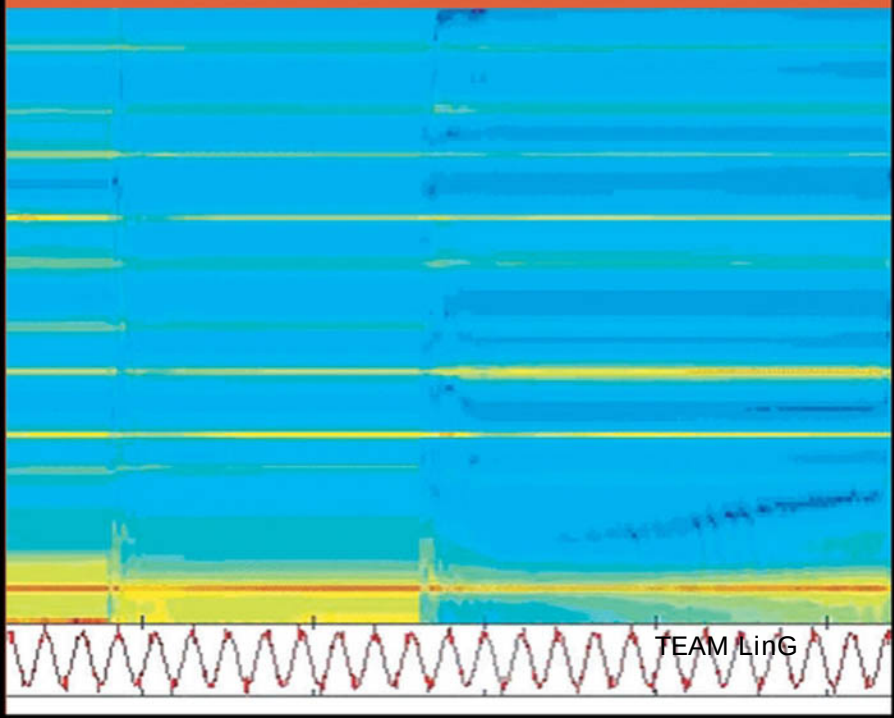
*Wiley Series in Adaptive and Learning Systems for Signal Processing, Communications, and Control*
*Simon Haykin, Series Editor*

# Model-Based Signal Processing

*James V. Candy*



# MODEL-BASED SIGNAL PROCESSING

# MODEL-BASED SIGNAL PROCESSING

James V. Candy

Lawrence Livermore National Laboratory
University of California
Santa Barbara, CA

IEEE PRESS

A JOHN WILEY & SONS, INC., PUBLICATION

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.



***Library of Congress Cataloging-in-Publication Data:***

Candy, J. V.
Model-based signal processing / James V. Candy.
p. cm.
ISBN-13 978-0-471-23632-0 (cloth)
ISBN-10 0-471-23632-2 (cloth)
1. Signal processing—Digital techniques—Textbooks. I. Title.
TK5102.9.C32 2005
621.382'2–dc22

2004065042

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

Praise the Lord, Jesus Christ, Our Savior! In times of great need and distress—He comforts us!

# CONTENTS

# PREFACE

This text develops the "model-based approach" to signal processing for a variety of useful model-sets, including what has become popularly termed "physics-based" models. It presents a unique viewpoint of signal processing from the model-based perspective. Although designed primarily as a graduate text, it will prove useful to practicing signal processing professionals and scientists, since a wide variety of case studies are included to demonstrate the applicability of the model-based approach to real-world problems. The prerequisite for such a text is a melding of undergraduate work in linear algebra, random processes, linear systems, and digital signal processing. It is somewhat unique in the sense that many texts cover some of its topics in piecemeal fashion. The underlying model-based approach of this text is uniformly developed and followed throughout in the algorithms, examples, applications, and case studies. It is the model-based theme, together with the developed hierarchy of physics-based models, that contributes to its uniqueness. This text has evolved from two previous texts, Candy ([1], [2]) and has been broadened by a wealth of practical applications to real-world model-based problems.

The place of such a text in the signal processing textbook community can best be explained by tracing the technical ingredients that form its contents. It can be argued that it evolves from the digital signal processing area, primarily from those texts that deal with random or statistical signal processing or possibly more succinctly "signals contaminated with noise." The texts by Kay ([3], [4], [5]), Therrien [6], and Brown [7] provide the basic background information in much more detail than this text, so there is little overlap with them.

This text additionally prepares the advanced senior or graduate student with enough theory to develop a fundamental basis and go onto more rigorous texts like Jazwinski [8], Sage [9], Gelb [10], Anderson [11], Maybeck [12], Bozic [13],

Kailath [14], and more recently, Mendel [15], Grewel [16], and Bar-Shalom [17]. These texts are rigorous and tend to focus on Kalman filtering techniques, ranging from continuous to discrete with a wealth of detail in all of their variations. The model-based approach discussed in this text certainly includes the state-space models as one of its model classes (probably the most versatile), but the emphasis is on various classes of models and how they may be used to solve a wide variety of signal processing problems. Some more recent texts of about the same technical level, but again, with a different focus, are Widrow [18], Orfanidis [19], Sharf [20], Haykin [21], Hayes [22], Brown [7], and Stoica [23]. Again, the focus of these texts is not the model-based approach but rather a narrow set of specific models and the development of a variety of algorithms to estimate them. The system identification literature and texts therein also provide some overlap with this text, but the approach is again focused on estimating a model from noisy data sets and is not really aimed at developing a model-based solution to a particular signal processing problem. The texts in this area are Ljung ([24], [25]), Goodwin [26], Norton [27] and Soderstrom [28].

The approach we take is to introduce the basic idea of model-based signal processing (MBSP) and show where it fits in terms of signal processing. It is argued that MBSP is a natural way to solve basic processing problems. The more a priori information we know about data and its evolution, the more information we can incorporate into the processor in the form of mathematical models to improve its overall performance. This is the theme and structure that echoes throughout the text. Current applications (e.g., structures, tracking, equalization, and biomedical) and simple examples to motivate the organization of the text are discussed. Next, in Chapter 2, the "basics" of stochastic signals and systems are discussed, and a suite of models to be investigated in the text, going from simple time series models to state-space and wave-type models, is introduced. The state-space models are discussed in more detail because of their general connection to "physical models" and their availability limited to control and estimation texts rather than the usual signal processing texts. Examples are discussed to motivate all the models and prepare the reader for further developments in subsequent chapters. In Chapter 3, the basic estimation theory required to comprehend the model-based schemes that follow are developed establishing the groundwork for performance analysis (bias, error variance, Cramer-Rao bound, etc.). The remaining chapters then develop the model-based processors for various model sets with real-world-type problems discussed in the individual case studies and examples. Chapter 4 develops the model-based scheme for the popular model sets (*AR*, *MA*, *ARMA*, etc.) abundant in the signal processing literature and texts today, following the model-based approach outlined in the first chapter and presenting the unified framework for the algorithms and solutions. Highlights of this chapter include the real-world case studies as well as the "minimum variance" approach to processor design along with accompanying performance analysis. Next we begin to lay the foundation of the "physics-based" processing approach by developing the linear state-space, Gauss-Markov model-set, leading to the well-known Kalman filter solution in Chapter 5. The Kalman filter is developed from the innovations viewpoint with its optimality properties

analyzed within. The solution to the minimum variance design is discussed (tuned filter) along with a "practical" cookbook approach (validated by theory). Next critical special extensions of the linear filter are discussed along with a suite of solutions to various popular signal processing problems (identification, deconvolution/bias estimation, etc.). Here the true power of the model-based approach using state-space models is revealed and developed for difficult problems that are easily handled within this framework. A highlight of the chapter is a detailed processor design for a storage tank, unveiling all of the steps required to achieve a minimum variance design. Chapter 6 extends these results even further to the case of nonlinear state-space models. Theoretically each processor is developed in a logical fashion leading to some of the more popular structures with example problems throughout. This chapter ends with a case study of tracking the motion of a super tanker during docking. Next the adaptive version of the previous algorithms is developed, again, within the model-based framework. Here many interesting and exciting examples and applications are presented along with some detailed case studies demonstrating their capability when applied to real-world problems. Here, in Chapter 7, we continue with the basic signal processing models and apply them to a suite of applications. Next, in Chapter 8, we extend the state-space model sets (linear and nonlinear) to the adaptive regime. We develop the adaptive Kalman-type filters and apply them to a real-world ocean acoustic application (case study). Finally, in Chapter 9, we develop a suite of physics-based models ranging from reentry vehicle dynamics (*ARMAX*), to nondestructive evaluation using laser ultrasound (*FIR*), to a suite of state-space models for vibrating structures, ocean acoustics, dispersive waves, and distributed groundwater flow. In each case the processor along with accompanying simulations is discussed and applied to various data sets, demonstrating the applicability and power of the model-based approach.

In closing, we must mention some of the new and exciting work currently being performing in nonlinear estimation. Specifically, these are the unscented Kalman filter [29] (Chapter 6), which essentially transforms the nonlinear problem into an alternate space without linearization (and its detrimental effects) to enhance performance, and the particle filter, which uses probabilistic sampling-resampling theory (Markov chain/Monte Carlo methods) (MCMC) to handle the non-gaussian type problems. Both approaches are opening new avenues of thought in estimation that has been stagnant since the 1970s. These approaches have evolved because of the computer power (especially the MCMC techniques) now becoming available ([29], [30]).

JAMES V. CANDY
*Danville, CA*

## REFERENCES


1. J. Candy, *Signal Processing: The Model-Based Approach*, New York: McGraw-Hill, 1986.
2. J. Candy, *Signal Processing: The Modern Approach*, New York: McGraw-Hill, 1988.
3. S. Kay, *Modern Spectral Estimation: Theory and Applications*, Englewood Cliffs, NJ: Prentice-Hall, 1988.
4. S. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*, Englewood Cliffs, NJ: Prentice-Hall, 1993.
5. S. Kay, *Fundamentals of Statistical Signal Processing: Detection Theory*, Englewood Cliffs, NJ: Prentice-Hall, 1998.
6. C. Therrien, *Random Signal Processing: Detection Theory*, Englewood Cliffs, NJ: Prentice-Hall, 1991.
7. R. Brown and P. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*, New York: Wiley, 1997.
8. A. Jazwinski, *Stochastic Processes and Filtering Theory*, New York: Academic Press, 1970.
9. A. Sage and J. Melsa, *Estimation Theory with Applications to Communications and Control*, New York: McGraw-Hill, 1971.
10. A. Gelb, *Applied Optimal Estimation*, Cambridge: MIT Press, 1974.
11. B. Anderson and J. Moore, *Optimum Filtering*, Englewood Cliffs, NJ: Prentice-Hall, 1979.
12. P. Maybeck, *Stochastic Models, Estimation and Control*, New York: Academic Press, 1979.
13. M. S. Bozic, *Linear Estimation Theory*, Englewood Cliffs, NJ: Prentice-Hall, 1979.
14. T. Kailath, *Lectures on Kalman and Wiener Filtering Theory*, New York: Springer-Verlag, 1981.
15. J. Mendel, *Lessons in Estimation Theory for Signal Processing, Communications, and Control*, Englewood Cliffs, NJ: Prentice-Hall, 1995.
16. M. Grewal and A. Andrews, *Kalman Filtering: Theory and Practice*, Englewood Cliffs, NJ: Prentice-Hall, 1993.
17. Y. Bar-Shalom and X. Li, *Estimation and Tracking: Principles, Techniques and Software*, Norwood, MA: Artech House, 1993.
18. B. Widrow and S. Stearns, *Adaptive Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1985.
19. S. Orfanidis, *Optimal Signal Processing*, New York: Macmillan, 1988.
20. L. Sharf, *Statistical Signal Processing: Detection, Estimation and Time Series Analysis*, Reading, MA: Addison-Wesley, 1991.
21. S. Haykin, *Adaptive Filter Theory*, Englewood Cliffs, NJ: Prentice-Hall, 1993.
22. M. Hayes, *Statistical Digital Signal Processing and Modeling*, New York: Wiley, 1996.
23. P. Stoica and R. Moses, *Introduction to Spectral Analysis*, Englewood Cliffs, NJ: Prentice-Hall, 1997.
24. L. Ljung, *System Identification: Theory for the User*, Englewood Cliffs, NJ: Prentice-Hall, 1987.

25. L. Ljung and T. Soderstrom, *Theory and Practice of Recursive Identification*, Cambridge: MIT Press, 1983.
26. G. Goodwin and K. S. Sin, *Adaptive Filtering, Prediction and Control*, Englewood Cliffs, NJ: Prentice-Hall, 1984.
27. J. Norton, *An Introduction to Identification*, New York: Academic Press, 1986.
28. T. Soderstrom and P. Stoica, *System Identification*, New York: Academic Press, 1989.
29. S. Haykin and N. de Freitas, eds., "Sequential state estimation: From Kalman filters to particle filters," *Proc. IEEE*, **92** (3), 399–574, 2004.
30. P. Djuric, J. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. Bugallo, and J. Miguez, "Particle filtering," *IEEE Signal Proc. Mag.*, **20** (5), 19–38, 2003.

# ACKNOWLEDGMENTS

My beautiful wife, Patricia, and children, Kirstin and Christopher, have shown extraordinary support and patience during the writing of this text. I would like to thank Dr. Simon Haykin whose kindness and gentle nudging helped me achieve the most difficult "first step" in putting together this concept. My colleagues, especially Drs. Edmund Sullivan and David Chambers, have always been inspirational sources of motivation and support in discussing many ideas and concepts that have gone into the creation of this text. The deep questions by UCSB student, Andrew Brown, led to many useful insights during this writing. God bless you all.

JVC

# 1

# INTRODUCTION

## 1.1 BACKGROUND

Perhaps the best way to start a text such as this is through an example that will provide the basis for this discussion and motivate the subsequent presentation. The processing of noisy measurements is performed with one goal in mind—to extract the desired information and reject the extraneous [1]. In many cases this is easier said than done. The first step, of course, is to determine what, in fact, *is* the desired information, and typically this is not the task of the signal processor but that of the phenomenologist performing the study. In our case we assume that the investigation is to extract information stemming from measured data. Many applications can be very complex, for instance, in the case of waves propagating through various media such as below the surface of the earth [2] or through tissue in biomedical [3] or through heterogeneous materials of critical parts in nondestructive evaluation (NDE) investigations [4]. In any case, the processing typically involves manipulating the measured data to extract the desired information, such as location of an epicenter, or the detection of a tumor or flaw in both biomedical and NDE applications.

Another view of the underlying processing problem is to decompose it into a set of steps that capture the strategic essence of the processing scheme. Inherently, we believe that the more "a priori" knowledge about the measurement and its underlying phenomenology we can incorporate into the processor, the better we can expect the processor to perform—as long as the information that is included is correct! One strategy called the *model-based approach* provides the essence of model-based

*Model-Based Signal Processing*, by James V. Candy

signal processing [1]. Some believe that all signal processing schemes can be cast into this generic framework. Simply, the model-based approach is "incorporating mathematical models of both physical phenomenology and the measurement process (including noise) into the processor to extract the desired information." This approach provides a mechanism to incorporate knowledge of the underlying physics or dynamics in the form of mathematical process models along with measurement system models and accompanying noise as well as model uncertainties directly into the resulting processor. In this way the model-based processor enables the interpretation of results directly in terms of the problem physics. The model-based processor is actually a modeler's tool enabling the incorporation of any a priori information about the problem to extract the desired information. As depicted in Figure 1.1, the fidelity of the model incorporated into the processor determines the complexity of the model-based processor with the ultimate goal of increasing the inherent signal-to-noise ratio (*SNR*). These models can range from simple, implicit, nonphysical representations of the measurement data such as the Fourier or wavelet transforms to parametric black-box models used for data prediction, to lumped mathematical representations characterized by ordinary differential equations, to distributed representations characterized by partial differential equation models to capture the underlying physics of the process under investigation. The dominating factor of which model is the most appropriate is usually determined by how severe the measurements are contaminated with noise and the underlying uncertainties. If the *SNR* of the measurements is high, then simple nonphysical techniques can be used to extract the desired information; however, for low *SNR* measurements more and more of the physics and instrumentation must be incorporated for the extraction.

This approach of selecting the appropriate processing technique is pictorially shown in Figure 1.1. Here we note that as we progress up the *modeling steps*

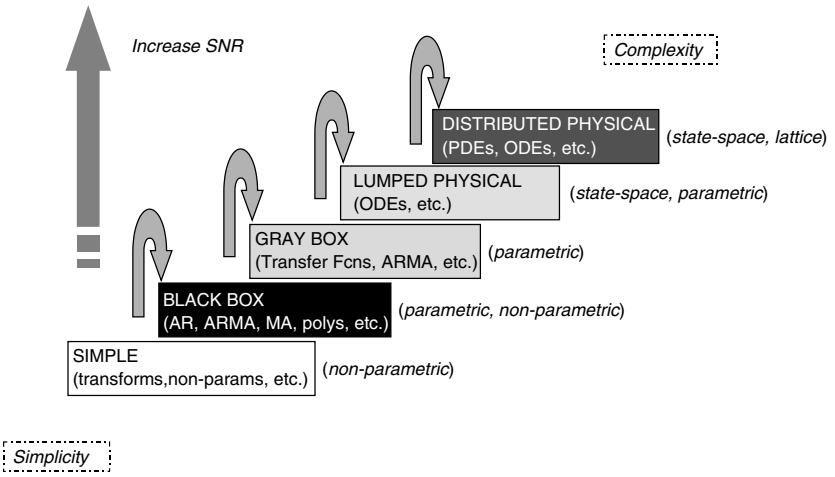



**Figure 1.1.** Model-based signal processing: model staircase.

to increase *SNR*, the model and algorithm complexity increases proportionally to achieve the desired results. Examining each of the steps individually leads us to realize that the lowest step involving no explicit model (*simple*) essentially incorporates little a priori information; it is used to analyze the information content (spectrum, time-frequency, etc.) of the raw measurement data to attempt to draw some rough conclusions about the nature of the signals under investigation. Examples of these simple techniques include Fourier transforms and wavelet transforms, among others. Progressing up to the next step, we have *black-box* models that are basically used as data prediction mechanisms. They have a parametric form (polynomial, transfer function, etc.), but again there is little physical information that can be gleaned from their outputs. At the next step, the *gray-box* models evolve that can use the underlying black-box structures; however, now the parameters can be used extract limited physical information from the data. For instance, a black-box transfer function model fit to the data yields coefficient polynomials that can be factored to extract resonance frequencies and damping coefficients characterizing the overall system response being measured. Progressing farther up the steps, we finally reach the true model-based techniques that explicitly incorporate the process physics using a *lumped physical* model structure usually characterized by ordinary differential or difference equations. The top step leads us to processes that are captured by *distributed physical* model structures in the form of partial differential equations. This level is clearly the most complex, since much computer power is devoted to solving the physical propagation problem. So we see that model-based signal processing offers the ability to operate directly in the physical space of the phenomenologist with the additional convenience of providing a one-to-one correspondence between the underlying phenomenology and the model embedded in the processor. This text is concerned with the various types of model-based processors that can be developed based on the model set selected to capture the physics of the measured data and the inherent computational algorithms that evolve. Before we proceed any further, let us consider a simple example to understand these concepts and their relationship to the techniques that evolve. In the subsequent section we will go even further to demonstrate how an explicit model-based solution can be applied to solve a wave-type processing problem.

