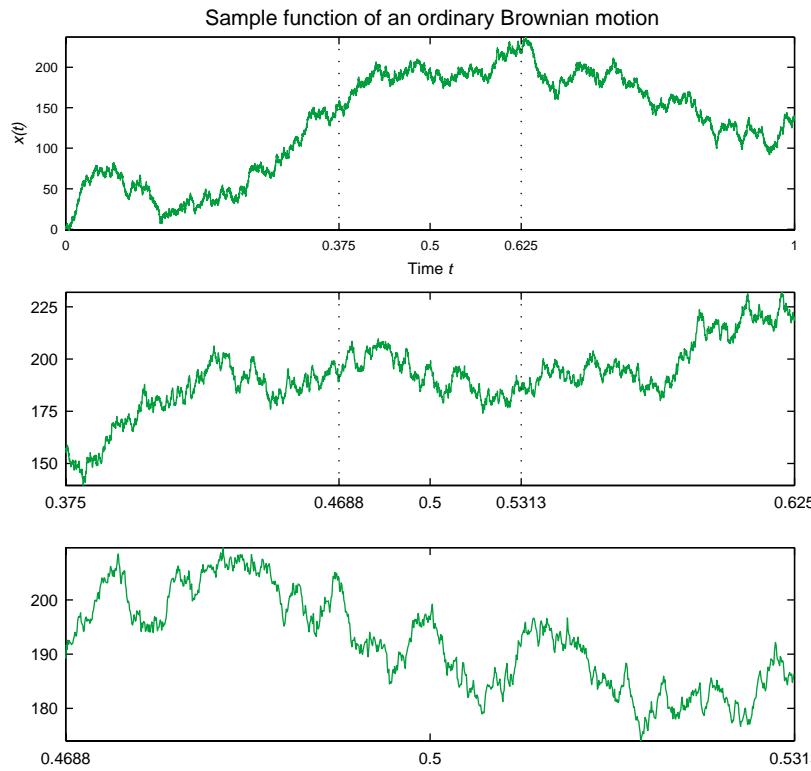


Figure 12.13b: Self-affine property of the ordinary Brownian motion

```

title('Sample function of an ordinary Brownian motion','fontsize',title_fontsize);
set(gca,'xtick',[0,0.375,0.5,0.625,1]);
set(gca,'ytick',[floor(min(x)/50)*50:50:ceil(max(x)/50)*50]); hold off;

% Zoom-1
subplot(3,1,2); plot(n(6145:1:10240),x(6145:1:10240),'g'); hold on;
axis([0.375,0.625,min(x(6145:1:10240)),max(x(6145:1:10240))]);
plot([15/32,15/32],[min(x(6145:1:10240)),max(x(6145:1:10240))],'w',...
[17/32,17/32],[min(x(6145:1:10240)),max(x(6145:1:10240))],'w');
set(gca,'xtick',[3/8,15/32,1/2,17/32,5/8],'fontsize',8);
set(gca,'ytick',[floor(min(x(6145:1:10240))/25)*25:25:...
ceil(max(x(6145:1:10240))/25)*25]); hold off;
hold off;

% Zoom-2
subplot(3,1,3); plot(n(7681:1:8704),x(7681:1:8704),'g');
axis([15/32,17/32,min(x(7681:1:8704)),max(x(7681:1:8704))]);
set(gca,'xtick',[15/32,1/2,17/32],'fontsize',8);
set(gca,'ytick',[floor(min(x(7681:1:8704))/10)*10:10:...
ceil(max(x(7681:1:8704))/10)*10]); hold off;

```

- 12.14** Develop a MATLAB function to generate the fractional Brownian motion trace according to the steps given for the spectral synthesis method in Section 12.6.3. The format of the function should be $x = \text{fbm_spectral}(H, N)$.

The MATLAB function $x = \text{fbm_spectral}(H, N)$ is given below.

```

function x = fbm_spectral(H,N);
% Function to generate N samples of a fraction Brownian motion trace x
% of self-similarity index H using the Spectral Synthesis method
% x = fbm_spectral(H,N)

N1 = N; N = 2^nextpow2(2*N1);
beta = 2*H+1;

x = randn(N,1);
X = fft(x,2*N);
phase = (rand(N+1,1)-0.5)*pi;
k = (1:N)';
X_H = 10*(k.^(-beta/2)).*abs(X(2:N+1));
X_H = [0; X_H; flipud(X_H(1:N-1))];
phase = [phase;-flipud(phase(2:N))];
X_H = X_H.*exp(j*phase);
x_H = ifft(X_H); N = length(x_H);
x_H = real(x_H(N/2-N1/2+1:N/2+N1/2));
x = x_H;

```

- (a) Generate 1024 samples of the FBM $B_H(t)$ over $0 \leq t \leq 1$ for $H = 0.3$. Investigate the self-affine property of $B_{0.3}(t)$.