However, before we begin our journey, let us look at a relatively simple example to motivate the idea of incorporating a model into a signal processing scheme. Suppose that we have a measurement of a sinusoid at 10 Hz in random noise and we would like to extract this sinusoid (the information) as shown in Figure 1.2*a*. The data are noisy as characterized by the dotted line with the true deterministic sinusoidal signal (solid line) overlayed in the figure. Our first attempt to analyze the raw measurement data is to take its Fourier transform (implicit sinusoidal model) and examine the resulting frequency bands for spectral content. The result is shown in Figure 1.2*b*, where we basically observe a noisy spectrum and a set of potential resonances—but nothing absolutely conclusive. After recognizing that the data are noisy, we apply a classical power spectral estimator using an inherent black-box model (implicit all-zero transfer function or polynomial model) with the resulting spectrum shown in Figure 1.2*c*. Here we note that the resonances have clearly

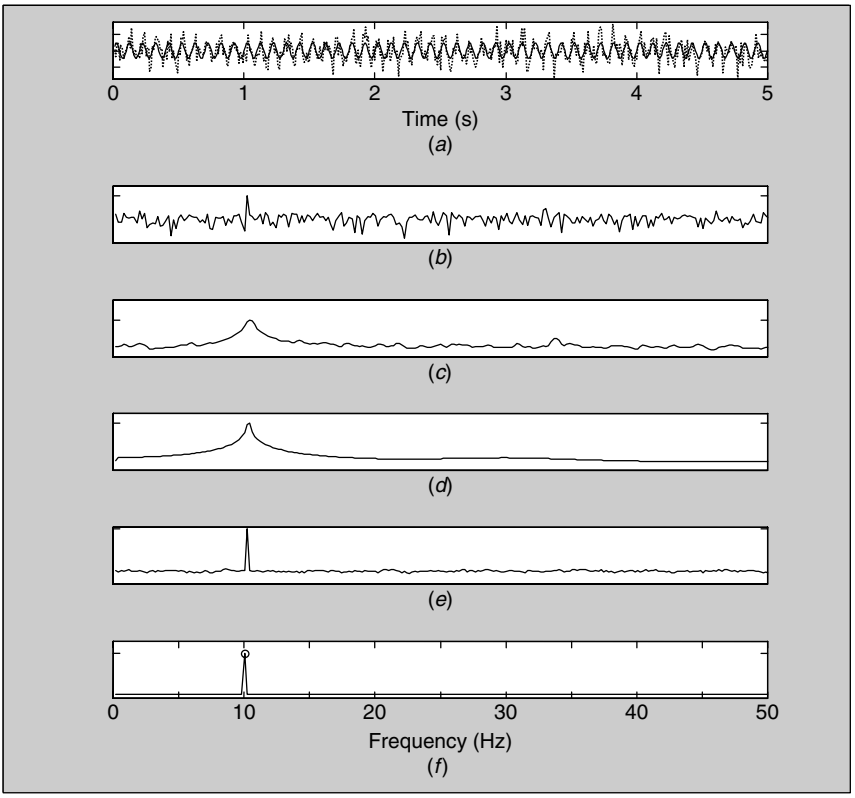



**Figure 1.2.** Sinusoid in noise example. (*a*) Noisy data and 10 Hz sinusoid. (*b*) Fourier spectrum. (*c*) Black-box power spectrum. (*d*) Gray-box polynomial power spectrum. (*e*) Gray-box harmonic power spectrum. (*f*) Model-based power spectrum.

been enhanced (smoothed) and the noise spectra attenuated by the processor, but their still remains a significant amount of uncertainty in the spectrum. However, observing the resonances in the power spectrum, we proceed next to a gray-box model explicitly incorporating a polynomial model that can be solved to extract the resonances (roots) even further, as shown in Figure 1.2*d*. Next, we use this extracted model to develop an explicit model-based processor (*MBP*). At this point we *know* from the analysis that the problem is a sinusoid in noise, and we design a processor incorporating this a priori information. Noting the sharpness of the peak, we may incorporate a harmonic model that captures the sinusoidal nature of the resonance but does not explicitly capture the uncertainties created by the measurement instrumentation and noise into the processing scheme. The results are shown in Figure 1.2*e*. Finally, we develop a set of harmonic equations for a sinusoid (ordinary differential equations) in noise as well as the measurement instrumentation model and noise statistics to construct a *MBP*. The results are shown in Figure 1.2*f*, clearly demonstrating the superiority of the model-based approach,

when the embedded models are correct. The point of this example is to demonstrate that progressing up the *steps* incorporating more and more sophisticated models, we can enhance the SNR and extract the desired information.

## 1.2 SIGNAL ESTIMATION

If a measured signal is free from extraneous variations and is repeatable from measurement to measurement, then it is defined as a *deterministic signal*. However, if it varies extraneously and is no longer repeatable, then it is a *random signal*. This section is concerned with the development of processing techniques to extract pertinent information from random signals utilizing any a priori information available. We call these techniques *signal estimation* or *signal enhancement*, and we call a particular algorithm a *signal estimator* or just *estimator*. Symbolically, we use the caret (ˆ) notation throughout this text to annotate an estimate (e.g., $s \rightarrow \hat{s}$). Sometimes estimators are called filters (e.g., Wiener filter) because they perform the same function as a deterministic (signal) filter except for the fact that the signals are random; that is, they remove unwanted disturbances. Noisy measurements are processed by the estimator to produce "filtered" data.

Estimation can be thought of as a procedure made up of three primary parts:

- Criterion function
- Models
- Algorithm

The criterion function can take many forms and can also be classified as deterministic or stochastic. Models represent a broad class of information, formalizing the a priori knowledge about the process generating the signal, measurement instrumentation, noise characterization, underlying probabilistic structure, and so forth as discussed in the previous section. Finally, the algorithm or technique chosen to minimize (or maximize) the criterion can take many different forms depending on (1) the models, (2) the criterion, and (3) the choice of solution. For example, one may choose to solve the well-known least-squares problem recursively or with a numerical-optimization algorithm. Another important aspect of most estimation algorithms is that they provide a "measure of quality" of the estimator. Usually what this means is that the estimator also predicts vital statistical information about how well it is performing.

Intuitively, we can think of the estimation procedure as the

- Specification of a criterion
- Selection of models from a priori knowledge
- Development and implementation of an algorithm

Criterion functions are usually selected on the basis of information that is meaningful about the process or the ease with which an estimator can be developed.

Criterion functions that are useful in estimation can be classified as deterministic and probabilistic. Some typical functions are as follows:

- *Deterministic*
  Squared error
  Absolute error
  Integral absolute error
  Integral squared error
- *Probabilistic*
  Maximum likelihood
  Maximum a posteriori (Bayesian)
  Maximum entropy
  Minimum (error) variance

Models can also be deterministic as well as probabilistic; however, here we prefer to limit their basis to knowledge of the process phenomenology (physics) and the underlying probability density functions as well as the necessary statistics to describe the functions. Phenomenological models fall into the usual classes defined by the type of underlying mathematical equations and their structure, namely linear or nonlinear, differential or difference, ordinary or partial, time invariant or varying. Usually these models evolve to a stochastic model by the inclusion of uncertainty or noise processes.

Finally, the estimation algorithm can evolve from various influences. A preconceived notion of the structure of the estimator heavily influences the resulting algorithm. We may choose, based on computational considerations, to calculate an estimate recursively rather than as a result of a batch process because we require an online, pseudo-real-time estimate. Also each algorithm must provide a measure of estimation quality, usually in terms of the expected estimation error. This measure provides a means for comparing estimators. Thus the estimation procedure is a combination of these three major ingredients: criterion, models, and algorithm. The development of a particular algorithm is an interaction of selecting the appropriate criterion and models, as depicted in Figure 1.3.

Conceptually, this completes the discussion of the general estimation procedure. Many estimation techniques have been developed independently from various viewpoints (optimization, probabilistic) and have been shown to be equivalent. In most cases, it is easy to show that they can be formulated in this framework. Perhaps it is more appropriate to call the processing discussed in this chapter "model based" to differentiate it somewhat from pure statistical techniques. There are many different forms of model-based processors depending on the models used and the manner in which the estimates are calculated. For example, there are process model-based processors (Kalman filters [5], [6], [7]), statistical model-based processors (Box-Jenkins filters [8], Bayesian filters [1], [9]), statistical model-based processors (covariance filters [10]), or even optimization-based processors (gradient filters [11], [12]).

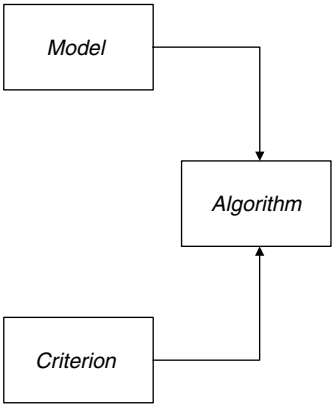



**Figure 1.3.** Interaction of model and criterion in an estimation algorithm.

This completes the introductory discussion on model-based signal processing from a heuristic point of view. Next we examine these basic ideas in more detail in the following sections.

## 1.3 MODEL-BASED PROCESSING EXAMPLE

In this section we develop a simple, yet more meaningful example to solidify these concepts and give a more detailed view of how model-based signal processing is developed using implicit and explicit model sets. The example we discuss is the passive localization of a source or target with its associated processing. This problem occurs in a variety of applications such as the seismic localization of an earthquake using an array of seismometers [2], the passive localization of a target in ocean acoustics [13], and the localization of a flaw in NDE [4]. Here we simulate the target localization problem using typical oceanic parameters, but we could have just as easily selected the problem parameters to synthesize any of the other applications mentioned.

For our problem in ocean acoustics the model-based approach is depicted in Figure 1.4. The underlying physics is represented by an acoustic propagation (process) model depicting how the sound propagates from a source or target to the sensor (measurement) array of hydrophones. Noise in the form of the background or ambient noise, shipping noise, uncertainty in the model parameters, and so forth is shown in the figure as input to both the process and measurement system models. Besides the model parameters and initial conditions, the raw measurement data is input to the processor with the output being the filtered signal and unknown parameters.

Assume that a 50 Hz plane wave source (target) at a bearing angle of $45^\circ$ is impinging on a two-element array at a 10 dB *SNR*. The plane wave signal can be characterized mathematically by

$$s_\ell(t) = \alpha e^{j\kappa_o(\ell-1)\triangle \sin\theta_o - j\omega_o t} \tag{1.1}$$

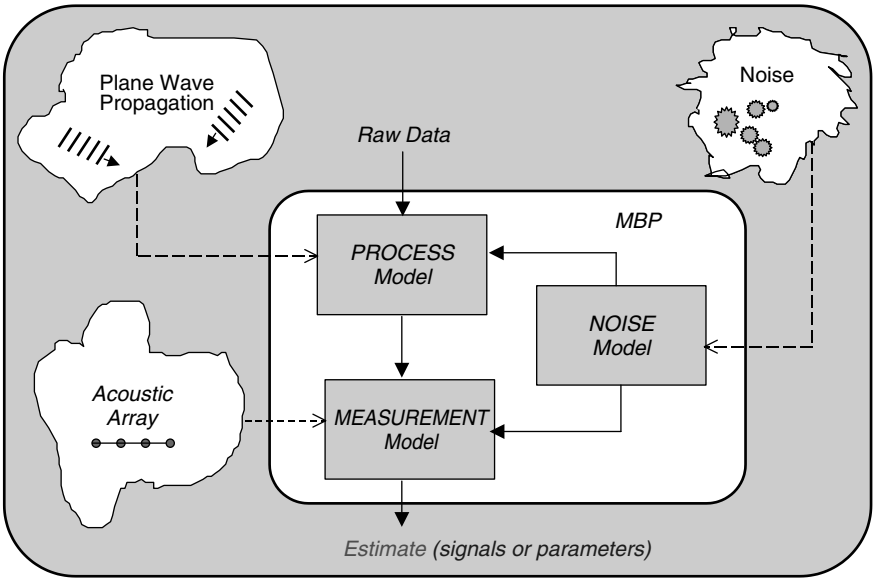



**Figure 1.4.** Model-based approach: Structure of the model-based processor showing the incorporation of propagation (ocean), measurement (sensor array), and noise (ambient) models.

where $s_\ell(t)$ is the space-time signal measured by the $\ell$th sensor, $\alpha$ is the plane wave amplitude factor with $\kappa_o, \triangle, \theta_o, \omega_o$ the respective wavenumber, sensor spacing, bearing angle, and temporal frequency parameters. We would like to solve two basic ocean acoustic processing problems: (1) signal enhancement, and (2) extraction of the source bearing angle, $\theta_o$, and temporal frequency, $\omega_o$ parameters. The basic problem geometry and synthesized measurements are shown in Figure 1.5.

First, we start with the *signal enhancement problem*, which is as follows:

**GIVEN** a set of noisy array measurements $\{p_\ell(t)\}$. **FIND** the best estimate of the signal, $s_\ell(t)$, that is, $\hat{s}_\ell(t)$.

This problem can be solved classically [13] by constructing a 50 Hz bandpass filter with a narrow 1 Hz bandwidth and filtering each channel. The model-based approach would be to define the various models as described in Figure 1.4 and incorporate them into the processor structure. For the *plane wave* enhancement problem we have the following models:

- *Signal model:*

$$s_\ell(t) = \alpha e^{j\kappa_o(\ell-1)\triangle \sin\theta_o - j\omega_o t}$$

- *Measurement:*

$$p_\ell(t) = s_\ell(t) + n_\ell(t)$$

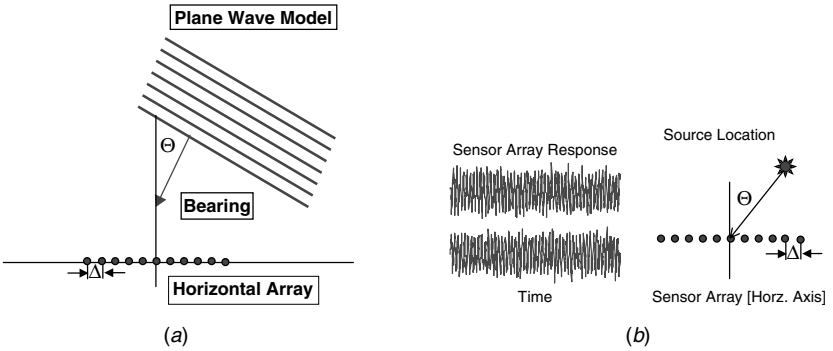



**Figure 1.5.** Plane wave propagation: (*a*) Problem geometry. (*b*) Synthesized 50 Hz, 45°, plane wave impinging on a two-element sensor array at 10 dB SNR.

- *Noise model:*

$$n \sim \mathcal{N}(0, \sigma_n^2)$$

where $n_\ell(t)$ is zero mean, random (uncorrelated) gaussian[1] noise with variance, $\sigma_n^2$.

The results of the classical and *MBP* outputs are shown in Figure 1.6. In Figure 1.6*a* the classical bandpass filter design is by no means optimal as noted by the random amplitude fluctuations created by the additive measurement noise process discussed above. The output of the optimal *MBP* is, in fact, optimal for this problem, since it embeds the correct process (plane wave), measurement (hydrophone array), and noise statistic (white, gaussian) models into its internal structure. We observe the optimal outputs in Figure 1.6*b*.

Summarizing this example, we see that the classical processor design is based on the a priori knowledge of the desired signal frequency (50 Hz) but does not incorporate knowledge of the process physics or noise into its design explicitly. On the other hand, the *MBP uses* the a priori information about the plane wave propagation signal and sensor array measurements along with any a priori knowledge of the noise statistics in the form of mathematical models embedded into its processing scheme to achieve optimal performance. The next variant of this problem is even more compelling.

Consider now the same plane wave, the same noisy hydrophone measurements with a more realistic objective of estimating the bearing angle and temporal frequency of the target. In essence, this is a problem of estimating a set of parameters, $\{\theta_o, \omega_o\}$ from noisy array measurements, $\{p_\ell(t)\}$. More formally, the *source-bearing angle and frequency estimation problem* is stated as follows:

> **GIVEN** a set of noisy array measurements $\{p_\ell(t)\}$. **FIND** the best estimates of the plane wave bearing angle ($\theta_o$) and temporal frequency ($\omega_o$) parameters, $\hat{\theta}_o$ and $\hat{\omega}_o$.

[1]We use the notation, "$\mathcal{N}(m, v)$" to define a gaussian or normal probability distribution with mean, $m$, and variance, $v$.

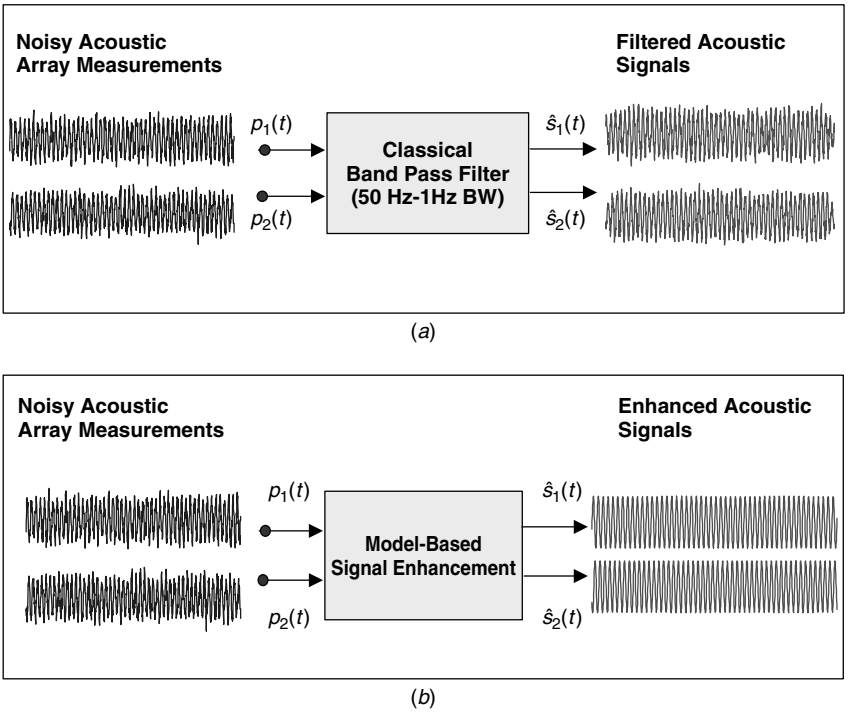



**Figure 1.6.** Plane wave—signal enhancement problem: (*a*) Classical bandpass filter (50 Hz, 1 Hz BW) approach. (*b*) Model-based processor using 50 Hz, 45°, plane wave model impinging on a two-element sensor array.

The classical approach to this problem is to first take one of the sensor channels and perform spectral analysis on the filtered time series to estimate the temporal frequency, $\omega_o$. The bearing angle can be estimated independently by performing classical beamforming [14] on the array data. A beamformer can be considered a spatial spectral estimator that is scanned over bearing angle indicating the true source location at maximum power. The results of applying this approach to our problem are shown in Figure 1.7*a* depicting the outputs of both spectral estimators peaking at the correct frequency and angle parameters.

The *MBP* is implemented as before by incorporating the plane wave propagation, hydrophone array, and noise statistical models; however, the temporal frequency and bearing angle parameters are now unknown and must be estimated along with simultaneous enhancement of the signals. The solution to this problem is performed by "augmenting" the unknown parameters into the *MBP* structure and solving the joint estimation problem [1], [9]. This is the *parameter adaptive* form of the *MBP* used in many applications [15], [16]. Here the problem becomes nonlinear due to the augmentation and is more computationally intensive; however, the results are appealing as shown in Figure 1.7*b*. We see the bearing angle and temporal frequency estimates as a function of time eventually converging to the true values

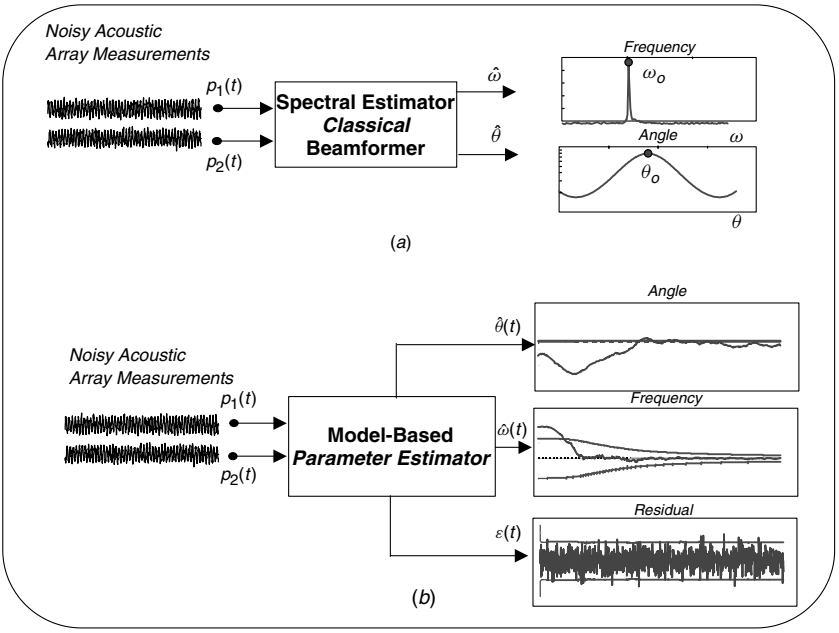



**Figure 1.7.** Plane wave impinging on a two-element sensor array–frequency and bearing estimation problem: (*a*) Classical spectral (temporal and spatial) estimation approach. (*b*) Model-based approach using parametric adaptive (nonlinear) processor to estimate bearing angle, temporal frequency, and the corresponding residual or innovations sequence.

($\omega_o = 50\,\text{Hz}$, $\theta_o = 45^\circ$). The *MBP* also produces a "residual sequence" (shown in the figure) that is used to determine its performance.

We summarize the classical and model-based solutions to the temporal frequency and bearing angle estimation problem. The classical approach simply performs spectral analysis temporally and spatially (beamforming) to extract the parameters from noisy data, while the model-based approach embeds the unknown parameters into its propagation, measurement, and noise models through augmentation enabling a solution to the joint estimation problem. The *MBP* also monitors its performance by analyzing the statistics of its residual (or innovations) sequence. It is this sequence that indicates the optimality of the *MBP* outputs. This completes the example, next we begin a more formal discussion to model-based signal processing.

## 1.4 MODEL-BASED SIGNAL PROCESSING CONCEPTS

In this section we introduce the concepts of model-based signal processing. We discuss a procedure for processor design and then develop the concepts behind model-based processing.

We also discuss in more detail the levels of models that can be used for processing purposes. It is important to investigate what, if anything, can be gained

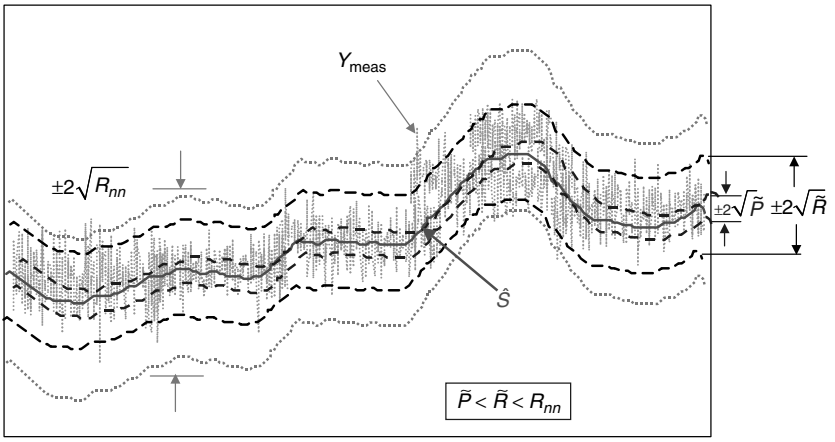



**Figure 1.8.** Model-based processing of a noisy measurement demonstrating conceptually the effect of incorporating more and more a priori information into the processor and decreasing the error variance.

by filtering the data. The amount of information available in the data is related to the precision (variance) of the particular measurement instrumentation used as well as any signal processing employed to enhance the outputs. As we utilize more and more information about the physical principles underlying the given data, we expect to improve our estimates (decrease estimation error) significantly.

A typical measurement $y$ is depicted in Figure 1.8. In the figure we see the noisy measurement data and corresponding "final" estimates of the true signal bounded by its corresponding confidence intervals [9]. As we develop more and more sophisticated processors by including a priori knowledge into the algorithm, the uncertainty in the estimate decreases further as shown by the dashed bounds. Mathematically we illustrate these ideas by the following.

In Figure 1.9*a*, the ideal measurement instrument is given by

$$Y_{\text{meas}} = S_{\text{true}} + N_{\text{noise}} \qquad \text{(Ideal measurement)}$$

A more realistic model of the measurement process is depicted in Figure 1.9*b*.

If we were to use $Y$ to estimate $S_{\text{true}}$ (i.e., $\hat{S}$), we have the noise lying within the $\pm 2\sqrt{R_{nn}}$ confidence limits superimposed on the signal (see Figure 1.8). The best estimate of $S_{\text{true}}$ we can hope for is only within the accuracy (bias) and precision (variance) of the instrument. If we include a model of the measurement instrument (see Figure 1.9*b*) as well as its associated uncertainties, then we can improve the SNR of the noisy measurements. This technique represents the processing based on our instrument and noise (statistical) models given by

$$Y_{\text{meas}} = C(S_{\text{true}}) + N_{\text{noise}}, \quad N \sim \mathcal{N}(0, R_{nn}) \qquad \text{(Measurement and noise)}$$

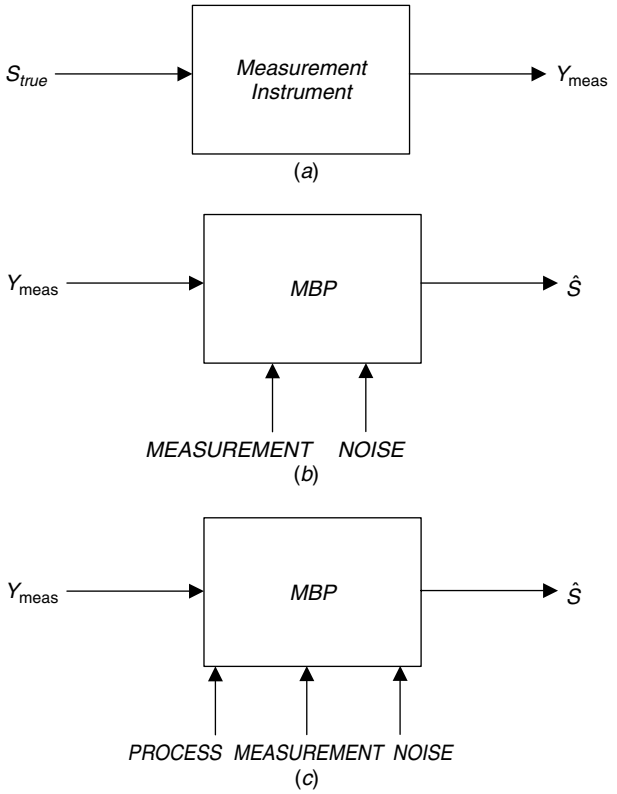



**Figure 1.9.** Classes of model-based signal processors: (*a*) Simple measurement system. (*b*) Model-based signal processing with measurement instrument modeling. (*c*) Model-based signal processing with measurement uncertainty and process modeling.

where $C(S_{\text{true}})$ is the measurement system model and $\mathcal{N}(0, R_{nn})$ is the noise statistics captured by a *gaussian* or *normal distribution* of zero-mean and variance specified by $R_{nn}$.

We can also specify the estimation error variance $\tilde{R}$ or equivalently the quality of this processor in estimating $S$. Finally, if we incorporate not only instrumentation knowledge, but also knowledge of the physical process (see Figure 1.9*c*), then we expect to do even better, that is, we expect the estimation error ($\tilde{S} = S_{\text{true}} - \hat{S}$) variance $\tilde{P}$ to be small (see Figure 1.8 for $\pm 2\sqrt{\tilde{P}}$ confidence limits or bounds) as we incorporate more and more knowledge into the processor, namely

$$\hat{S} = A(S_{\text{true}}) + W_{\text{noise}}, \quad W \sim \mathcal{N}(0, R_{ww}) \qquad \text{(Process model and noise)}$$

$$Y_{\text{meas}} = C(S_{\text{true}}) + N_{\text{noise}}, \quad N \sim \mathcal{N}(0, R_{nn}) \qquad \text{( Measurement model and noise)}$$

where $A(S_{\text{true}})$ is the process system model and $\mathcal{N}(0, R_{ww})$ is the process noise statistics captured by a zero-mean, gaussian distribution with variance, $R_{ww}$.

It can be shown that this is the case because

$$\tilde{P} < \tilde{R} < R_{nn}\,(\text{instrument variance})$$

This is the basic idea in model-based signal processing: "The more a priori information we can incorporate into the algorithm, the smaller the resulting error variance." The following example illustrates these ideas:

***Example 1.1*** The voltage at the output of an *RC* circuit is to be measured using a high-impedance voltmeter shown in Figure 1.10. The measurement is contaminated with random instrumentation noise that can be modeled by

$$e_{\text{out}} = K_e e + v$$

where

$$\begin{aligned} e_{\text{out}} &= \text{measured voltage} \\ K_e &= \text{instrument amplification factor} \\ e &= \text{true voltage} \\ v &= \text{zero-mean random noise of variance, } R_{vv} \end{aligned}$$

This model corresponds to that described in Figure 1.9*b*. A processor is to be designed to improve the precision of the instrument. Then, as in the figure, we have the measurement

$$\begin{aligned} S &\rightarrow e \\ Y &\rightarrow e_{\text{out}} \\ C(\cdot) &\rightarrow K_e \\ N &\rightarrow v \\ R_{nn} &\rightarrow R_{vv} \end{aligned}$$

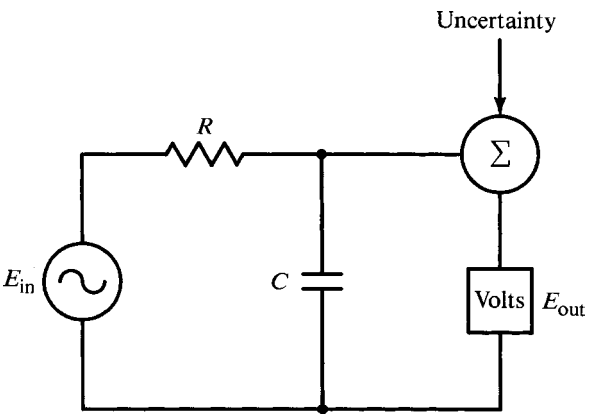



**Figure 1.10.** Model-based processing of an *RC* circuit.

For the filter we have

$$\hat{s} \rightarrow \hat{e}$$
$$\tilde{R}_{ss} \rightarrow \tilde{R}_{ee}$$

The precision of the instrument can be improved even further by including a model of the process (circuit). Writing the Kirchoff node equations, we have

$$\dot{e} = \frac{1}{C} I_{\text{in}} - \frac{e}{RC} + q$$

where

$$R = \text{resistance}$$
$$C = \text{capacitance}$$
$$I_{in} = \text{excitation current}$$
$$q = \text{zero-mean, random noise of variance } R_{qq}$$

The improved model-based processor employs both measurement and process models as in Figure 1.9*c*. Thus we have

- Process

$$\dot{s} \rightarrow \dot{e}$$
$$A(\cdot) \rightarrow -\frac{e}{RC} + \frac{1}{C} I_{\text{in}}$$
$$W \rightarrow q$$
$$R_{ww} \rightarrow R_{qq}$$

- Measurement

$$S \rightarrow e$$
$$Y \rightarrow e_{\text{out}}$$
$$C(\cdot) \rightarrow K_e$$
$$N \rightarrow v$$
$$R_{nn} \rightarrow R_{vv}$$

Therefore the filter becomes

$$\hat{s} \rightarrow \hat{e}$$
$$\tilde{P}_{ss} \rightarrow \tilde{P}_{ee}$$

such that

$$\tilde{P}_{ee} < \tilde{R}_{ee} < R_{vv}$$

This completes the example.

## 1.5 NOTATION AND TERMINOLOGY

The notation used throughout this text is standard in the literature. Vectors are usually represented by boldface, lowercase, $\mathbf{x}$ or $\underline{x}$, and matrices by boldface, uppercase, $\mathbf{A}$. We denote the real part of a signal by $Re\ x$ and its imaginary part by $Im\ x$. We define the notation $\underline{N}$ to be a shorthand way of writing $1, 2, \ldots, N$. It will be used in matrices, $A(\underline{N})$ to mean there are $N$-columns of $A$. As mentioned previously, estimators are annotated by the caret, $\hat{x}$. We also define partial derivatives at the component level by $\partial/\partial\theta_i$, the $N_\theta$ gradient vector by $\nabla_\theta$ and higher order partials by $\nabla^2_\theta$. Inner product is $<\mathbf{a}, \mathbf{b}> := \mathbf{a}'\mathbf{b}$ and $|\mathbf{a}'\mathbf{b}|^2 := \mathbf{b}'\mathbf{a}\mathbf{a}'\mathbf{b}$. The conjugate is $a^*$ and convolution is $a * b$.

The most difficult notational problem will be with the "time" indexes. Since this text is predominantly on discrete time, we will use the usual time symbol, $t$ to mean a discrete-time index (i.e., $t \in \mathcal{I}$ for $\mathcal{I}$ the set of integers). However, and hopefully not too confusing, $t$ will also be used for continuous time (i.e., $t \in \mathcal{R}$ for $\mathcal{R}$ the set of real numbers denoting the continuum). When used as a continuous-time variable, $t \in \mathcal{R}$ it will be represented as a *subscript* to distinguish it (i.e., $x_{\mathbf{t}}$). This approach of choosing $t \in \mathcal{I}$ primarily follows the system identification literature and for the ease of recognizing discrete-time variable in transform relations (e.g., discrete Fourier transform). The rule-of-thumb is therefore to "interpret $t$ as a discrete-time index unless noted by a subscript as continuous in the text." With this in mind we will define a variety of discrete estimator notations as $\hat{x}(t|t-1)$ to mean the estimate at time (discrete) $t$ based on all of the previous data up to $t-1$. We will define these symbols prior to their use with the text to ensure that no misunderstandings arise.

Also we will use the symbol $\sim$ to mean "distributed according to" as in $x \sim \mathcal{N}(m, v)$ defining the random variable $x$ as gaussian distributed with mean $m$ and variance $v$.

## 1.6 SUMMARY

In this chapter we introduced the concept of model-based signal processing based on the idea of incorporating more and more available a priori information into the processing scheme. Various signals were classified as deterministic or random. When the signal is random, the resulting processors are called estimation filters or estimators. A procedure was defined (estimation procedure) leading to a formal decomposition that will be used throughout this text. After a couple of examples motivating the concept of model-based signal processing, a more formal discussion followed with an $RC$-circuit example, completing the chapter.

## *MATLAB*® NOTES

*MATLAB*® is command-oriented vector-matrix package with a simple yet effective command language featuring a wide variety of embedded $C$ language constructs making it ideal for signal processing applications and graphics. All of the algorithms

we have applied to the examples and problems in this text are *MATLAB*-based in solution ranging from simple simulations to complex applications. We will develop these notes primarily as a summary to point out to the reader many of the existing commands that already perform the signal-processing operations discussed in the presented chapter and throughout the text.

## REFERENCES


1. J. V. Candy, *Signal Processing: The Model-Based Approach*, New York: McGraw-Hill, 1986.
2. E. Robinson and S. Treitel, *Geophysical Signal Analysis*, Englewood Cliffs, NJ: Prentice-Hall, 1980.
3. A. C. Kak and M. Slaney, *Principles of Computerized Tomographic Imaging*, New York: IEEE Press, 1988.
4. K. Langenburg, "Applied inverse problems for acoustic, electromagnetic and elastic wave scattering," in P. C. Sabatier, ed., *Basic Methods of Tomography and Inverse Problems*, Philadelphia: Hilger, 1987.
5. M. S. Grewal and A. P. Andrews, *Kalman Filtering: Theory and Practice*, Englewood Cliffs, NJ: Prentice-Hall, 1993.
6. J. M. Mendel, *Lessons in Estimation Theory for Signal Processing, Communications, and Control*, Englewood Cliffs, NJ: Prentice-Hall, 1995.
7. R. G. Brown and P. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*, New York: Wiley, 1997.
8. G. E. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*, San Francisco: Holden-Day, 1976.
9. A. Jazwinski, *Stochastic Processes and Filtering Theory*, New York: Academic Press, 1970.
10. G. J. Bierman, *Factorization Methods for Discrete Sequential Estimation*, New York: Academic Press, 1977.
11. G. C. Goodwin and R. L. Payne, *Dynamic System Identification*, New York: Academic Press, 1976.
12. L. J. Ljung, *System Identification: Theory for the User*, Englewood Cliffs, NJ: Prentice-Hall, 1987.
13. R. Nielson, *Sonar Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1995.
14. D. Johnson and R. Mersereau, *Array Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1993.
15. H. Sorenson, Editor, *Kalman Filtering: Theory and Application*, New York: IEEE Press, 1985.
16. J. V. Candy and E. J. Sullivan, "Ocean acoustic signal processing: A model-based approach," *J. Acoust. Soc. Am.*, **92** (12), 3185–3201, 1992.


## PROBLEMS

**1.1** We are asked to estimate the displacement of large vehicles (semi-trailers) when parked on the shoulder of a freeway and subjected to wind gusts created

by passing vehicles. We measure the displacement of the vehicle by placing an accelerometer on the trailer. The accelerometer has inherent inaccuracies which is modeled as

$$y = K_a x + n$$

with $y$, $x$, and $n$, the measured and actual displacement and white measurement noise of variance, $R_{nn}$ and $K_a$ the instrument gain. The dynamics of the vehicle can be modeled by a simple mass-spring-damper.

**(a)** Construct and identify the measurement model of this system.

**(b)** Construct and identify the process model and model-based estimator for this problem.

**1.2** Think of measuring the temperature of a liquid in a breaker heated by a burner. Suppose that we use a thermometer immersed in the liquid and periodically observe the temperature and record it.

**(a)** Construct a measurement model assuming that the thermometer is linearly related to the temperature, that is, $y(t) = k\triangle T(t)$. Also model the uncertainty of the visual measurement as a random sequence $v(t)$ with variance $R_{vv}$.

**(b)** Suppose that we model the heat transferred to the liquid from the burner as

$$Q(t) = C\ A\ \triangle T(t)$$

where $C$ is the coefficient of thermal conductivity, $A$ is the cross-sectional area, and $\triangle T(t)$ is the temperature gradient with assumed random uncertainty $w(t)$ and variance $R_{ww}$. Using this process model and the models developed above, identify the model-based processor representation.

**1.3** We are given an $RLC$ series circuit (below) driven by a noisy voltage source $V_{in}(t)$, and we use a measurement instrument that linearly amplifies by $K$ and measures the corresponding output voltage. We know that the input voltage is contaminated by and additive noise source, $w(t)$ with covariance, $R_{ww}$ and the measured output voltage is similarly contaminated with noise source, $v(t)$ with $R_{vv}$.

**(a)** Determine the model for the measured output voltage, $V_{\text{out}}(t)$ (measurement model).

**(b)** Determine a model for the circuit (process model).

**(c)** Identify the general model-based processor structures depicted in Figure 1.10. In each scheme specify the models for the process, measurement and noise.

**1.4** A communications satellite is placed into orbit and must be maneuvered using thrusters to orientate its antennas. Restricting the problem to the single axis

perpendicular to the page, the equations of motion are

$$J\frac{d^2\theta}{dt^2} = T_c + T_d$$

where $J$ is the moment of inertia of the satellite about its center of mass, $T_c$ is the thruster control torque, $T_d$ is the disturbance torque, and $\theta$ is the angle of the satellite axis with respect to the inertial reference (no angular acceleration). Develop signal and noise models for this problem and identify each model-based processor component.

**1.5** Consider a process described by a set of linear differential equations

$$\frac{d^2c}{dt^2} + \frac{dc}{dt} + c = Km$$

The process is to be controlled by a proportional-integral-derivative (PID) control law governed by the equation

$$m = K_p\left(e + \frac{1}{T_i}\int edt + T_d\frac{de}{dt}\right)$$

and the controller reference signal $r$ is given by

$$r = e + c$$

Suppose that the reference is subjected to a disturbance signal and the measurement sensor, which is contaminated with additive noise, measures the "square" of the output. Develop the model-based signal and noise models for this problem as in Figure 1.10.