The MATLAB script is given below and the plots are shown in Figure 12.14a.

```

close all; clc;
set(0,'defaultaxesfontsize',default_fontsize);

% (a) Generate 1024 samples for H = 0.3
N = 1024; n = [0:N-1]/N;
H = 0.3; x = fbm_spectral(H,N);

% (a1) Self-affine property
Hf_1 = figure('Units','SCRUN','position',SCRPOS,'color',[0,0,0],...
    'Paperunits',PAPUN,'PaperPosition',[0,0,4.9,4.5]);
set(Hf_1,'NumberTitle','off','Name','P1214a');

subplot(2,1,1); plot(n,x,'g'); axis([0,1,min(x),max(x)]); hold on;
plot([0.375,0.375],[min(x),max(x)],'w:',[0.625,0.625],[min(x),max(x)],'w:');
xlabel('Time \itt','fontsize',label_fontsize);
ylabel('\it{x}(\it{t})','fontsize',label_fontsize);
title('Sample function of a fractional Brownian motion: H = 0.3',...
    'fontsize',title_fontsize);
set(gca,'xtick',[0,0.375,0.5,0.625,1]); vr = 0.1;
set(gca,'ytick',[floor(min(x)/vr)*vr:vr:ceil(max(x)/vr)*vr]); hold off;

% Zoom-1
I1 = (3/8)*N+1; I2 = (5/8)*N+1;
subplot(2,1,2); plot(n(I1:1:I2),x(I1:1:I2),'g'); hold on;
axis([0.375,0.625,min(x(I1:1:I2)),max(x(I1:1:I2))]);

```

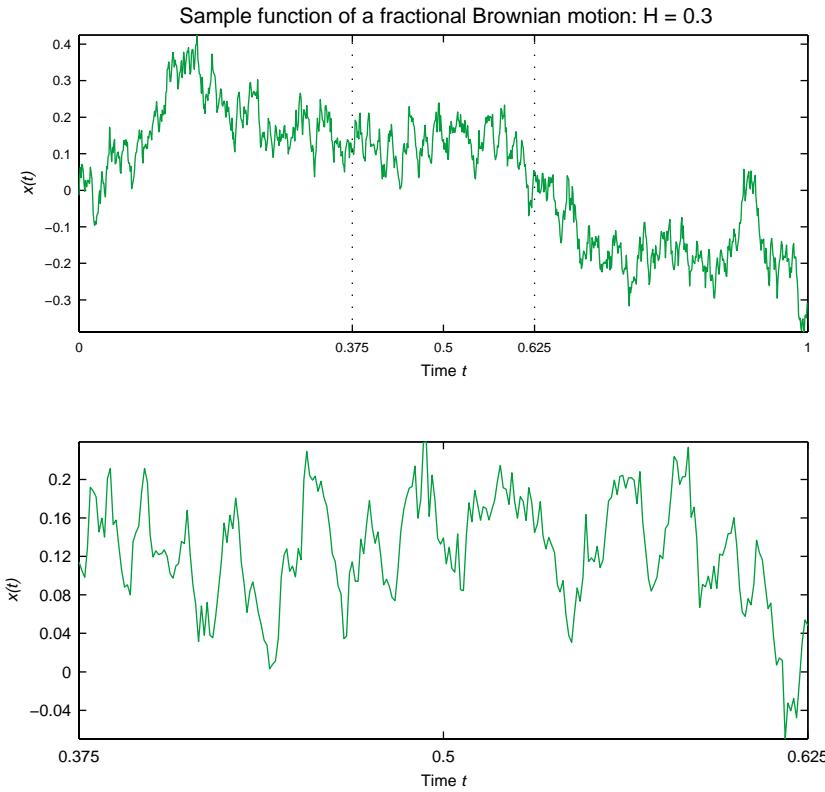


Figure 12.14a: Plots of a fractional Brownian motion with $H = 0.3$

```

xlabel('Time \itt','fontsize',label_fontsize);
ylabel('\it{x}(\it{t})','fontsize',label_fontsize);
set(gca,'xtick',[3/8,1/2,5/8],'fontsize',8); vr = 0.04;
set(gca,'ytick',[floor(min(x(I1:1:I2))/vr)*vr:vr:...
ceil(max(x(I1:1:I2))/vr)*vr]); hold off;

```

- (b) Generate 1024 samples of the FBM $B_H(t)$ over $0 \leq t \leq 1$ for $H = 0.7$. Investigate the self-affine property of $B_{0.7}(t)$.