**1.6** The elevation of a tracking telescope is controlled by a dc motor. If the telescope has a moment of inertia $J$ and damping $B$ due to friction, the equation of motion is given by

$$J\frac{d^2\theta}{dt^2} + B\frac{d\theta}{dt} = T_m + T_d$$

where $T_m$ and $T_d$ are the motor and disturbance torques and $\theta$ is the elevation angle. Assume a sensor transforms the telescope elevation into a proportional voltage that is contaminated with noise. Develop the signal and noise models for the telescope, and identify all of the model-based processor components.

# 2

# DISCRETE RANDOM SIGNALS AND SYSTEMS

## 2.1 INTRODUCTION

We develop the concept of discrete random signals by employing a direct analogy from the deterministic (nonrandom) case. We classify signals as *deterministic*, if they are repeatable; that is, continued measurements reproduce the identical signal over and over again. However, when a signal is classified as *random*, it is no longer repeatable and therefore continued measurements produce different signals each time.

Recall that signal processing is concerned with extracting the useful information from a signal and discarding the extraneous. When a signal is classified as deterministic, then techniques from systems theory can be applied to extract the required signal information.

## 2.2 DETERMINISTIC SIGNALS AND SYSTEMS

A signal can be considered as an ordered set of values containing information about the process under investigation. Mathematically it can be represented by a scalar or vector function of one or more variables constituting the index set. Typically a scalar *discrete signal* is defined by the function

$$x(t) \quad \text{for } t \in \mathcal{I} \tag{2.1}$$

*Model-Based Signal Processing*, by James V. Candy

where $t$ is the discrete time index which is a member of the set of integers $\mathcal{I}$. *Continuous signals* are represented in similar manner by

$$x_t \qquad \text{for } t \in \mathcal{R} \tag{2.2}$$

where $t$ is the continuous-time index[1] that is a member of the set of real numbers $\mathcal{R}$.

Signals are classified according to some unique characteristic that can usually be represented mathematically. A signal is classified as *deterministic* if it is repeatable, that is, if we measure its value over a specified period of time and repeat the measurement some time later the signal has the same values. If a signal does not achieve the same values, then it is no longer repeatable and therefore considered *random*. A deterministic signal is defined as *periodic* with period $M$, if

$$x(t) = x(t + M) \qquad \forall t \in \mathcal{I} \tag{2.3}$$

A signal that is not periodic is called *aperiodic*. A discrete signal is called a *sampled* signal if it is the result of sampling a continuous time signal, that is, $x_{t_k}$ is a sampled signal if

$$x_s(t) := x_{t_k} \big|_{t_k = k\triangle T} \tag{2.4}$$

where $\triangle T$ is the specified sampling interval. Finally, a discrete (continuous) signal is defined as *random*, if it can be represented by the process,

$$x(t, \xi) \qquad t \in \mathcal{I}(\text{ or } \mathcal{R}), \xi \in \Xi \tag{2.5}$$

where $t$ is a member of the index sets $\mathcal{I}$ or $\mathcal{R}$ and $\xi$ is a member of the event space, $\Xi$. We will discuss random signals subsequently and concentrate on deterministic signals here.

Fundamental deterministic signals are useful for analysis. The *unit impulse* is defined by

$$\delta(t) := \begin{cases} 1, & t = 0 \\ 0, & t \neq 0 \end{cases} \tag{2.6}$$

The *unit step* is defined by

$$\mu(t) := \begin{cases} 0, & t < 0 \\ 1, & t \geq 0 \end{cases} \tag{2.7}$$

Two other basic discrete functions of interest are the *unit ramp* given by

$$x(t) = t \qquad \text{for } t > 0 \tag{2.8}$$

and the discrete *complex exponential* defined by

$$x(t) = \alpha^t \qquad \text{for } \alpha \in \mathbf{C} \tag{2.9}$$

[1]Recall that we characterize continuous signals by $t \in \mathcal{R}$ and represent it as a subscript (i.e. $x_{\mathbf{t}}$) to avoid confusion with the usual $t \in \mathcal{I}$.

We can think of $\alpha$ as

$$\alpha = e^{\Theta} \tag{2.10}$$

where $\Theta$ can be real or complex analogous to continuous time signals. If $\alpha$ is real, then $x$ is an increasing or decreasing exponential for $\alpha > 1$ or $\alpha < 1$, respectively. Note that with $\alpha < 1$, the sign of $x(t)$ alternates. For the case of $\Theta$ purely imaginary, we have the discrete exponential

$$x(t) = e^{j\Omega_k t} \tag{2.11}$$

for $\Omega_k$ the discrete frequency.

Similar to the continuous case, exponentials are related to sinusoidal signals of the form

$$x(t) = A\cos(\Omega_k t + \phi) \tag{2.12}$$

where $\Omega$ and $\phi$ are in units of radians per second and radians. It is interesting to note that the complex exponential and sinusoid are related through the *Euler* relations

$$e^{j\Omega_k t} = \cos\Omega_k t + j\sin\Omega_k t \tag{2.13}$$

and

$$A\cos(\Omega_k t + \phi) = \frac{A}{2}e^{j\phi}e^{j\Omega_k t} + \frac{A}{2}e^{-j\phi}e^{-j\Omega_k t} \tag{2.14}$$

We note in passing that unlike its continuous counterpart, the discrete exponential possesses unique properties. First, consider a complex exponential with frequency $\Omega_k + 2\pi$, then we have

$$e^{j(\Omega_k+2\pi)t} = e^{j2\pi t}e^{j\Omega_k t} = 1 \times e^{j\Omega_k t} = e^{j\Omega_k t} \tag{2.15}$$

which implies that the discrete exponential is *periodic* with period $2\pi$ unlike the continuous exponentials that are all distinct for distinct values of frequency. Thus discrete exponentials require only investigation in an interval of length $2\pi$. Note that it does *not* have a continually increasing rate of oscillation as $\Omega_k$ is increased in magnitude. Rather, oscillation rates increase from 0 until $\Omega_k = \pi$ and then decreases until $2\pi$. High frequencies are reached near $\pm\pi$.

Also from the periodicity property, we have that (see [1])

$$e^{j\Omega_k M} = 1 \tag{2.16}$$

or equivalently,

$$\Omega_k M = 2\pi k \tag{2.17}$$

or

$$\Omega_k = \frac{2\pi k}{M} \tag{2.18}$$

which implies that $\Omega_k$ is periodic only if $\Omega/2\pi$ or $k/M$ is a rational number. Here we define $M$ as the *fundamental period* of the complex exponential or sinusoid.

## 2.3 SPECTRAL REPRESENTATION OF DISCRETE SIGNALS

In this subsection we discuss various representations of discrete signals. First we consider various forms or models that exploit properties of the signal under investigation. From the signal processor's viewpoint, once we assume a particular class of signal, then the analysis that follows must be consistent with that assumption. For instance, if we assume that the signal under investigation is a transient, then representations of a signal that we assume to be periodic do not make any sense and should be discarded.

If we assume a signal is discrete and periodic, then we know that it can be represented by a Fourier series [1]. The *discrete Fourier series* (*DFS*) representation of a signal is given by the pair

$$x(t) = \sum_{m=0}^{M-1} a_m e^{j\Omega_m t} \tag{2.19}$$

and

$$a_m = \frac{1}{M} \sum_{t=0}^{M-1} x(t) e^{-j\Omega_m t} \tag{2.20}$$

where $\Omega_m = 2\pi m/M$. These equations are called the *synthesis* and *analysis* relations, respectively, and the $a_m$ are called the *discrete Fourier coefficients*. These relations imply that the discrete signal under investigation can be decomposed into a unique set of harmonics. Consider the following example to illustrate this representation.

***Example 2.1*** Assume that a signal under investigation is periodic, and we would like to investigate its properties. We model the periodic signal as

$$x(t) = \cos \Omega_k t$$

From the Euler relations we know that

$$\cos \Omega_k t = \frac{e^{j\Omega_k t} + e^{-j\Omega_k t}}{2}$$

which implies that the discrete Fourier series representation is given by

$$x(t) = \cos \Omega_k t = \frac{1}{2} e^{j\Omega_k t} + \frac{1}{2} e^{-j\Omega_k t}$$

or

$$a_m = \begin{cases} \frac{1}{2}, & m = \pm 1 \\ 0, & \text{elsewhere} \end{cases}$$

So we see that a periodic signal can be represented by a discrete harmonic series, and the corresponding coefficients can be considered an equivalent representation of the information in the given signal.

Analogous to continuous-time processes, we use transform representations as an alternative to analyze the information in a signal. For instance, if we believe a signal has periodic components, it may be possible to discern the individual periods from the signal; however, it is much more convenient to represent or transform the signal to the frequency domain where sinusoids appear (in theory) as spectral lines. Transformations to the frequency domain exist for discrete-time signals, just as they do for continuous-time signals. In the continuous domain we use the *Laplace transform* and its inverse to transform the signal to the frequency domain with $s = \sigma \pm j\omega$ as the complex variable. The *Laplace transform* and its inverse are defined by the pair

$$\begin{aligned} X(s) &:= \int_{-\infty}^{\infty} x_t e^{-jst}\, dt \\ x_t &= \int_{-\infty}^{\infty} X(s) e^{jst}\, ds \end{aligned} \tag{2.21}$$

where $x_t$ is the continuous-time signal. The *Laplace transform* provides nice convergence properties and possesses the continuous *Fourier transform*, $X(\omega)$, as a special case when the region of convergence is the $j\omega$-axis, that is, when $s = j\omega$.

In discrete time we have an analogous situation, the *Z-transform* plays the role of its continuous-time counterpart, the Laplace transform, and possesses similar convergence properties. It has the *discrete-time Fourier transform* as a special case when the complex variable $z$ is confined to the *unit circle*, that is, $z = e^{j\Omega}$. To be more precise, we define the *Z-transform* of a signal as

$$X(z) := \sum_{t=-\infty}^{\infty} x(t) z^{-t} \tag{2.22}$$

and the corresponding *inverse Z-transform* as

$$x(t) = \frac{1}{2\pi j} \oint X(z) z^{t-1}\, dz \tag{2.23}$$

We see that the direct transform of $x(t)$ is an infinite power series in $z^{-1}$ with coefficients $\{x(t)\}$, while the inverse transform is given by the contour integral that encircles the origin of the $Z$-plane. The basis of the $Z$-transform is the Laurent series of complex variable theory (see [2] for details). It converges for those values of $z$ that lie in the so-called *region of convergence* (*ROC*). Thus the $Z$-transform represents an analytic function at every point inside the *ROC*, and therefore the transform and all its derivatives must be continuous functions of $x(t)$ within the *ROC*. The $Z$-transform representation of a discrete signal enables us to analyze the spectral content of aperiodic signals and therefore determine such quantifiers

as bandwidth and spectral shape. From the signal processor's perspective, the $Z$-transform representation offers a convenient domain to analyze spectral properties of a discrete signal. Consider the following example.

***Example 2.2*** Consider the sinusoid of the previous example windowed by a unit step function, that is,

$$x(t) = \cos \Omega_k t \times \mu(t)$$

Then the corresponding $Z$-transform is given by

$$X(z) = Z\{\cos \Omega_k t \;\; \mu(t)\} = \sum_{t=0}^{\infty} \cos \Omega_k t \;\; z^{-t}$$

Using the Euler relations, however, we obtain

$$X(z) = \frac{1}{2} \sum_{t=0}^{\infty} e^{j\Omega_k t} z^{-t} + \frac{1}{2} \sum_{t=0}^{\infty} e^{-j\Omega_k t} z^{-t}$$

or

$$X(z) = \frac{1}{2} \sum_{t=0}^{\infty} (e^{j\Omega_k} z^{-1})^t + \frac{1}{2} \sum_{t=0}^{\infty} (e^{-j\Omega_k} z^{-1})^t$$

Using the relations for the geometric series,[2] we have

$$X(z) = \frac{1}{2} \frac{1}{1 - e^{j\Omega_k} z^{-1}} + \frac{1}{2} \frac{1}{1 - e^{-j\Omega_k} z^{-1}}$$

or combining under a common denominator and using the Euler relations, we obtain

$$X(z) = \frac{1 - \cos \Omega_k z^{-1}}{1 - 2 \cos \Omega_k z^{-1} + z^{-2}}$$

So we see that the $Z$-transform offers an alternate means of representing a discrete time aperiodic signal.

### 2.3.1 Discrete Systems

In this subsection we develop the concept of a discrete-time system analogous to that of a signal; that is, we define a *discrete-time system* as a transformation of a discrete-time input signal to a discrete-time output signal. Thus there is a clear distinction between a signal and a system, since the system basically operates on or transforms a signal. Constraining this transformation to be linear (homogeneity

[2]Recall that the *geometric series* is given by $\sum_{n=0}^{\infty} \alpha^n = 1/(1 - \alpha)$ for $|\alpha| < 1$.

and additivity) then leads to a *linear discrete-time system*. A discrete linear system is defined in terms of its (unit) impulse response, $h(\cdot)$, that is,

$$y(t) = T[x(t)] = \sum_k h(k,t)x(t) \tag{2.24}$$

With this in mind, we can now define further properties of discrete systems. If we constrain the system to be *time invariant*, then for any time shift of the input we must have a corresponding shift of the output, that is, $x(t-m) \longrightarrow y(t-m)$. So we see that a *linear time-invariant* (*LTI*) discrete system can be completely characterized by its impulse response as

$$y(t) = h(t) * x(t) := \sum_k h(k)x(t-k) = \sum_k x(k)h(t-k) \tag{2.25}$$

where $*$ is the convolution operator.

If we constrain the impulse response to be null for negative values of $k$, then the *LTI* system is said to be *causal*, that is,

$$y(t) = \sum_{k=0}^{\infty} h(k)x(t-k) \tag{2.26}$$

We also mention that the *LTI* is *stable* if for a bounded input, its output is bounded which requires that $\sum_t |h(t)| < \infty \ \forall t$ (see [1] for details).

Discrete *LTI* systems are powerful representations when coupled with the convolution relation of Eq. (2.25), enables us to determine the response of the system to *any* input given that we have its impulse response.

An important class of discrete *LTI* systems is that for which the input and output are related through linear constant coefficient difference equations. Just as in the continuous-time case where systems characterized by linear constant coefficient differential equations are important, this represents the discrete analogue. The differential/difference equation representation is extremely important in signal processing because through the sampling process most continuous systems can be modeled using difference equations on a digital computer. In fact these representations lead us to the modern model-based techniques of signal processing.

We define an $N_a$th order linear constant coefficient *difference equation* by

$$y(t) + a_1 y(t-1) + \cdots + a_{N_a} y(t-N_a) = b_1 x(t-1) + \cdots + b_{N_b} x(t-N_b)$$

or

$$\sum_{i=0}^{N_a} a_i y(t-i) = \sum_{i=1}^{N_b} b_i x(t-i) \qquad \text{for } N_b \le N_a,\ a_0 = 1 \tag{2.27}$$

Another way of writing this equation is in polynomial operator form. That is, if we define $q^{-k}$ the *backward shift operator* as

$$q^{-k} y(t) := y(t-k) \tag{2.28}$$

then we can write Eq. (2.27) as

$$A(q^{-1})y(t) = B(q^{-1})x(t) \tag{2.29}$$

where

$$\begin{aligned} A(q^{-1}) &= 1 + a_1 q^{-1} + \cdots + a_{N_a} q^{-N_a} \\ B(q^{-1}) &= b_1 q^{-1} + b_2 q^{-2} + \cdots + b_{N_b} q^{-N_b} \end{aligned} \tag{2.30}$$

A discrete system characterized by this form of the difference equation is called *recursive* because its present output ($y(t)$) is determined from past outputs ($\{y(t-i)\}$) and past inputs ($\{x(t-i)\}$). This representation can be extended to the random or stochastic case leading to the *ARMAX* model, which we discuss subsequently. This form is also called infinite impulse response (*IIR*) because the response consists of an infinite number of impulse response coefficients, which we will also discuss shortly. Similarly, when $A(q^{-1}) = 1$, Eq. (2.27) becomes

$$y(t) = B(q^{-1})x(t) = \sum_{i=1}^{N_b} b_i x(t-i) \tag{2.31}$$

which is called *nonrecursive* because its present output depends only on past inputs. If we identify this relation with Eq. (2.26), then we note that this is a finite sum of impulse-response weights ($h(i) \to b_i$). Thus this form is called finite impulse response (*FIR*) as well. We will call systems characterized by linear constant coefficient difference equations *parametric* and those characterized by impulse response weights *nonparametric*.

The $Z$-transform method enables difference equations to be solved through algebraic techniques much the same as the Laplace transform enables differential equations to be solved. An important class of $Z$-transforms are those for which $X(z)$ is a ratio of two polynomials, that is, a *rational function*. Recall that for $X(z)$ rational,

$$X(z) = \frac{B(z^{-1})}{A(z^{-1})} = \frac{\prod_{i=1}^{N_b}(z - z_i)}{\prod_{i=1}^{N_a}(z - p_i)}$$

then the *zeros* of $X(z)$ are those roots of the numerator polynomial for which $X(z)$ is null, that is,

$$\text{Zero:} \quad X(z)\big|_{z=z_i} = 0, \qquad i = 1, \cdots, N_b$$

Similarly the *poles* of $X(z)$ are the roots of the denominator polynomial for which $X(z)$ is infinite, that is,

$$\text{Pole:} \quad X(z)\big|_{z=p_i} = \infty, \qquad i = 1, \cdots, N_a$$

A discrete *LTI* is *stable* if and only if its poles lie within the unit circle. The $Z$-transform has many important properties that prove useful for solving difference equations. We also note that the $Z$-transform is linear, replaces convolution

of discrete signals with multiplication, and vice versa, and possesses a shifting property, that is,

$$Z[x(t-i)] = z^{-i}X(z) \tag{2.32}$$

which proves useful in solving difference equations.

The $Z$-transform can be used to replace a difference equation by polynomials in the complex variable $z$, which will eventually lead to algebraic solutions much like the Laplace transform is used to solve differential equations in continuous time. Using the time-shifting property, we see that the recursive difference equation

$$\sum_{i=0}^{N_a} a_i y(t-i) = \sum_{i=1}^{N_b} b_i x(t-i), \qquad a_0 = 1$$

can be represented as

$$(1 + a_1 z^{-1} + \cdots + a_{N_a} z^{-N_a})Y(z) = (b_1 z^{-1} + \cdots + b_{N_b} z^{-N_b})X(z)$$

by taking $Z$-transforms. We can also represent the system under investigation in rational form by

$$H(z) = \frac{Y(z)}{X(z)} = \frac{B(z^{-1})}{A(z^{-1})} = \frac{\sum_{i=1}^{N_b} b_i z^{-1}}{1 + \sum_{i=1}^{N_a} a_i z^{-1}} = \frac{\prod_{i=1}^{N_b}(z - z_i)}{\prod_{i=1}^{N_a}(z - p_i)}$$

where the last term is obtained by factoring the polynomials into poles and zeros.

We define the *transfer function* of the system as the rational function

$$H(z) := \left.\frac{Y(z)}{X(z)}\right|_{IC=0} = \frac{B(z^{-1})}{A(z^{-1})} = \sum_{t=0}^{\infty} h(t) z^{-t} \tag{2.33}$$

resulting from the transformation of the recursive difference equation with *zero initial conditions*. From this relation we now see more clearly why a rational system is termed infinite impulse response (*IIR*), since the division of numerator by denominator polynomials leads to a "infinite" set of impulse response coefficients. In fact this is a brute force method of calculating the inverse $Z$-transform [1]. Similarly an *FIR* system defined by

$$H(z) = B(z^{-1}) = z^{-N_b}(b_1 z^{N_b - 1} + \cdots + b_{N_b}) = \frac{\prod_{i=1}^{N_b}(z - z_i)}{z^{N_b}}$$

has $N_b$-poles at the origin and is therefore always stable (poles inside unit circle).