The MATLAB script is given below and the plots are shown in Figure 12.14b.

```

% (b) Generate 1024 samples for H = 0.7
N = 1024; n = [0:N-1]/N;
H = 0.7; x = fbm_spectral(H,N);

% (b1) Self-affine property
Hf_1 = figure('Units',SCRUN,'position',SCRPOS,'color',[0,0,0],...
'Paperunits',PAPUN,'PaperPosition',[0,0,4.9,4.5]);
set(Hf_1,'NumberTitle','off','Name','P1214b');

subplot(2,1,1); plot(n,x,'g'); axis([0,1,min(x),max(x)]); hold on;
plot([0.375,0.375],[min(x),max(x)],'w:',[0.625,0.625],[min(x),max(x)],'w:');
xlabel('Time \itt','fontsize',label_fontsize);
ylabel('\it{x}(\it{t})','fontsize',label_fontsize);
title('Sample function of a fractional Brownian motion: H = 0.7',...
'fontsize',title_fontsize);

```

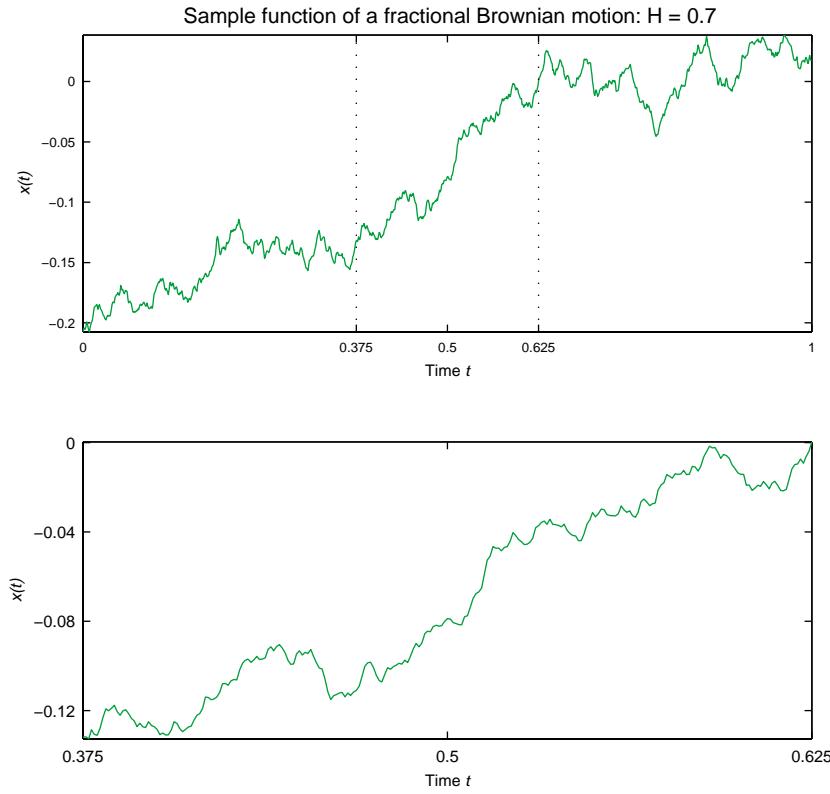


Figure 12.14b: Plots of a fractional Brownian motion with $H = 0.7$

```

set(gca,'xtick',[0,0.375,0.5,0.625,1]); vr = 0.05;
set(gca,'ytick',[floor(min(x)/vr)*vr:vr:ceil(max(x)/vr)*vr]); hold off;

% Zoom-1
I1 = (3/8)*N+1; I2 = (5/8)*N+1;
subplot(2,1,2); plot(n(I1:1:I2),x(I1:1:I2),'g'); hold on;
axis([0.375,0.625,min(x(I1:1:I2)),max(x(I1:1:I2))]);
xlabel('Time \itt','fontsize',label_fontsize);
ylabel('it{x}(it{t})','fontsize',label_fontsize);
set(gca,'xtick',[3/8,1/2,5/8],'fontsize',8); vr = 0.04;
set(gca,'ytick',[floor(min(x(I1:1:I2))/vr)*vr:vr:...
ceil(max(x(I1:1:I2))/vr)*vr]); hold off;