### 2.3.2 Frequency Response of Discrete Systems

If the *ROC* of the $Z$-transform includes the unit circle ($|z| = 1$), then we can define the *discrete-time Fourier transform* (*DtFT*) by substituting $z = e^{j\Omega}$ in the

$Z$-transform pair, that is, the *DtFT* is given by

$$X(\Omega) := DtFT[x(t)] = \sum_{t=-\infty}^{\infty} x(t)e^{-j\Omega t} \tag{2.34}$$

and restricting the contour of integration to the unit circle with $dz = je^{j\Omega}d\Omega$, we obtain the *inverse discrete-time Fourier transform* (*IDtFT*) as

$$x(t) = IDtFT[X(\Omega)] = \frac{1}{2\pi}\int_{2\pi} X(\Omega)e^{j\Omega}\,d\Omega \tag{2.35}$$

An important feature of the *DtFT* is that it is *periodic* with period $2\pi$, which follows intuitively since each revolution of the unit circle leads to a periodic repetition of the transform.

The frequency response of discrete-time systems is important for both analysis and synthesis purposes. As mentioned previously, the frequency response of a discrete *LTI* can be determined from its impulse response (nonparametric) or rational transfer function (parametric) representation.

The nonparametric impulse response is simply transformed using the *DtFT*, that is,

$$DtFT[h(t)] \longrightarrow H(e^{j\Omega})$$

where $H$ is a function of the complex variable, $z = e^{j\Omega}$,

$$H(e^{j\Omega}) = Re\ H(e^{j\Omega}) + jIm\ H(e^{j\Omega}) = |H(e^{j\Omega})|\angle H(e^{j\Omega})$$

The magnitude and phase are determined in the usual manner

$$\begin{aligned} |H(e^{j\Omega})| &= \sqrt{[Re\ H(e^{j\Omega})]^2 + [Im\ H(e^{j\Omega})]^2} \\ \angle H(e^{j\Omega}) &= \arctan\left(\frac{Im\ H(e^{j\Omega})}{Re\ H(e^{j\Omega})}\right) \end{aligned} \tag{2.36}$$

So we see that the frequency response of a discrete *LTI* can be determined directly from its impulse response using the *DtFT*.

If the system is represented by a rational transfer function in the $Z$-domain,

$$H(z) = \frac{B(z^{-1})}{A(z^{-1})}$$

then the *frequency response* of the system is given by

$$H(e^{j\Omega}) = H(z)\big|_{z=e^{j\Omega}} = \frac{B(e^{-j\Omega})}{A(e^{-j\Omega})} = \left|\frac{B(e^{-j\Omega})}{A(e^{-j\Omega})}\right| \angle \frac{B(e^{-j\Omega})}{A(e^{-j\Omega})} \tag{2.37}$$

It is important to note that the frequency response of a discrete *LTI* system is *periodic* because of the periodic property of the discrete-time exponential, that is,

$$H(e^{j\Omega}) = H(e^{j(\Omega+2\pi)})$$

which follows immediately from the *DtFT*.

Recall that if we have the transfer function in pole-zero form, then the frequency response is easily determined by its factors, that is,

$$|H(z)| = \frac{B(z^{-1})}{A(z^{-1})} = \frac{\prod_{i=1}^{N_b} |z - z_i|}{\prod_{i=1}^{N_a} |z - p_i|} = \frac{|z - z_1| \cdots |z - z_{N_b}|}{|z - p_i| \cdots |z - p_{N_a}|} \tag{2.38}$$

where $\{z_i\}$ are the zeros and $\{p_i\}$ the poles of $H(z)$ and

$$\angle H(e^{j\Omega}) = \sum_{i=1}^{N_b} \arctan\left(\frac{Im\ z_i}{Re\ z_i}\right) - \sum_{i=1}^{N_a} \arctan\left(\frac{Im\ p_i}{Re\ p_i}\right) \tag{2.39}$$

The relationship between continuous-time and sampled signal frequencies is controlled by the sampling interval $\triangle T$ through the transformation from the Laplace $S$-plane to the $Z$-plane. This transformation is given by

$$z = e^{s\triangle T} = e^{(d \pm j\omega)\triangle T} \tag{2.40}$$

or

$$z = |z|\angle z = \left|e^{d\triangle T}\right| \arctan\{\tan(\pm\omega\triangle T)\} \tag{2.41}$$

for $d$ the damping coefficient and $\omega$, the continuous-time angular frequency. Thus any point in the $S$-plane maps to a unique point in the $Z$-plane with magnitude $|z| = e^{d\triangle T}$ and angle $\angle z = \omega\triangle T$.

We note that the $j\omega$ axis of the $S$-plane maps into the unit circle of the $Z$-plane, that is,

$$|z| = |e^{j\Omega}| = 1 \qquad \text{(Unit circle)} \tag{2.42}$$

and therefore stable poles in the left-half $S$-plane map into stable poles within the unit circle of the $Z$-plane. In fact, those points of constant damping in the $S$-plane map into circles of radius $|z| = e^{d\triangle T}$, while lines of constant frequency in the $S$-plane correspond to radial lines of angle $\angle z = \omega\triangle T$ in the $Z$-plane. This mapping of continuous-time poles to the unit circle leads to the so-called *impulse invariant method* of transforming continuous (Laplace) transfer functions to equivalent discrete ($Z$) transfer functions, that is (see [1] for more details),

$$H(s) = \sum_{i=1}^{N} \frac{R_i}{s - s_i} \longrightarrow H(z) = \sum_{i=1}^{N} \frac{R_i}{1 - e^{s_i\triangle T} z^{-1}} \tag{2.43}$$

The essence of this method is that the impulse response of the discrete system is precisely the sampled values of the continuous response, that is,

$$h(k\triangle T) = h_t\big|_{t=k\triangle T} = \sum_{i=1}^{N} R_i e^{s_i k\triangle T} \tag{2.44}$$

Thus in this transformation the residues $\{R_i\}$ are preserved, and a pole of $H(s)$ at $s = s_i$ is mapped to a pole in $H(z)$ at $z = e^{s_i \triangle T}$. Note that the zeros of $H(s)$ do not correspond to the zeros of $H(z)$. Note also that the Laplace transform of the sampled response is periodic due to the periodic sampler, that is,

$$H(s) = \mathcal{L}\left[h_t \times \delta(t - m\triangle T)\right] = \frac{1}{\triangle T} \sum_{m=-\infty}^{\infty} H(s - jm\omega_s) \tag{2.45}$$

where $\omega_s$ is the corresponding sampling frequency. Therefore, for this transformation to be useful, it is crucial that the continuous system be bandlimited say with $\omega_B$ the continuous-time angular frequency bandwidth. It must be sampled according to the Nyquist sampling theorem [1] with $\omega_s \geq 2\omega_B$ (in practice, $\omega_s \geq 2.5\omega_B$) to avoid aliasing of higher frequency poles into those of apparent lower frequencies.

The impulse invariant transformation enables us to determine new system coefficients when the sampling interval is changed. For instance, if we were to "fit" a model to data (system identification) and we wanted to use that model for a new sampling interval, then we would have to rerun the experiment and perform the fit. However, this method circumvents the problem by merely transforming the coefficients. Thus we see that transforming from the $S$-plane to the $Z$-plane can prove quite useful especially when systems are characterized in rational form.

## 2.4 DISCRETE RANDOM SIGNALS

### 2.4.1 Motivation

In analyzing a deterministic signal, we may choose a particular representation to extract specific information. For example, Fourier transformed signals enable us to easily extract spectral content, that is,

$$\text{Transforms}: \quad x(t) \longleftrightarrow X(\Omega)$$

Systems theory plays a fundamental role in our understanding of the operations on deterministic signals. The convolution operation provides the basis for the analysis of signals passed through a linear system as well as its corresponding spectral content:

$$\text{Convolution:} \qquad y(t) = h(t) * x(t)$$

$$\text{Multiplication:} \qquad Y(\Omega) = H(\Omega)X(\Omega)$$

For example, a digital filter is designed to act on a deterministic signal altering its spectral characteristics in a prescribed manner, that is, a low-pass filter eliminates

all of the signal spectral content above its cutoff frequency and the resulting output is characterized by

$$\text{Filtering:} \quad Y_f(\Omega) = H_f(\Omega)X(\Omega)$$

Consider the following example to illustrate the application of transforms and linear systems theory in deterministic signal processing:

***Example 2.3*** Suppose that we have a measured signal, $x(t)$, consisting of two sinusoids

$$\underbrace{x(t)}_{\text{Measurement}} = \sin 2\pi \underbrace{(10)\,t}_{\substack{10\text{ Hz}\\ \text{Signal}}} + \sin 2\pi \underbrace{(20)\,t}_{\substack{20\text{ Hz}\\ \text{Disturbance}}}$$

and we would like to remove the disturbance at 20 Hz. A low-pass filter $h_F(t) \leftrightarrow H_F(\Omega)$ can easily be designed extract the signal at 10 Hz. Using *MATLAB*, we simulate this signal and the results are shown in Figure 2.1*a*. Here we see the temporal sinusoidal signal and corresponding spectrum showing the signal and disturbance

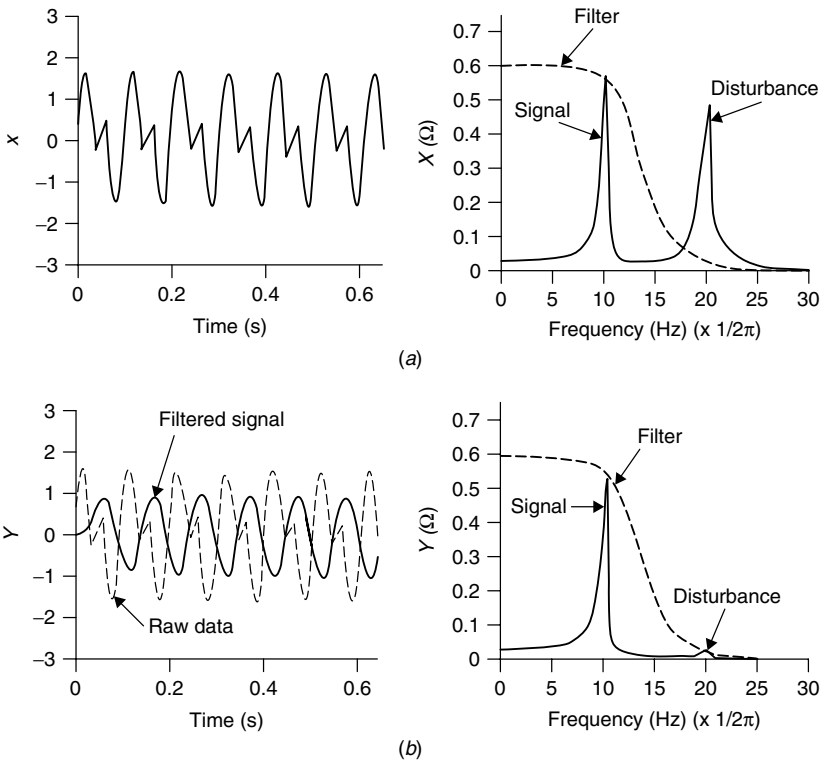



**Figure 2.1.** Deterministic signal processing: (*a*) Analysis of raw data and spectrum. (*b*) Processed data and spectrum.

spectral peaks at 10 and 20 Hz, respectively. After designing the parameters of the filter, we know from linear systems theory that the filtered or processed output, $y(t)$, and its corresponding spectrum satisfy the following relations

$$y_f(t) = h_f(t) * x(t)$$
$$Y_f(\Omega) = H_f(\Omega)X(\Omega)$$

The results of the processing are shown in Figure 2.1*b*. Here we see the operation of the deterministic filter on the measured data. Analyzing the results demonstrates that the processor has extracted the desired signal information at 10 Hz and rejected the disturbance at 20 Hz. This completes the example.

Clearly, we can apply a filter to a random signal, but since its output is still random, we must find a way to eliminate or reduce this randomness in order to employ the powerful techniques available from systems theory. We will show that techniques from statistics combined with linear systems theory can be applied to extract the desired signal information and reject the disturbance or noise. In this case the filter is called an *estimation filter* or simply an *estimator* and it is required to extract the useful information (signal) from noisy or random measurements.

Techniques similar to deterministic linear systems theory hold when a random signal is transformed to its corresponding (auto) *covariance sequence* and its Fourier spectrum transformed to the (auto) *power spectrum*, that is,

$$\text{(Random)} \quad x(t) \longrightarrow R_{xx}(k) \quad \text{(Deterministic)}$$
$$\text{(Random)} \quad X(\Omega) \longrightarrow S_{xx}(\Omega) \quad \text{(Deterministic)}$$

Once this transformation is accomplished, then the techniques of linear systems theory can be applied to obtain results similar to deterministic signal processing. In fact, we will show that the covariance sequence and power spectrum constitute a Fourier transform pair, analogous to a deterministic signal and its corresponding spectrum, that is, we will show that

$$\text{Fourier transform:} \quad R_{xx}(k) \longleftrightarrow S_{xx}(\Omega)$$

As in the deterministic case we can analyze the spectral content of a random signal by investigating its power spectrum.

Techniques of linear systems theory for random signals are valid, just as in the deterministic case where the covariance at the output of a system excited by a random signal $x(t)$, is given by the convolutional relationships as

$$\text{Convolution:} \quad R_{yy}(k) = h(t) * h(-t) * R_{xx}(k)$$
$$\text{Multiplication:} \quad S_{yy}(\Omega) = H(\Omega)H^*(\Omega)S_{xx}(\Omega)$$

Analogously, the filtering operation is performed by an estimation filter, $\hat{H}_f$, designed to shape the output *PSD*, similar to the deterministic filtering operation, that is,

$$\text{Filtering:} \quad S_{yy}(\Omega) = |\hat{H}_f(\Omega)|^2 S_{xx}(\Omega)$$

where the $\hat{}$ notation is used to specify an estimate as defined previously.

Consider the following example of analyzing a random signal using covariance and spectral relations:

***Example 2.4*** Suppose that we have a measured signal given by

$$\underset{\text{Measurement}}{x(t)} = \underset{\text{Signal}}{s(t)} + \underset{\text{Noise}}{n(t)}$$

and we would like to extract the signal and reject the noise, so we design an estimation filter,

$$\hat{y}_f(t) = \hat{h}_f(t) * x(t)$$
$$\hat{Y}_f(\Omega) = \hat{H}_f(\Omega) X(\Omega)$$

Using *MATLAB* our signal in this case will be the sinusoids at 10 and 20 Hz, and the noise is additive random. In Figure 2.2*a* we see the random signal and its raw (random) Fourier spectrum. Note that the noise severely obscures the sinusoidal

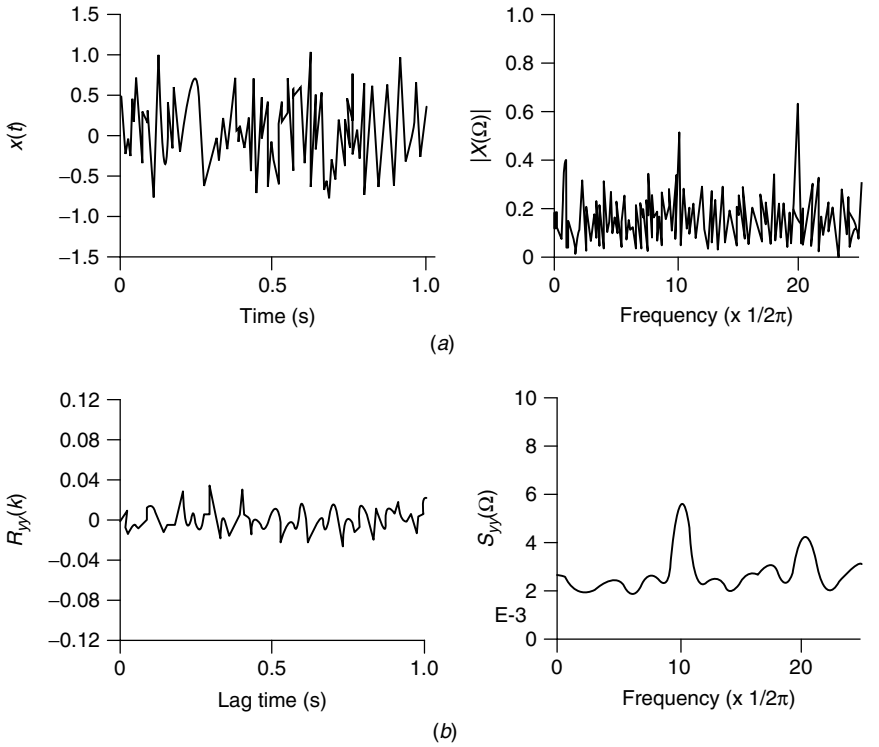



**Figure 2.2.** Random signal processing. *(a)* Raw data and spectrum. *(b)* Covariance data and power spectrum.

signals and there are many false peaks that could erroneously be selected as sinusoidal signals. Transforming the random signal to its corresponding covariance sequence, and estimating the power spectrum using statistical methods, we can now easily see the sinusoidal signals at the prescribed frequencies as depicted in Figure 2.2*b*. So we see that analogous to deterministic signal processing techniques that replacing (estimating) random signals with covariance and power spectra enable the powerful methods of linear systems theory to be applied to extract useful signal information for noisy data. This completes the introduction to random signals.

### 2.4.2 Random Signals

In this subsection we review some of the basic concepts of stochastic processes which are essential to characterize a random signal. We limit our discussion to discrete processes and refer the reader to a more specialized text such as [3] for more detailed information (also see Appendix A).

Stochastic processes find their roots in probability theory, which provides the basis for problem solving. The idea of an underlying probability or sample space is essential to the treatment of stochastic processes. Recall that a *probability or sample space*, $\Xi$ is a collection of samples or experimental outcomes, $\xi$, which are elements of the space. Certain subsets of $\Xi$ or collections of outcomes are called *events*. For example, if we consider flipping a fair coin, then the sample space consists of the set of outcomes, $\Omega = \{H, T\}$ and the events, $\{0, \{H\}, \{T\}\}$. When we attempt to specify how likely it is a particular event will occur during a trial or experiment, then we define the notion of *probability*. In terms of our probability space, we define a *probability function*, Pr(.), as a function defined on a class of events that assigns a number to the likelihood of a particular event occurring during an experiment. We must constrain the class of events on which the probability function is defined so that set operations performed produce sets that are still events. Therefore the class of sets satisfying this constraint is called a *field (Borel)*. Now, if we have a space, $\Xi$, a field, $B$, and a probability function, Pr, we can precisely define an *experiment* as the triple, $\{\Xi, B, \Pr\}$. With the idea of a sample space, probability function, and experiment in mind, we can now start to define the concept of a discrete random signal more precisely.

Recall that a signal is considered *random* if its precise value cannot be predicted (not repeatable). In order to characterize this notion, we recall that a discrete *random variable* is defined as a real function whose value is determined by the outcome of an experiment. Simply it assigns (or maps) a real number to each point of a sample space $\Xi$, which consists of all the possible outcomes of the experiment. Notationally, a random variable $X$ is written as

$$X(\xi) = x \qquad \text{for } \xi \epsilon \Xi \tag{2.46}$$

Once we have idea of a (discrete) random variable, then it is possible to define our probability space with associated *probability mass function* defined in terms of

the random variable,[3] that is,

$$\mathrm{P}_X(x_i) = \Pr(X(\xi_i) = x_i) \tag{2.47}$$

and the cumulative *probability distribution function* is defined by

$$\mathrm{F}_X(x_i) = \Pr(X(\xi_i) \leq x_i) \tag{2.48}$$

These are related by

$$\mathrm{P}_X(x_i) = \sum_i \mathrm{F}_X(x_i)\delta(x - x_i) \tag{2.49}$$

$$\mathrm{F}_X(x_i) = \sum_i \mathrm{P}_X(x_i)\mu(x - x_i) \tag{2.50}$$

where $\delta$, and $\mu$ are the unit impulse and step functions, respectively.