```

- 12.15** Develop a MATLAB function to generate the fractional Brownian motion trace according to the steps given for the random midpoint replacement method in Section 12.6.3. The format of the function should be $x = \text{fbm_replace}(N)$.

The MATLAB function $x = \text{fbm_replace}(H, N)$ is given below.

```

function x = fbm_replace(H,N)
% This function generates a fractional Brownian motion with a given
% self-similarity index H, using the midpoint displacement method.
% The function receives as inputs, the self-similarity index H,
% and the number of points N to be generated.
%

```

```
% The function has the following format:
%
% x = fbm_replace(H,N)
%
%     H = self-similarity index
%     N = number of points to be generated
%
%
sigma = 1.0; alpha = 2;
f(1) = 0; % initialize the endpoints
f(2) = 1;
M = 2;
int = round(log(N)/log(2) + 0.5);
for j = 1:int
    delta = sqrt(((1 - 2^(2 * H - 2)) * 2^(-2*H*j))/2) * sigma;
    stable_rv = stable(alpha,0,M-1,1);
    for i = 1:M-1
        new_f(2*i-1) = f(i);
        new_f(2*i) = (f(i) + f(i+1))/2 + delta * stable_rv(i);
    end;
    new_f(2*M-1) = f(M);
    M = 2*M - 1;
    f = new_f;
end
f = f - ((1:M)-1)/M;
x = f(1:N);
```

- (a) Generate 1024 samples of the FBM $B_H(t)$ over $0 \leq t \leq 1$ for $H = 0.5$. Compare visually $B_{0.5}(t)$ with that obtained by using the cumulative-sum method. Comment on your observations.

The MATLAB script is given below and the plots are shown in Figure 12.15a. Visually the plots appear to be similar.

```
close all; clc;
set(0,'defaultaxesfontsize',default_fontsize);

% (a) Generate 1024 samples for H = 0.5 and compare with the cumsum method
N = 1024; n = [0:N-1]/N;
H = 0.5; x1 = fbm_replace(H,N); x2 = obm_cumsum(N);

Hf_1 = figure('Units','SCRUN','position',SCRPOS,'color',[0,0,0],...
    'Paperunits',PAPUN,'PaperPosition',[0,0,4.9,4.5]);
set(Hf_1,'NumberTitle','off','Name','P1215a');

subplot(2,1,1); plot(n,x1,'g'); axis([0,1,min(x1),max(x1)]);
xlabel('Time \itt','fontsize',label_fontsize);
ylabel('\it{x}(\it{t})','fontsize',label_fontsize);
title('Sample function of an ordinary Brownian motion: Replacement',...
    'fontsize',title_fontsize);
```