It is easy to show that the distribution function is a monotonically increasing function (see [3] for details) satisfying the following properties:

$$\lim_{x_i \to -\infty} \mathrm{F}_X(x_i) = 0$$

and

$$\lim_{x_i \to \infty} \mathrm{F}_X(x_i) = 1$$

These properties can be used to show that the mass function satisfies

$$\sum_i \mathrm{P}_X(x_i) = 1$$

Either the distribution or probability mass function completely describe the properties of a random variable. Given the distribution or mass function, we can calculate probabilities that the random variable has values in any set of events.

If we extend the idea that a random variable is also a function of time or index set as well, then we can define a random signal. More formerly, a random signal or equivalently *stochastic process* is a two dimensional function of $t$ and $\xi$:

$$X(t, \xi), \qquad \xi \in \Xi, t \in \mathcal{I} \tag{2.51}$$

where $\mathcal{I}$ is a set of index parameters (continuous or discrete) and $\Xi$ is the sample space. There are four possible cases of *X(., .)* above:

i. For any $t, \xi$ varying—$X(t, \xi)$—random time function
ii. For fixed t and $\xi$ varying—$X(t_i, \xi)$—random variable

[3] $\mathrm{P}_X(x_i)$ is a shorthand notation with $X$ defining the random variable and $x_i$ its realization.

iii. For fixed $\xi$ and $t$ varying—$X(t, \xi_i)$—random signal
iv. For fixed $\xi$ and fixed $t$—$X(t_i, \xi_i)$—number

So we see that a discrete random signal precisely defined by

$$x_i(t) := X(t, \xi_i) \tag{2.52}$$

is a particular realization of a stochastic process in which each distinct value of time can be interpreted as a random variable. Thus we can consider a random signal, simply a sequence of ordered (in time) random variables. A collection of realizations of a random signals is called an *ensemble*. A common representation of a random process is given in the next example.

***Example 2.5*** *Random Walk* Suppose that we toss a fair coin at each time instant, that is,

$$X(t, \xi_i), \qquad t \in T = \{0, 1, \ldots, N-1\}, \ \xi \in \Xi = \{0, \{H\}, \{T\}\}$$

where the random variable is given by

$$X(t, \xi_i) = \begin{cases} +K, & \xi_1 = H \\ -K, & \xi_2 = T \end{cases}$$

We define the corresponding probability mass function as

$$\mathrm{P}_X(X(t, \xi_i) = \pm K) = \tfrac{1}{2}$$

Here we see that for each $t \in T$, $X(t, \xi)$ is just a random variable. Now, if we define the function,

$$y(N, \xi_i) = \sum_{t=0}^{N-1} X(t, \xi_i), \qquad i = 1, 2, \ldots$$

as the position at $t$ for the $i$th realization, then for each $t$, $y$ is a sum of discrete random variables, $X(., \xi_i)$, and for each $i$, $X(t, \xi_i)$ is the $i$th realization of the process. For a given trial, we have $\xi_1 = \{HHTHTT\}$, and $\xi_2 = \{THHTTT\}$, then for these realizations, we see the ensemble depicted in Figure 2.3. Note that $y(5, \xi_1) = 0$, and $y(5, \xi_2) = -2K$, which can be predicted using the given random variables and realizations. This completes the example.

In summary, we see that a random signal is a discrete stochastic process, which is a two-dimensional function of time (or index set) and samples of the probability space. In fact, this is the case, the resulting statistic merely becomes a function of the index parameter $t$ (time) in our case.

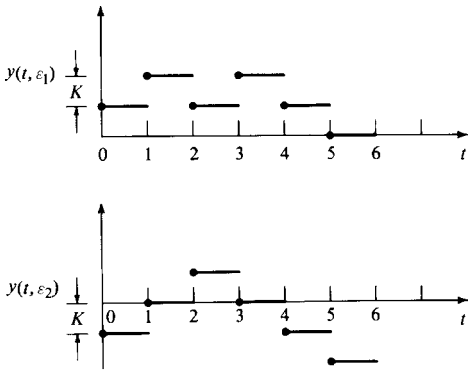



**Figure 2.3.** Ensemble of random walk realizations.

The basic statistics and properties hold for stochastic processes, that is,

| | | |
|---|---|---|
| Expected value: | $m_x(t)$ | $= E\{x(t)\} = \sum_i x_i(t)\mathrm{P}_X(x_i(t))$ |
| Mean squared: | $\Psi_{xx}(t,k)$ | $= E\{x(t)x(k)\}$ |
| Variance: | $R_{xx}(t,k)$ | $= E\{(x(t)-m_x(t))(x(k)-m_x(k))\}$ |
| | | $= \Psi_{xx}(t,k) - m_x(t)m_x(k)$ |
| Covariance: | $R_{xy}(t,k)$ | $= E\{(x(t)-m_x(t))(y(k)-m_y(k))\}$ |
| Linearity: | $E\{ax(t)+b\}$ | $= aE\{x(t)\}+b = am_x(t)+b$ |
| Independence: | $E\{x(t)y(t)\}$ | $= E\{x(t)\}E\{y(t)\}$ |
| Covariance: | | |
| Uncorrelated: | $E\{x(t)y(t)\}$ | $= E\{x(t)\}E\{y(t)\} \quad \{R_{xy}(t,k)=0\}$ |
| Orthogonal: | $E\{x(t)y(t)\}$ | $= 0$ |

Consider the following example that demonstrates the statistical operations on a discrete stochastic process.

***Example 2.6*** Suppose that we have a discrete random signal given by

$$y(t) = a + bt$$

where $a$, $b$ are random variables, then the mean and variance of this process can be calculated by

$$\begin{aligned} m_y(t) &= E\{y(t)\} = E\{a+bt\} = E\{a\} + E\{b\}t = m_a + m_b t \\ R_{yy}(t,k) &= E\{(a+bt)(a+bk)\} - m_y(t)m_y(k) \\ &= E\{a^2\} + E\{ab\}(t+k) + E\{b^2\}tk - m_y(t)m_y(k) \\ R_{yy}(t,k) &= \Psi_{aa} + R_{ab}[t+k] + \Psi_{bb}[tk] - m_y(t)m_y(k) \end{aligned}$$

This completes the example.

Note that the expected value operation implies that for random signals these basic statistics are calculated *across* the ensemble. For example, if we want to calculate the mean of a process, that is,

$$m_x(t) = E\{X(t, \xi_i) = x_i(t)\}$$

we simply take the values of $t = 0, 1, \ldots$ and calculate the mean for each value of time *across* $(i = 1, 2, \ldots)$ the ensemble. So we see that dealing with random signals is similar to dealing with random variables except that we must account for the time indexes.

We must consider properties of random signals that make them useful, but as before, we first develop the ideas from probability theory and then define them in terms of random signals. Suppose that we have more than one random variable, then we define the *joint mass* and distribution functions of an $N$-dimensional random variable as

$$\mathrm{P}_X(x_1, \ldots, x_N), \quad \mathrm{F}_X(x_1, \ldots, x_N)$$

All of the basic statistical definitions remain as before, except that we replace the scalar with the joint functions. Clearly, if we think of a random signal as a sequence of ordered random variables, then we are dealing with joint probability functions. Suppose that we have two random variables, $x_1$ and $x_2$, and we know that the latter has already assumed a particular value, then we can define the *conditional probability mass function* of $x_1$ given that $X(\xi_2) = x_2$ has occurred by

$$\Pr(x_1|x_2) := \mathrm{P}_X(X(\xi_1)|X(\xi_2) = x_2) \tag{2.53}$$

and it can be shown from basic probabilistic axioms (see [3]) that

$$\Pr(x_1|x_2) = \frac{\Pr(x_1, x_2)}{\Pr(x_2)} \tag{2.54}$$

Note also that this equation can also be written as

$$\Pr(x_1, x_2) = \Pr(x_2|x_1)\Pr(x_1) \tag{2.55}$$

Substituting this equation into Eq. (2.54) gives *Bayes' rule*, that is,

$$\Pr(x_1|x_2) = \Pr(x_2|x_1)\frac{\Pr(x_1)}{\Pr(x_2)} \tag{2.56}$$

If we use the definition of joint mass function and substitute in the previous definitions, then we obtain the *probabilistic chain rule* of conditional probabilities,

$$\begin{aligned}&\Pr(x_1, \ldots, x_N)\\ &\quad = \Pr(x_1|x_2, \ldots, x_N)\ \Pr(x_2|x_3, \ldots, x_N) \cdots\ \Pr(x_{N-1}|x_N)\Pr(x_N)\end{aligned} \tag{2.57}$$

Along with these definitions follows the idea of conditional expectation, that is,

$$E\{x_i|x_j\} = \sum_i X_i \Pr(x_i|x_j)$$

The conditional expectation is also a *linear* operator that possesses many useful properties, that is,

$$E\{ax_i + b|x_j\} = aE\{x_i|x_j\} + b \tag{2.58}$$

A common class of useful random signals is called Markov processes. Markov processes are defined in terms of a set of random signals $\{X(t)\}$, we say that a process is *Markov* if and only if

$$\Pr(X(t+1)|X(t), X(t-1), \ldots, X(1)) = \Pr(X(t+1)|X(t)) \tag{2.59}$$

that is, the future $X(t+1)$ depends only on the present $X(t)$ and not the past $\{X(t-1), \ldots, X(1)\}$. Thus, if a process is Markov, then the probabilistic chain rule of Eq. (2.57) simplifies considerably, that is,

$$\Pr(x_1, \ldots, x_N) = \Pr(x_1|x_2)\Pr(x_2|x_3)\cdots\Pr(x_{N-1}|x_N)\Pr(x_N)$$

Just as in systems theory the concept of a time-invariant system is essential for analytical purposes, so is the equivalent concept of stationarity essential for the analysis of random signals. We say that a stochastic process is *stationary* if the joint mass function is time invariant, that is,

$$\Pr(X(1)\cdots X(N)) = \Pr(X(1+k)\cdots X(N+k)) \qquad \forall\ k \tag{2.60}$$

If the equation is only true for values of $t \leq M < N$, then it is said to be *stationary of order M*. Special cases arising are as follows:

- For $M = 1$, $\Pr(X(t)) = \Pr(X(t+k)) = \Pr(X)\ \forall k \Rightarrow m_x(t) = m_x$ a constant (*mean stationary*).
- For $M = 2$, $\Pr(X(t_1), X(t_2)) = \Pr(X(t_1+k), X(t_2+k))$ or for $k = t_2 - t_1$, $\Pr(X(t_2), X(t_2+k))$ (*covariance stationary*), which implies that

$$R_{xx}(t, t+k) = R_{xx}(k) = E\{x(t)x(t+k)\} - m_x^2$$

- A process that is both mean and covariance stationary is called *wide-sense stationary* (*WSS*).

***Example 2.7*** Determine if the following process is wide-sense stationary

$$x(t) = A\cos t + B\sin t \quad \text{(For } A \text{ and } B \text{ uncorrelated zero-mean with variance } \sigma^2\text{)}$$

$$
\begin{aligned}
m_x(t) &= E\{A\}\cos t + E\{B\}\sin t = 0 \\
R_{xx}(t,k) &= E\{x(t)x(t+k)\} - 0 \\
&= E\{(A\cos t + B\sin t)(A\cos(t+k) + B\sin(t+k))\} \\
&= E\{A^2\}\cos t\cos(t+k) + E\{B^2\}\sin t\sin(t+k) \\
&\quad + E\{AB\}\sin t\cos(t+k) \\
&= \sigma^2\cos(t-t+k) = \sigma^2\cos k
\end{aligned}
$$

which implies that the process is *WSS*.

A process is said to be *ergodic* if its time average is identical to its ensemble average, that is,

$$E\{x(t,\xi_i)\} = E_T\{x(t)\} = \lim_{N\to 0}\frac{1}{2N+1}\sum_{t=-N}^{N} x(t) \tag{2.61}$$

This means that all of the statistical information in the ensemble can be obtained from one realization of the process. For instance, consider an ensemble of random signals, in order to calculate the mean of this process, we must average across the ensemble for each instant of time to determine, $m_x(t)$. However, if the process is ergodic, then we may select any member (realization) of the ensemble and calculate its time average $E_T\{x_i(t)\}$ to obtain the required mean, that is,

$$m_x(t) = m_x$$

Note that an ergodic process must be *WSS*, but the inverse does not necessarily hold.

We define two important processes that will be used extensively to model random signals. The first is the *uniformly* distributed process which is specified by the mass function

$$\mathrm{P}_X(x_i) = \frac{1}{b-a}, \qquad a < x_i < b \tag{2.62}$$

or simply

$$x \sim \mathcal{U}(a,b)$$

where the random variable can assume any value within the specified interval. The corresponding mean and variance of the uniform random variable is given by

$$m_x = \frac{b+a}{2}, \quad R_{xx} = \frac{(b-a)^2}{12} \tag{2.63}$$

Second, the *gaussian* or *normal process* is defined by its probability mass function

$$\Pr(X(t)) = \frac{1}{\sqrt{2\pi R_{xx}}}\exp\left\{-\frac{1}{2}\frac{(X(t)-m_x)^2}{R_{xx}}\right\} \tag{2.64}$$

or simply,

$$x \sim \mathcal{N}(m_x, R_{xx})$$

where $m_x$ and $R_{xx}$ are the respective mean and variance completely characterizing the gaussian process, $X(t)$.

The *central limit theorem* makes gaussian processes very important, since it states that the sum of a large number of *independent random variable*s tends to be gaussian, that is, for $\{x(i)\}$ independent then $y(t) \sim \mathcal{N}(0, 1)$, where

$$y(t) = \sum_{i=1}^{N} \frac{(x(i) - m_x(i))}{\sqrt{N R_{xx}(i)}} \tag{2.65}$$

Both of these distributions will become very important when we attempt to simulate stochastic processes on the computer. Other properties of the gaussian process that are useful are as follows:

- *Linear transformation.* Linear transformations of gaussian variables are gaussian; that is, if $x \sim \mathcal{N}(m_x, R_{xx})$ and $y = ax + b$. Then

$$y \sim \mathcal{N}(am_x + b, a^2 R_{xx}) \tag{2.66}$$

- *Uncorrelated gaussian variables.* Uncorrelated gaussian variables are independent.
- *Sums of gaussian variables.* Sums of independent gaussian variables yield a gaussian distributed variable with mean and variance equal to the sums of the respective means and variances; that is,

$$x_i \sim \mathcal{N}(m_x(i), R_{xx}(i)) \quad \text{and} \quad y = \sum_i k_i x(i)$$

Then

$$y \sim N\left(\sum_i k_i m_x(i), \sum_i k_i^2 R_{xx}(i)\right) \tag{2.67}$$

- *Conditional gaussian variables.* Conditional gaussian variables are gaussian distributed; that is, if $x_i$ and $x_j$ are jointly gaussian. Then

$$E\{x_i | x_j\} = m_{x_i} + R_{x_i x_j} R_{x_j x_j}^{-1} (x_j - m_{x_j})$$

and

$$R_{x_i | x_j} = R_{x_i x_i} - R_{x_i x_j} R_{x_j x_j}^{-1} R_{x_j x_i} \tag{2.68}$$

- *Orthogonal errors.* The estimation error $\tilde{x}$ is orthogonal to the conditioning variable; that is, for $\tilde{x} = x - E\{x|y\}$. Then

$$E\{y\tilde{x}\} = 0$$

- *Fourth gaussian moments.* The fourth moment of zero-mean gaussian variables is given by

$$E\{x_1x_2x_3x_4\} = E\{x_1x_2\}E\{x_3x_4\} + E\{x_1x_3\}E\{x_2x_4\} + E\{x_1x_4\}E\{x_2x_3\} \tag{2.69}$$

This discussion completes the introductory concepts of relating random signals to the probabilistic notion of stochastic processes. In the course of applications the ideas of probability spaces, and the like, sometimes get lost in an effort to simplify notation. So it is very important for the reader to be aware that in many texts on random signals, it will be defined that $x(t)$ is a random signal and it is assumed that the reader will know that

$$x(t) \longrightarrow X(t, \xi_i) \qquad t \in \mathcal{I}, \xi \in \Xi$$

We now have all of the probabilistic ingredients to begin to characterize useful random signals and the corresponding tools to analyze as well as synthesize them. In the next section we discuss the covariance and power spectrum and show how they can be used to specify the properties of random signals.

## 2.5 SPECTRAL REPRESENTATION OF RANDOM SIGNALS

In many engineering problems before the design of processing algorithms can proceed, it is necessary to analyze the measured signal. As we have stated, analogous to the deterministic case, the covariance function replaces the random signal and the power spectrum its transform. In this section we derive the relationship between the covariance function and power spectrum and show how they can be used to characterize fundamental random signals.[4]

We begin by defining the power spectrum of a discrete random signal. Recall that the discrete-time Fourier transform *DtFT* pair is given by

$$X(e^{j\Omega}) := DtFT[x(t)] = \sum_{t=-\infty}^{\infty} x(t)e^{-j\Omega t}$$

$$x(t) = IDtFT[X(e^{j\Omega})] = \frac{1}{2\pi}\int_{2\pi} X(e^{j\Omega})e^{j\Omega t}d\Omega \tag{2.70}$$

The transform converges if $x(t)$ is absolutely summable, that is,

$$\sum_{t=-\infty}^{\infty} |x(t)| < \infty$$

[4]We will return to more mathematically precise notation for this development, that is, $X(\Omega) = X(e^{j\Omega})$.

or if the sequence has finite energy

$$\sum_{t=-\infty}^{\infty} |x(t)|^2 < \infty$$

If $x(t)$ is random, then $X(e^{j\Omega})$ is *also* random because

$$x(t, \xi_i) \Longleftrightarrow X(e^{j\Omega}, \xi_i) \qquad \forall\, i$$

both are simply realizations of a random signal over the ensemble generated by $i$. Also, and more important, $X(e^{j\Omega})$ for stationary processes almost never exists because any nonzero realization $x(t, \xi_i)$ is not absolutely summable in the ordinary sense (these integrals can be modified (see [4] for details)). Even if the $Z$-transform is applied instead of the *DtFT*, it can be shown that convergence problems will occur in the inverse transform. So we must somehow modify the constraints on our signal to ensure convergence of the *DtFT*.