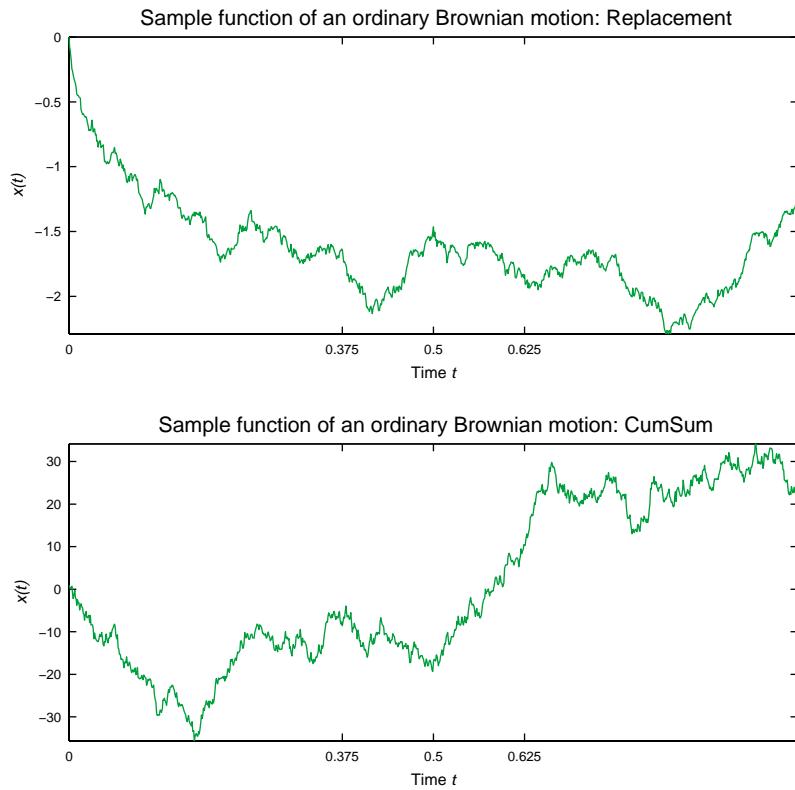


Figure 12.15a: Comparison of the ordinary Brownian motion plots using the replacement and the cumsum methods

```

set(gca,'xtick',[0,0.375,0.5,0.625,1]); vr = 0.1;

% Zoom-1
subplot(2,1,2); plot(n,x2,'g'); axis([0,1,min(x2),max(x2)]);
xlabel('Time \it{t}', 'fontsize',label_fontsize);
ylabel('\it{x}(\it{t})', 'fontsize',label_fontsize);
title('Sample function of an ordinary Brownian motion: CumSum',...
      'fontsize',title_fontsize);
set(gca,'xtick',[0,0.375,0.5,0.625,1]); vr = 0.1;

```

- (b) Generate 1024 samples of the FBM $B_H(t)$ over $0 \leq t \leq 1$ for $H = 0.99$. Investigate the artifact discussed in the chapter for $H \rightarrow 1$.

The MATLAB script is given below and the plots are shown in Figure 12.15b. Visually, the trace has self repeating (but increasing) structures. Thus the statistical properties will be different at different times and at different scales.

```

% (b) Generate 1024 samples for H = 0.99
N = 1024; n = [0:N-1]/N;
H = 0.99; x = ffbm_replace(H,N);

% (b1) Self-affine property
Hf_2 = figure('Units','SCRUN','position',SCRPOS,'color',[0,0,0],...
    'Paperunits',PAPUN,'PaperPosition',[0,0,4.9,1.5]);
set(Hf_2,'NumberTitle','off','Name','P1215b');

```

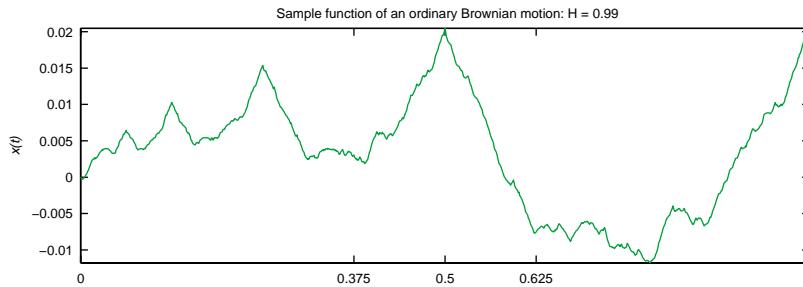


Figure 12.15b: Plots of a fractional Brownian motion with $H = 0.99$

```

plot(n,x,'g');
axis([0,1,min(x),max(x)]);
xlabel('Time \it{t}','fontsize',label_fontsize);
ylabel('\it{x}(\it{t})','fontsize',label_fontsize);
title('Sample function of an ordinary Brownian motion: H = 0.99',...
      'fontsize',title_fontsize);
set(gca,'xtick',[0,0.375,0.5,0.625,1]);

```

- 12.16** Based on Equation (12.6.54), develop a MATLAB function $[H, \text{sigma}_H] = \text{est_H_mad}(x)$ that computes an estimate of the self-similarity index H and the variance σ_H^2 of an FBM process.

The MATLAB function $[H, \text{sigma}] = \text{est_H_mad}(x)$ is given below.

```

function [H,sigma] = est_H_mad(x)
%
% function [H,sigma] = est_H_mad(x)
%
% This program calculates the fractal dimension of an input signal
% f(n) as described in the article 'Fractal-Based Description of
% Natural Scenes' by Alex Pentland in IEEE Transactions on Pattern
% Analysis and Machine Intelligence, Nov. 1984.
%
N = length(x);
maxscale = min(round(0.1*N),100);

% Find the expected values of the absolute value of f(x + delta) - f(x)
% for delta = 1 to maximum scale.

for delta = 1:maxscale
    scale(delta) = delta;
    expdif(delta) = (1/(N-delta))*sum(abs(x(1+delta:N)-x(1:N-delta)));
end

% Create the self-similarity plot, with the initial estimate of the fit

c_init = polyfit(log(scale),log(expdif),1);
fit_init = polyval(c_init,log(scale));

```

```

loglog(scale,expdif,'.',scale,exp(fit_init));
d_est = 2 - c_init(1);
sigma = - c_init(2);
H = 2-d_est;
title(['Estimated H = ',num2str(H)]);
xlabel('Scale \Delta');
ylabel('E{|x(n+\Delta) - x(n)|}');
grid;

```

- (a) Use function $x = \text{fbm_replace}(N)$ to generate $N = 1024$ samples of an FBM process with $H = 0.3$, and use the function $[H, \sigma_H] = \text{est_H_mad}(x)$ to estimate H and σ_H .

The MATLAB script is given below and the plots are shown in Figure 12.16a.

```

close all; clc;
set(0,'defaultaxesfontsize',10);

% (a) Generate 1024 samples for H = 0.3 using the replacement method
% and estimate H and the process initial variance
Hf_1 = figure('Units','SCRUN','position',SCRPOS,'color',[0,0,0],...
    'Paperunits',PAPUN,'PaperPosition',[0,0,4.9,4.5]);
set(Hf_1,'NumberTitle','off','Name','P1216a');

N = 1024; n = [0:N-1]/N;
H = 0.3; x = fbm_replace(H,N);
[H,sigma] = est_H_mad(x)
H =
    0.26705987484094
sigma =
    2.35161153837552

```

- (b) Repeat the previous task for $H = 0.7$.

The MATLAB script is given below and the plots are shown in Figure 12.16b.

```

% (b) Generate 1024 samples for H = 0.7 using the replacement method
% and estimate H and the process initial variance
Hf_1 = figure('Units','SCRUN','position',SCRPOS,'color',[0,0,0],...
    'Paperunits',PAPUN,'PaperPosition',[0,0,4.9,4.5]);
set(Hf_1,'NumberTitle','off','Name','P1216b');

N = 1024; n = [0:N-1]/N;
H = 0.7; x = fbm_replace(H,N);
[H,sigma] = est_H_mad(x)
H =
    0.64831632433527
sigma =
    5.63245919105432

```

- (c) Perform a Monte Carlo simulation using 100 trials and compute the mean and standard deviation of the estimates for H and σ_H in (a) and (b). To be completed.

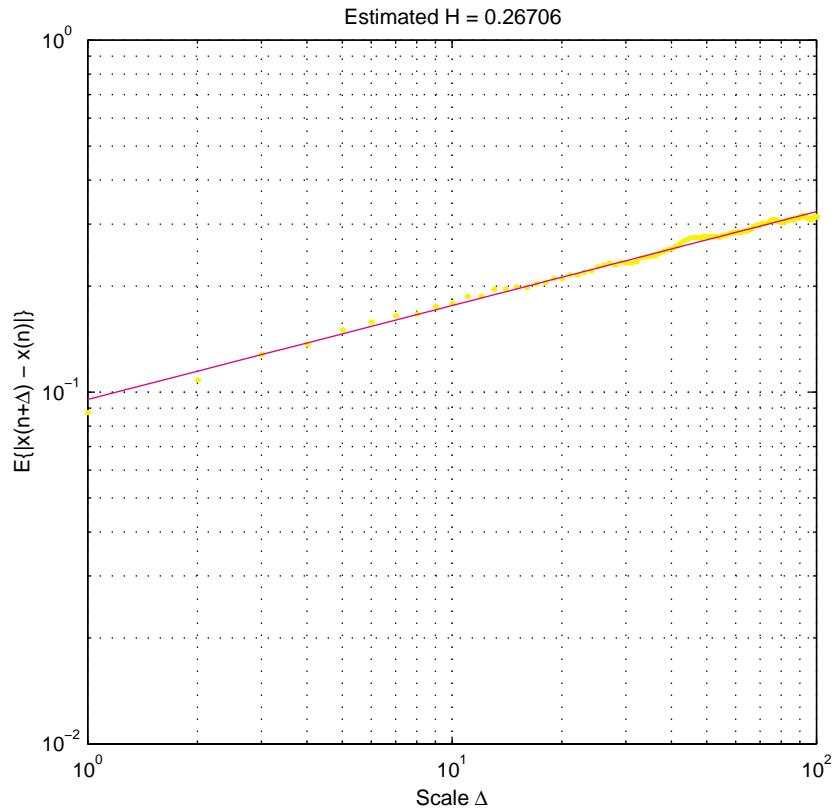


Figure 12.16a: The log-log plot for the estimation of H given a trace using $H = 0.3$