Consider a finite duration realization defined by

$$x_N(t) := \begin{cases} x_N(t, \xi_i), & |t| \le N < \infty \\ 0, & |t| > N \end{cases} \tag{2.71}$$

Note that $x_N(t)$ will be absolutely summable ($N$ finite) if $x(t)$ has a finite mean-squared value. Also $x_N(t)$ will have finite energy, so it will be Fourier transformable. The average power of $x_N(t)$ over the interval $(-N, N)$ is

$$\text{Average power} = \frac{1}{2N+1} \sum_{t=-N}^{N} x^2(t) = \frac{1}{2N+1} \sum_{t=-\infty}^{\infty} x_N^2(t) \tag{2.72}$$

but by Parseval's theorem [1] for discrete signals, we have

$$\sum_{t=-\infty}^{\infty} x_N^2(t) = \frac{1}{2\pi} \int_{2\pi} |X_N(e^{j\Omega})|^2 d\Omega \tag{2.73}$$

where $|X_N(e^{j\Omega})|^2 = X_N(e^{j\Omega}) X_N^*(e^{j\Omega})$, and $^*$ denotes the complex conjugate. Substituting Eq. (2.73) into Eq. (2.72), we obtain

$$\text{Average power} = \int_{2\pi} \left( \frac{|X_N(e^{j\Omega})|^2}{2N+1} \right) \frac{d\Omega}{2\pi} \tag{2.74}$$

The quantity in brackets represents the average power per unit bandwidth and is called the power spectral density of $x_N(t)$. Since $x_N$ is a realization of a random

signal, we must average over the ensemble of realizations. Therefore we define the *power spectral density* (*PSD*) of $x_N(t)$ as

$$S_{x_N x_N}(e^{j\Omega}) = E\left\{\frac{|X_N(e^{j\Omega})|^2}{2N+1}\right\} \tag{2.75}$$

But, since $x_N \to x$ as $N \to \infty$, we have

$$S_{xx}(e^{j\Omega}) = \lim_{N\to\infty} E\left\{\frac{|X_N(e^{j\Omega})|^2}{2N+1}\right\} \tag{2.76}$$

Now let us develop the relationship between the *PSD* and covariance. We can write Eq. (2.76) as

$$\begin{aligned} S_{xx}(e^{j\Omega}) &= \lim_{N\to\infty} \frac{1}{2N+1} E\left\{X_N(e^{j\Omega}) X_N^*(e^{j\Omega})\right\} \\ &= \lim_{N\to\infty} \frac{1}{2N+1} E\left\{\left(\sum_{t=-N}^{N} x(t)e^{-j\Omega t}\right)\left(\sum_{m=-N}^{N} x(m)e^{+j\Omega m}\right)\right\} \end{aligned}$$

or

$$S_{xx}(e^{j\Omega}) = \lim_{N\to\infty} \frac{1}{2N+1} E\left\{\sum_{t=-N}^{N}\sum_{m=-N}^{N} x(t)x(m)e^{-j\Omega(t-m)}\right\}$$

Moving the expectation operation inside the summation gives the relation

$$S_{xx}(e^{j\Omega}) = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{t=-N}^{N}\sum_{m=-N}^{N} R_{xx}(t,m)e^{-j\Omega(t-m)} \tag{2.77}$$

Introduce a change of variable $k$ as

$$k = t - m \Rightarrow R_{xx}(m+k, m)$$

and substitute into Eq. (2.77) to obtain

$$S_{xx}(e^{j\Omega}) = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{k=-N-m}^{N}\sum_{m=-N}^{N-m} R_{xx}(m+k,m)e^{-j\Omega k}$$

or

$$S_{xx}(e^{j\Omega}) = \sum_{k=-\infty}^{\infty}\left\{\lim_{N\to\infty} \frac{1}{2N+1} \sum_{m=-N}^{N} R_{xx}(m+k,m)\right\} e^{-j\Omega k}$$

That is,

$$S_{xx}(e^{j\Omega}) = \sum_{k=-\infty}^{\infty} E_T\{R_{xx}(m+k,m)\}e^{-j\Omega k} \tag{2.78}$$

If we further assume that the process is wide-sense stationary, then $R_{xx}(m+k,m) = R_{xx}(k)$ is no longer a function of time. Therefore

$$S_{xx}(e^{j\Omega}) = \sum_{k=-\infty}^{\infty} R_{xx}(k)e^{-j\Omega k} \tag{2.79}$$

evolves as the well-known *Wiener-Khintchine* relation with the corresponding covariance given by *IDtFT*

$$R_{xx}(k) = \frac{1}{2\pi}\int_{2\pi} S_{xx}(e^{j\Omega})e^{j\Omega k}d\Omega \tag{2.80}$$

Note also that if we recall that the *DtFT* is just the $Z$-transform of $x(t)$ evaluated on the unit circle, that is,

$$S_{xx}(\Omega) = S_{xx}(z)\big|_{z=e^{j\Omega}}$$

Then we obtain the equivalent pair

$$S_{xx}(z) = Z[R_{xx}(k)] = \sum_{k=-\infty}^{\infty} R_{xx}(k)z^{-k}$$

and

$$R_{xx}(k) = \frac{1}{2\pi j}\int S_{xx}(z)z^{k-1}dz \tag{2.81}$$

***Example 2.8*** A discrete random signal given by

$$x(t) = A\sin(\Omega_0 t + \phi)$$

where $\phi$ is uniformly random with $\phi \sim \mathcal{U}(0, 2\pi)$. Suppose that we are asked to analyze its spectral content. We can determine the power spectrum by applying the Wiener-Khintchine theorem to the covariance function which must be determined. First, we must determine the mean, that is,

$$m_x(t) = E\{A\sin(\Omega_0 t + \phi)\} = A\sin\Omega_0 t\, E\{\cos\phi\} + A\cos\Omega_0 t\, E\{\sin\phi\} = 0$$

since $\phi$ is zero-mean. Thus, for a zero-mean process, the covariance or correlation is given by

$$R_{xx}(k) = E\{x(t)x(t+k)\} = E\{A\sin(\Omega_0 t + \phi)A\sin(\Omega_0(t+k) + \phi)\}$$

If we let $y = \Omega_0 t + \phi$, and $z = \Omega_0(t + k) + \phi$, then by the trigonometric identity,

$$\sin y \sin z = \tfrac{1}{2}\cos(y - z) - \tfrac{1}{2}\cos(y + z)$$

we obtain

$$\begin{aligned} R_{xx}(k) &= \frac{A^2}{2} E\{\cos \Omega_0 k - \cos(2(\Omega_0 t + \phi) + \Omega_0 k)\} \\ &= \frac{A^2}{2} \cos \Omega_0 k - \frac{A^2}{2} E\{\cos(2(\Omega_0 t + \phi) + \Omega_0 k)\} \end{aligned}$$

Using the fact that $\phi$ is uniform and calculating the expected value shows that the last term is zero. So we have

$$R_{xx}(k) = \frac{A^2}{2} \cos \Omega_0 |k|$$

From the theorem we have

$$S_{xx}(\Omega) = DtFT[R_{xx}] = DtFT\left[\frac{A^2}{2} \cos \Omega_0 k\right] = \pi A^2[\delta(\Omega - \Omega_0) + \delta(\Omega + \Omega_0)]$$

So we see that a sinusoidal signal with random phase can be characterized by a sinusoidal covariance and impulsive power spectrum at the specified frequency, $\Omega_0$. The functions are shown in Fig 2.4.

We define random signals in terms of their covariance and spectral density. For example, a purely random or *white noise* sequence say, $e(t)$, is a sequence in which all the $e(t)$ are mutually independent, that is, knowing $e(t)$ in no way can be used

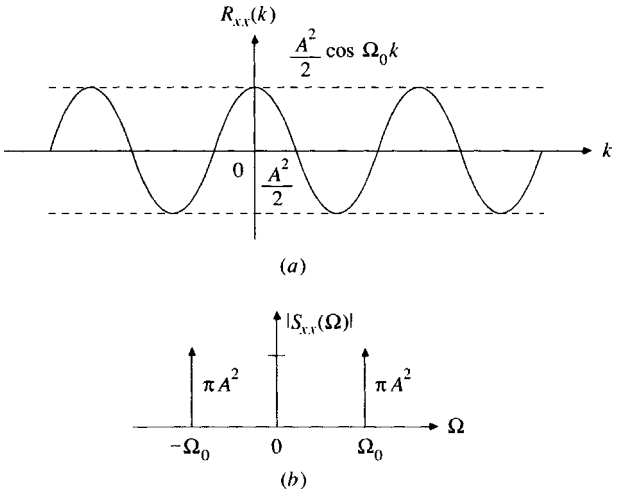



**Figure 2.4.** Random sinusoidal signal. *(a)* Auto-covariance. *(b)* Power spectrum.

to predict $e(t+1)$. A white sequence is called completely random or unpredictable or memoryless (no correlation). The *power spectral density* of white noise is a constant, that is,

$$S_{ee}(\Omega) = R_{ee}$$

The corresponding covariance is given by

$$R_{ee}(k) = R_{ee}\delta(k)$$

where $R_{ee}$ is the variance of the noise.

In fact, the white noise characterization of random signals is analogous to the unit impulse representation of deterministic signals, that is,

$$R_{ee}\delta(k) \Longleftrightarrow A\delta(t)$$

and

$$S_{ee}(\Omega) = R_{ee} \Longleftrightarrow H(\Omega) = A$$

As we will see, this sequence is the random counterpart for random systems of the unit impulse excitation for the analysis of LTI systems. We summarize other useful covariance-spectral density pairs below in Table 2.1. Note that random, or white sequences are also Markov, since they are uncorrelated:

$$\Pr(e(t)|e(t-1), \ldots, e(1)) = \Pr(e(t))$$

Sequences of this type have historically been called white because of their analogy to white light, which possesses all frequencies (constant power spectrum).

If each of the $e(t)$ is also gaussian-distributed, then the sequence is called *white gaussian*. White gaussian noise is easily simulated on a digital computer

**Table 2.1. Covariance-Spectral Density Transformation Pairs**

| Discrete Process | Covariance | Power Spectrum |
|---|---|---|
| White noise | $R_{xx}\delta(k)$ | $R_{xx}$ |
| Bandlimited white noise | $\frac{R_{xx}\Omega_B}{\pi}\frac{\sin k\Omega_B}{k\Omega_B}$ | $R_{xx} \quad \lvert\Omega\rvert \le \Omega_B$ |
| Bias (constant) | $B_x^2 \quad \forall k$ | $2\pi B_x^2\delta(\Omega)$ |
| Exponential | $a^{\lvert k\rvert}, \quad a < 1$ | $\frac{R_{xx}}{(1-ae^{-j\Omega})(1-ae^{j\Omega})}$ |
| Sinusoidal | $\frac{A^2}{2}\cos\Omega_0\lvert k\rvert$ | $\pi A^2\{\delta(\Omega-\Omega_0)+\delta(\Omega+\Omega_0)\}$ |
| Triangular | $((1-\lvert k\rvert T_B) \quad \lvert k\rvert \le T_B$ | $1/T_B^2\sin^2(T_B+1)\Omega/2)/(\sin^2(\Omega/2))$ |

by applying the *central limit theorem*:

$$y(t) = \sum_{i=1}^{N} \frac{x(i) - m_x(i)}{\sqrt{N R_{xx}(i)}} \quad \text{and} \quad y \sim \mathcal{N}(0, 1)$$

The random number generator on the computer generates uniformly distributed random numbers between 0 and 1. Thus the sequence has mean of $\frac{1}{2}$ and variance $\frac{1}{12}$. If we apply the theorem, then we choose $N = 12$. So

$$y(t) = \sum_{i=1}^{12} \frac{x(i) - \frac{1}{2}}{\sqrt{12(\frac{1}{12})}} = \sum_{i=1}^{12} x(i) - 6, \qquad n = 1, 2, \ldots$$

Thus $y \sim \mathcal{N}(0, 1)$. If one desires to generate, for example, a sequence $v(t) \sim \mathcal{N}(m_v, R_{vv})$, then

$$v(t) = y(t)\sqrt{R_{vv}} + m_v$$

***Example 2.9*** A white gaussian sequence, $\mathcal{N}(1, 0.01)$ is generated on a computer using *MATLAB* and shown in Figure 2.5. Notice the two-sigma confidence interval about the sequence (95% of the samples should lie within $\pm 2\sqrt{R_{yy}}$) here only 4.8% of the $\{y(t)\}$ exceed the bound, or "we are 95% confident that the true sequence mean lies within $[1 - 0.2, 1 + 0.2]$." To check the whiteness of the sequence, we perform a statistical hypothesis test (see [5]) using the estimated covariance shown in the figure. The 95% confidence interval for the *whiteness test* is given by

$$I = \left[ R_{yy}(k) - \frac{1.96 R_{yy}(0)}{\sqrt{N}}, R_{yy}(k) + \frac{1.96 R_{yy}(0)}{\sqrt{N}} \right]$$

but $\pm 0.00123$ is the width, so we see that only 3.1% of the $R_{yy}(k)$ lie outside the limit. Therefore the sequence is "statistically" white. Note the covariance appears as a unit impulse and the power spectrum appears flat (constant).

Next let us examine some properties of the discrete covariance function for stationary processes and observe how the properties of the *PSD* evolve. Recall that the covariance function is given by

$$R_{xx}(k) = R_{xx}(t, t + k) = E_x\{x(t)x(t + k)\} - m_x^2$$

where $E$ is the ensemble expectation operator. Covariance functions satisfy the following important properties which we state without proof (see [3] for details):

- *Maximum value.* The autocovariance function has a maximum at zero lag, that is,

$$R_{xx}(0) \geq |R_{xx}(k)|$$

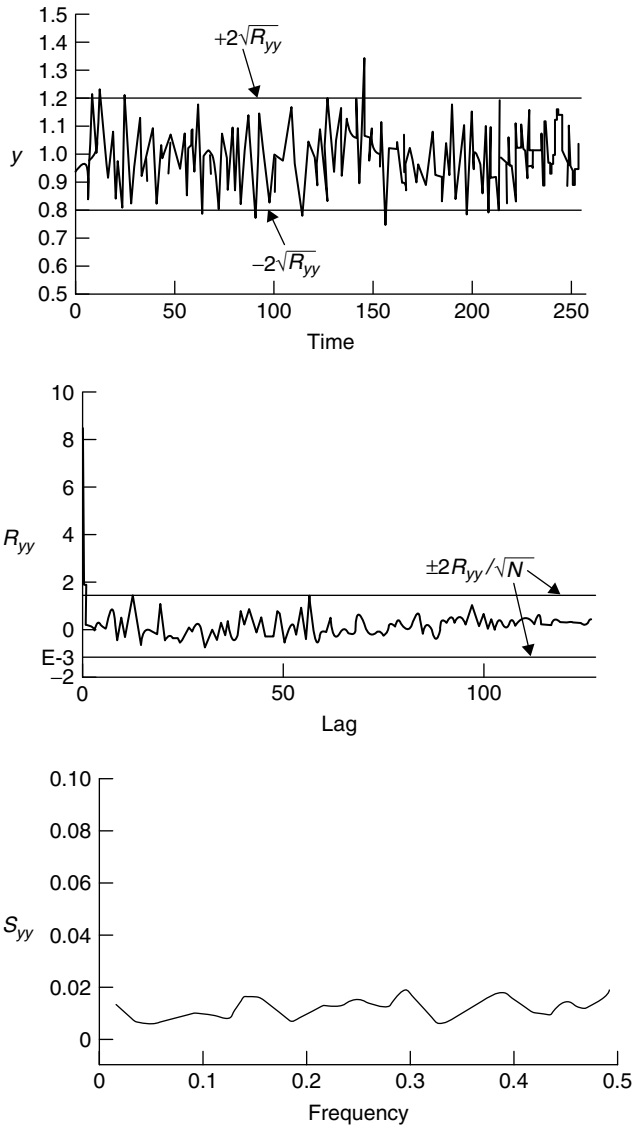



**Figure 2.5.** Simulated white noise, auto-covariance, and power spectrum.

Also the cross-covariance function is bounded by the corresponding maxima,

$$|R_{xy}(k)| \leq \tfrac{1}{2}(R_{xx}(0) + R_{yy}(0))$$

- *Symmetry conditions.* The autocovariance is an even function given by

$$R_{xx}(k) = R_{xx}(-k) = E\{x(t)x(t + (-k))\}$$

while the cross-covariance satisfies

$$R_{xy}(k) = R_{yx}(-k)$$

- *Mean-square value.* The autocovariance function is the mean-square of the process at zero lag,

$$R_{xx}(0) = E\{x^2(t)\}$$

while the cross-covariance mean-square is bounded by

$$|R_{xy}(k)|^2 \leq R_{xx}(0)R_{yy}(0)$$

- *dc Component.* The autocovariance function has a dc component if $x$ is non-zero-mean,

$$R_{xx}(0) = (DC)^2 + R_{xx}(k)$$

- *Periodic component.* The autocovariance has a periodic component if $x$ has a periodic component.

Utilizing these properties of covariance functions, it can be shown that the *average power* in the signal $x(t)$ is given by

$$R_{xx}(0) = E\{x^2(t)\} = \frac{1}{2\pi}\int_{2\pi} S_{xx}(e^{j\Omega})d\Omega$$

which follows directly from the *Wiener-Khintchine* conditions with $k = 0$. It also follows that the *PSD* is a real, even, nonnegative function. By definition, it follows immediately that

$$S_{xx}(e^{j\Omega}) = E\{X(\Omega)X^*(\Omega)\} = E\{|X^2(\Omega)|\}$$

Therefore the (auto) *PSD* is always *real*. The *PSD* is *even*, since

$$S_{xx}(e^{j\Omega}) = DtFT[R_{xx}(k)] = DtFT[R_{xx}(-k)] = S_{xx}(e^{j\Omega})$$

The (cross) *PSD* satisfies a similar property,

$$S_{xy}(e^{j\Omega}) = S_{yx}(e^{-j\Omega}) = S^*_{yx}(e^{j\Omega})$$

The proof that the (auto) *PSD* is nonnegative on the unit-circle, that is, $S_{xx}(e^{j\Omega}) \geq 0$ is involved and therefore we refer the interested reader to [6] for details. It is important to recognize that these properties are essential to analyze the information available in a discrete random signal. For instance, if we are trying to determine phase information about a particular measured signal, we immediately recognize that it is lost in both the autocovariance and corresponding *PSD*.

With these properties in mind, we can now re-examine the expression for the *PSD* and decompose it further as[5]

$$S(z) = \sum_{k=-\infty}^{\infty} R(k)z^{-k} = \sum_{k=0}^{\infty} R(k)z^{-k} + \sum_{k=-\infty}^{0} R(k)z^{-k} - R(0)$$

Letting $i = -k$ in the second sum, we obtain

$$S(z) = \underbrace{\sum_{k=0}^{\infty} R(k)z^{-k}}_{S^+(z)} + \underbrace{\sum_{i=0}^{\infty} R(i)z^{i}}_{S^-(z)} - R(0)$$

the *sum decomposition* of the *PSD*, which is given by

$$S(z) := S^+(z) + S^-(z) - R(0) \tag{2.82}$$

where the one-sided $Z$-transform is, $S^+(z) = Z_I\{R(k)\}$ and $S^-(z) = S^+(z^{-1})$. Note that here $S^{\pm}(z)$ use the $\pm$ to represent positive and negative time relative to inverse transforms. Also it can be shown that $S^+(z)$ has only poles inside the unit circle, while $S^-(z)$ has only those outside (see [6] for details). Thus the sum decomposition can be used to calculate the power spectrum using one-sided transforms of the covariance function. The inverse process is more complicated, since the *PSD* must be decomposed into sums (usually by partial fractions): one having only poles inside the unit circle, and the other only poles outside. The corresponding covariance is then determined using inverse $Z$-transform methods.

The sum decomposition for the power spectrum is analogous in deterministic systems to calculating the two-sided $Z$-transform (or Laplace) given the one-sided transform, since in our case the *PSD* is two-sided, that is,

$$X_{II}(z) = X_I(z) + X_I(z^{-1}) - x(0) \tag{2.83}$$

The sum decomposition is an important mechanism that can be utilized to simulate random signals, as we will see in the next section. Consider how it can be used to determine the *PSD* of a random signal with given covariance.

***Example 2.10*** Given the autocovariance of a zero-mean process as

$$R(k) = e^{-a|k|} = \begin{cases} e^{-ak}, & k > 0 \\ 1, & k = 0 \\ e^{+ak}, & k < 0 \end{cases}$$

[5]In the subsequent sections we will perform operations on the *PSD* using the $Z$-transform notation which follows from the fact that the *DtFT* is merely the $Z$-transform evaluated on the unit circle, that is, $S(\Omega) = S(z)|_{z=e^{j\Omega}}$.

Suppose that we would like to calculate the corresponding *PSD*. Then

$$S^+(z) = Z_I[R(k)] = Z_I\{e^{-ak}\} = \frac{z}{z - e^{-a}}$$

$$S^-(z) = S^+(z^{-1}) = \frac{z^{-1}}{z^{-1} - e^{-a}}$$

$$R(0) = 1$$

The sum decomposition gives

$$\begin{aligned} S(z) &= S^+(z) + S^-(z) - R(0) \\ &= \left(\frac{z}{z - e^{-a}}\right) + \left(\frac{z^{-1}}{z^{-1} - e^{-a}}\right) - 1 \end{aligned}$$

Multiplying and combining terms, we obtain

$$S(z) = \frac{1 - e^{-2a}}{(z - e^{-a})(z^{-1} - e^{-a})} = \frac{1 - e^{-2a}}{(1 + e^{-2a}) - e^{-a}(z + z^{-1})}$$

Let us now reverse the problem at hand. Suppose that we are given $S(z)$ above, how can we calculate $R(k)$? As discussed previously, we can use $Z$-transform inversion technique to obtain the correct result. Using the inversion integral approach [2], we have

$$R(k) = \text{Res}[S(z)z^{k-1};\ z = e^{-a}] = (z - e^{-a})\left[\frac{(1 - e^{-2a})(e^{-a})^{k-1}}{(z - e^{-a})(z^{-1} - e^{-a})}\right]_{z=e^{-a}}$$

$$\text{for} \quad k \geq 0$$

or

$$R(k) = \frac{e^a(1 - e^{-2a})e^{-ak}}{e^a - e^{-a}} = e^{-ak}, \qquad k \geq 0$$

the desired result. This completes the example.