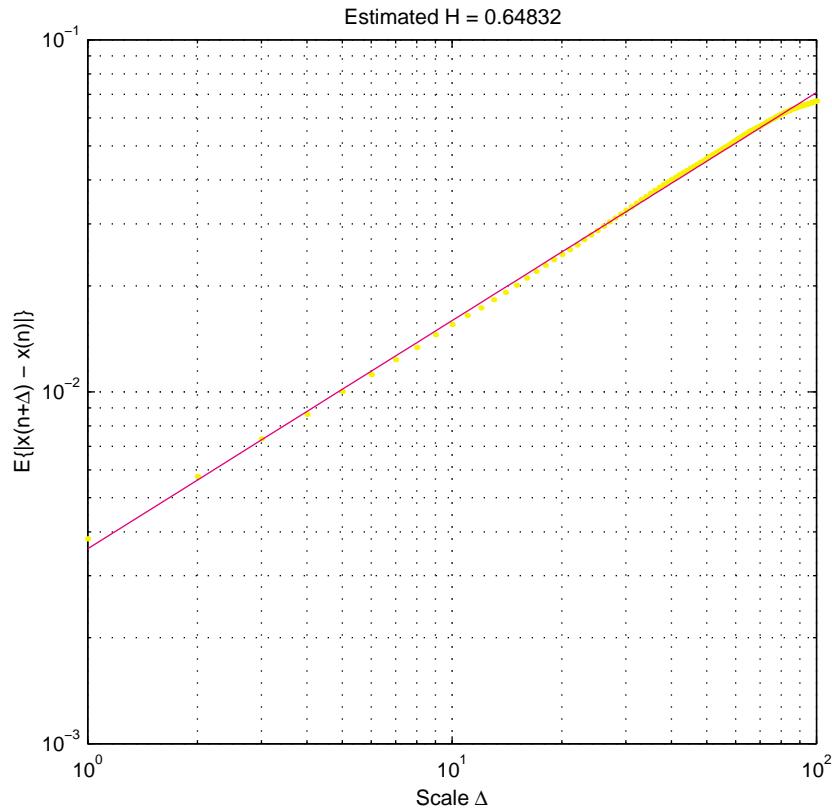


Figure 12.16b: The log-log plot for the estimation of H given a trace using $H = 0.7$

- 12.17** Repeat Problem 12.16 by developing a function that estimates the self-similarity index H by determining the slope of the first 10 percent values of the periodogram in a log-log plot.

The MATLAB function `[H,sigmaH] = est_H_psd(x)` is given below.

```
function [H] = est_H_psd(x)
% Estimates the self-similarity index by determining the slope of the
% first 10% of the periodogram values in log-log scale.
% [H] = est_H_psd(x)

M = nextpow2(length(x)*4);
NFFT = 2^M;
[R,F] = psd(x,NFFT,1.0,'none');

l2f = log2(F(2:end));
l2R = log2(R(2:end));

s = [2:NFFT*0.05];
cr1 = polyfit(l2f(s),l2R(s),1);
crf1 = polyval(cr1,l2f(s));
beta1 = cr1(1);
H = (-beta1-1)/2;
```

The MATLAB script is given below.

```
close all; clc;
set(0,'defaultaxesfontsize',10);

% (a) Generate 1024 samples for H = 0.3 using the replacement method
% and estimate H
N = 1024; n = [0:N-1]/N;
H = 0.3; x = fbm_replace(H,N);
[H] = est_H_psd(x)
H =
0.30246903766353

% (b) Generate 1024 samples for H = 0.7 using the replacement method
% and estimate H
N = 1024; n = [0:N-1]/N;
H = 0.7; x = fbm_replace(H,N);
[H] = est_H_psd(x)
H =
0.61908068053019
```