We summarize the important covariance *PSD* properties in Table 2.2.

In practice, the autocovariance and *PSD* find most application in the *analysis* of random signals yielding information about spectral content, periodicities, and so forth, while the cross-covariance and spectrum are used to estimate the properties of two distinct processes (e.g., input and output of a system). The following example illustrates how the cross-covariance and spectrum can be used to estimate the time delay of a signal.

***Example 2.11*** A random signal $x(t)$ that is transmitted through some medium (e.g. an acoustical wave in sonar or electromagnetic wave in radar) and received some time later by a receiver at the same location. Suppose that we would like to determine the delay time $\tau_d$ between the transmitted and received signals, where $\tau_d = d/v$ and $d$ is the distance, $v$ the propagation velocity in the medium. The

**Table 2.2. Properties of Covariance and Spectral Functions**

| Covariance | Power Spectrum |
|---|---|
| | Average power |
| $R_{xx}(0) = E\{x^2(t)\}$ | $R_{xx}(0) = \frac{1}{2\pi}\int_{2\pi} S_{xx}(z)z^{-1}dz$ |
| | Symmetry |
| $R_{xx}(k) = R_{xx}(-k)$ (even) | $S_{xx}(z) = S_{xx}(z^{-1})$ (even) |
| $R_{xy}(k) = R_{yx}(-k)$ | $S_{xy}(z) = S^*_{yx}(z)$ |
| | Maximum |
| $R_{xx}(0) \geq \lvert R_{xx}(k)\rvert$ | $S_{xx}(e^{j\Omega}) \geq 0$ |
| $\frac{1}{2}R_{xx}(0) + \frac{1}{2}R_{yy}(0) \geq \lvert R_{xy}(k)\rvert$ | |
| $R_{xx}(0)R_{yy}(0) \geq \lvert R_{xy}(k)\rvert^2$ | |
| | Real |
| | $S_{xx}(z) = E\{\lvert X(z)\rvert^2\}$ is real |
| | $S_{xy}(z) = E\{X(z)Y^*(z)\}$ is complex |
| | Sum decomposition |
| | $S_{xx}(z) = S^+_{xx}(z) + S^-_{xx}(z) - R_{xx}(0)$ |
| | $S^+_{xx}(z) = Z_I[R_{xx}(k)]$ |
| | $S^-_{xx}(z) = S^+_{xx}(z^{-1})$ |

received signal, if we assume only an attenuation $A$ of the medium, is characterized by

$$r(t) = Ax(t - \tau_d) + n(t)$$

where $n$ is white noise. The cross-covariance of the zero-mean transmitted and received signals is given by,

$$R_{xr}(k) = E\{x(t)r(t + k)\} = AE\{x(t)x(t - \tau_d + k)\} + E\{x(t)n(t + k)\}$$

or

$$R_{xr}(k) = AR_{xx}(k - \tau_d)$$

since $n$ is zero-mean, white, and uncorrelated with $x$. From the *maximum* property of the autocovariance function, we see that the cross-covariance function will achieve a maximum value, when $k = \tau_d$, that is,

$$R_{xr}(\tau_d) = AR_{xx}(0) \qquad \text{for} \quad k = \tau_d$$

If we take the *DtFT* of the received signal, then we have

$$R(\Omega) = AX(\Omega)e^{-j\Omega\tau_d} + N(\Omega)$$

The corresponding cross-spectrum is given by

$$S_{xr}(\Omega) = E\{X(\Omega)R^*(\Omega)\} = AE\{X(\Omega)X^*(\Omega)\}e^{j\Omega\tau_d} + E\{X(\Omega)N^*(\Omega)\}$$

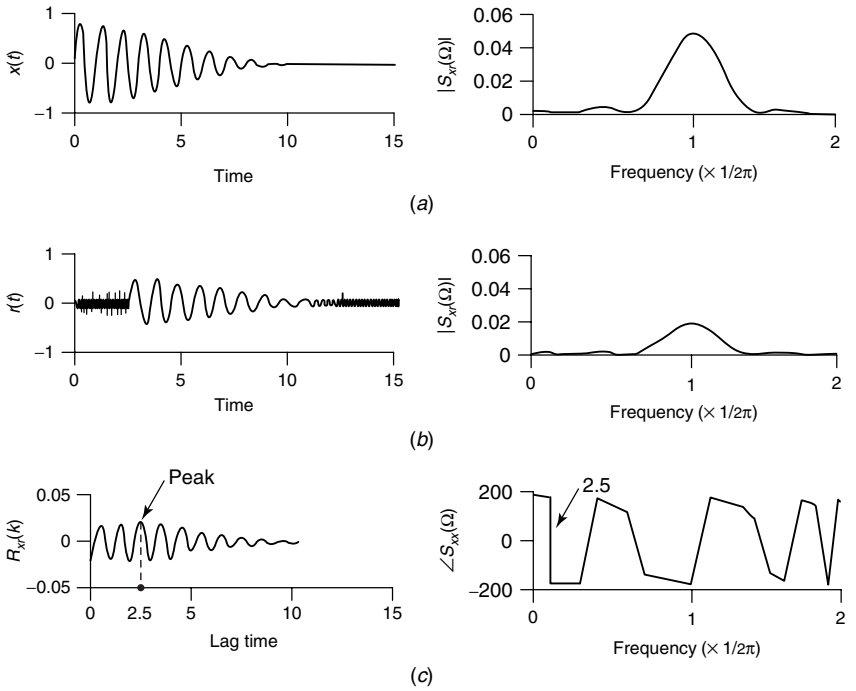



**Figure 2.6.** Time delay simulation. *(a)* Transmitted signal and spectrum. *(b)* Received signal and cross-spectrum magnitude. *(c)* Cross-covariance function and phase spectrum

or

$$S_{xr}(\Omega) = AS_{xx}(\Omega)e^{j\Omega\tau_d}$$

Thus the time delay results in a peak in the cross-covariance function and is characterized by a linear phase component which can be estimated from the slope of the cross-spectral phase function. We use *MATLAB* to simulate a sinusoidal transmitted signal of 1 Hz. The received signal was attenuated by $\frac{1}{2}$, delayed by 2.5 seconds and contaminated with white measurement noise with a variance of 0.01. The transmitted signal and spectrum is shown in Figure 2.6*a*, while the corresponding received signal and magnitude cross-spectrum is shown in Figure 2.6*b*. Note the similarity between both magnitude spectra. The corresponding cross-covariance and spectrum are shown in *c*. From the peak of the estimated cross-covariance function or the corresponding slope of the cross-spectrum, we estimate the delay at approximately $\hat{\tau}_d = 2.5$ seconds and the attenuation as the ratio $A = R_{xr}(\tau_d)/R_{xx}(0) = \frac{1}{2}$. Therefore we estimate

$$\hat{r}(t) = \tfrac{1}{2}\sin 2\pi(t - 2.5)$$

So we see in this case that the properties of covariance and spectra can be used not only to analyze the information available in random signal data but also to estimate

various signal characteristics. We also note in passing that this is precisely the principle behind *matched filtering* utilized extensively in sonar and radar receivers to detect the presence of a target.

## 2.6 DISCRETE SYSTEMS WITH RANDOM INPUTS

When random inputs are applied to linear systems, then covariance and power spectrum techniques must be applied replacing the signal and Fourier spectrum in deterministic signal theory. In this section we develop the relationship between systems and random signals. From linear systems theory we have the convolution or equivalent frequency relations

$$y(t) = h(t) * x(t) = \sum_{k=0}^{\infty} h(k)x(t-k) \tag{2.84}$$

Alternatively, taking discrete-time Fourier transforms, we obtain

$$Y(\Omega) = H(\Omega)X(\Omega) \tag{2.85}$$

If we assume that $x$ is a random signal then as we have seen in the previous section, we must resort to spectral representations of random processes. Exciting a causal linear system with a zero-mean random signal, we obtain the output covariance at lag $k$ as

$$R_{yy}(k) = E\{y(t+k)y(t)\} = E\left\{\left(\sum_{i=0}^{\infty} h(i)x(t+k-i)y(t)\right)\right\}$$

$$= \sum_{i=0}^{\infty} h(i)E\{x(t+k-i)y(t)\}$$

$$R_{yy}(k) = \sum_{i=0}^{\infty} h(i)R_{xy}(k-i) = h(k) * R_{xy}(k) \tag{2.86}$$

Now suppose that we calculate the output power spectrum based on this result, that is,

$$S_{yy}(z) = \sum_{k=-\infty}^{\infty} R_{yy}(k)z^{-k} = \sum_{k=-\infty}^{\infty}\left(\sum_{i=0}^{\infty} h(i)R_{xy}(k-i)\right)z^{-k}$$

We multiply by $z^i z^{-i}$ and interchange the order of summation to obtain

$$S_{yy}(z) = \left(\sum_{i=0}^{\infty} h(i)z^{-i}\right)\left(\sum_{k=-\infty}^{\infty} R_{xy}(k-i)z^{-(k-i)}\right)$$

Let $m = k - i$. Then

$$S_{yy}(z) = \sum_{i=0}^{\infty} h(i)z^{-i} \sum_{m=-\infty}^{\infty} R_{xy}(m)z^{-m} = H(z)S_{xy}(z) \tag{2.87}$$

**Table 2.3. Random Linear System Relations**

| Covariance | Spectrum |
|---|---|
| $R_{yy}(k) = h(k) * h(-k) * R_{xx}(k)$ | $S_{yy}(z) = H(z)H(z^{-1})S_{xx}(z)$ |
| $R_{yy}(k) = h(k) * R_{xy}(k)$ | $S_{yy}(z) = H(z)S_{xy}(z)$ |
| $R_{yx}(k) = h(k) * R_{xx}(k)$ | $S_{yx}(z) = H(z)S_{xx}(z)$ |
| where | |
| $R_{yy}(k) = \frac{1}{2\pi j} \oint S_{yy}(z)z^{k-1}dz$ | $S_{yy}(z) = \sum_{k=-\infty}^{\infty} R_{yy}(k)z^{-k}$ |
| $R_{xy}(k) = \frac{1}{2\pi j} \oint S_{xy}(z)z^{k-1}dz$ | $S_{xy}(z) = \sum_{k=-\infty}^{\infty} R_{xy}(k)z^{-k}$ |

Similar results can be obtained for auto- and cross-covariances and corresponding spectra. We summarize the results in Table 2.3.

***Example 2.12*** A digital filter is given by the difference equation

$$y(t) = ay(t-1) + x(t)$$

Suppose that we would like to analyze the impulse response and corresponding spectrum of this linear system under two conditions: $x(t)$ is deterministic, and $x(t)$ is random. First, we assume that $x$ is deterministic. Then taking the $Z$-transform, we have

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{1 - az^{-1}} \qquad \text{for } |z| > a, \ a < 1$$

and taking the inverse $Z$-transform, we have the corresponding impulse response

$$h(t) = a^t x(t)$$

Next we assume that $x$ is random, uniform with variance $\sigma_{xx}^2$. Using $H(z)$ from the deterministic solution, and the spectral relation from Table 2.3, We obtain

$$S_{yy}(z) = H(z)H(z^{-1})S_{xx}(z) = \left(\frac{1}{1 - az^{-1}}\right)\left(\frac{1}{1 - az}\right)\sigma_{xx}^2$$

From the sum decomposition of the previous section, we have

$$S_{yy}(z) = S_{yy}^{+}(z) + S_{yy}^{-}(z^{-1}) - R_{yy}(0) = \frac{1}{1 - az^{-1}} + \frac{1}{1 - az} - R_{yy}(0)$$

The variance can be calculated directly as

$$R_{yy}(0) = \frac{\sigma_{xx}^2}{1 - a^2}$$

Taking inverse $Z$-transforms, we have

$$\Longrightarrow S_{yy}^{+}(z) = \frac{\sigma_{xx}^2/1-a^2}{1-az^{-1}} \Longrightarrow R_{yy}(k) = \left(\frac{\sigma_{xx}^2}{1-a^2}\right)a^k, \qquad k > 0$$

We can now state an important result that is fundamental for modeling of random signals. Suppose that we have a linear time-invariant system that is asymptotically stable with rational transfer function. If we excite this system with white noise, we get the so-called *spectral factorization theorem*, which states that

**GIVEN** a stable, rational system, then there **EXISTS** a rational $H(\Omega)$ such that

$$S_{yy}(\Omega) = H(\Omega)H^*(\Omega) \tag{2.88}$$

where the poles and zeros of $H(\Omega)$ lie within the unit circle.

If we can represent spectral densities in properly factored form, then all stationary processes can be thought of as outputs of dynamical systems with white-noise inputs. Synthesizing random signals with given spectra, then, requires only the generation of white-noise sequences. We summarize this discussion with the *representation theorem*:[6]

**GIVEN** a rational spectral density $S_{yy}(\Omega)$. Then there **EXISTS** an asymptotically stable linear system when excited by white noise that produces a stationary output $y(t)$ with this spectrum.

The simulation of random signals with given statistics requires the construction of the power spectrum from the sum decomposition followed by a spectral factorization. This leads to the *spectrum simulation procedure*:

1. Calculate $S_{yy}(z)$ from the sum decomposition.
2. Perform the spectral factorization to obtain $H(z)$.
3. Generate a white-noise sequence of variance $R_{xx}$.
4. Excite the system $H(z)$ with the sequence.

The most difficult part of this procedure is to perform the spectral factorization. For simple systems the factorization can be performed by equating coefficients of the known $S_{yy}(z)$, obtained in step 1, with the unknown coefficients of the spectral factor and solving the resulting nonlinear algebraic equations, that is,

$$S_{yy}(z) = \begin{cases} S_{yy}^{+}(z) + S_{yy}^{-}(z) - R_{yy}(0) = \dfrac{N_s(z, z^{-1})}{D_s(z, z^{-1})} \\ H(z)H(z^{-1})R_{xx} = \dfrac{N_H(z)}{D_H(z)}\dfrac{N_H(z^{-1})}{D_H(z^{-1})}R_{xx} \end{cases} \tag{2.89}$$

[6]This is really a restricted variant of the famous Wold decomposition for stationary random signals (see [7] for details).

$$N_s(z, z^{-1}) = N_H(z)N_H(z^{-1})$$
$$D_s(z, z^{-1}) = D_H(z)D_H(z^{-1})$$

For higher order systems more efficient iterative techniques exist including multidimensional systems (see [8], [9], [10]). We conclude this section with a simple example to demonstrate the procedure.

***Example 2.13*** Suppose that we would like to generate a random sequence $\{y(t)\}$ with autocovariance

$$R_{yy}(kT) = a^{|kT|} \quad \text{for} \quad 0 < a < 1$$

From the definition of spectral density with, we obtain

$$S_{yy}(z) = S^{+}_{yy}(z) + S^{-}_{yy}(z) - R_{yy}(0)$$
$$S_{yy}(z) = \frac{1}{(1 - az^{-1})} + \frac{1}{(1 - az)} - 1$$
$$S_{yy}(z) = \frac{(1 - a^2)}{(1 - az^{-1})(1 - az)} = H(z)H(z^{-1})R_{xx}$$

Since $S_{yy}$ is already in factored form, we identify $H(z)$ by inspection,

$$N_s(z, z^{-1}) = N_H(z)N_H(z^{-1}) = 1$$

and

$$D_s(z, z^{-1}) = 1 - a(z + z^{-1}) + a^2 = D_H(z)D_H(z^{-1}) = (1 - az^{-1})(1 - az)$$

Using the representation theorem, we identify the white noise sequence

$$S_{xx}(z) = R_{xx} = (1 - a^2) \quad \text{and} \quad H(z) = \frac{1}{(1 - az^{-1})}$$

or

$$y(t) = h(t) * x(t) = \sum_{k=0}^{\infty} a^{|t-k|}x(k)$$

This example completes the section. In the subsequent sections we consider other models that can be used to represent random signals.

## 2.7 *ARMAX (AR, ARX, MA, ARMA)* MODELS

In the previous section we showed the relationship between linear systems and spectral shaping, that is, generating processes with specified statistics. To generate such a sequence, we choose to use two models which are equivalent–the input-output

or state-space models. Each model set has its own advantages: the input-output models are easy to use, while the state-space models are easily generalized.

The *input-output* or *transfer function* model is familiar to engineers and scientists because it is usually presented in the frequency domain with Laplace transforms. Similarly in the discrete-time case, it is called the *pulse transfer function* model and is given by

$$H(z) = \frac{B(z^{-1})}{A(z^{-1})} \tag{2.90}$$

where $A$ and $B$ are polynomials in $z$ or $z^{-1}$. Thus

$$A(z^{-1}) = 1 + a_1 z^{-1} + \ldots + a_{N_a} z^{-N_a} \tag{2.91}$$

$$B(z^{-1}) = b_0 + b_1 z^{-1} + \ldots + b_{N_b} z^{-N_b} \tag{2.92}$$

If we consider the equivalent time domain representation, then we have a *difference equation* relating the output sequence $\{y(t)\}$ to the input sequence $\{u(t)\}$.[7] We use the backward shift operator $q$ with the property that $q^{-k}y(t) = y(t-k)$:

$$A(q^{-1})y(t) = B(q^{-1})u(t) \tag{2.93}$$

or

$$y(t) + a_1 y(t-1) + \ldots + a_{N_a} y(t-N_a) = b_0 u(t) + \ldots + b_{N_b} u(t-N_b) \tag{2.94}$$

When the system is excited by random inputs, the models are given by the *auto regressive-moving average model with exogeneous inputs* (*ARMAX*)[8]

$$\underbrace{A(q^{-1})y(t)}_{AR} = \underbrace{B(q^{-1})u(t)}_{X} + \underbrace{C(q^{-1})e(t)}_{MA} \tag{2.95}$$

where $A$, $B$, $C$, are polynomials, and $\{e(t)\}$ is a white noise source, and

$$C(q^{-1}) = 1 + c_1 q^{-1} + \cdots + c_{N_c} q^{-N_c}$$

The *ARMAX* model, usually abbreviated by $ARMAX(N_a, N_b, N_c)$ represents the general form for popular time-series and digital filter models, that is,

[7]We change from the common signal-processing convention of using $x(t)$ for the deterministic excitation to $u(t)$, and we include the $b_0$ coefficient for generality.

[8]The *ARMAX* model can be interpreted in terms of the Wold decomposition of stationary times series, which states that a time series can be decomposed into a predictable or deterministic component ($u(t)$) and nondeterministic or random component ($e(t)$)[11].

- Pulse Transfer Function or Infinite Impulse Response (*IIR*) model: $C(\cdot) = 0$, or $ARMAX(N_a, N_b, 0)$, that is,

$$A(q^{-1})y(t) = B(q^{-1})u(t)$$

- Finite Impulse Response (*FIR*) model: $A(\cdot) = 1, C(\cdot) = 0$, or $ARMAX(1, N_b, 0)$, that is,

$$y(t) = B(q^{-1})u(t)$$

- Autoregressive (*AR*) model: $B(\cdot) = 0, C(\cdot) = 1$, or $ARMAX(N_a, 0, 1)$, that is,

$$A(q^{-1})y(t) = e(t)$$

- Moving Average (*MA*) model: $A(\cdot) = 1, B(\cdot) = 0$, or $ARMAX(1, 0, N_c)$, that is

$$y(t) = C(q^{-1})e(t)$$

- Autoregressive–Moving Average (*ARMA*) model: $B(\cdot) = 0$, or $ARMAX(N_a, 0, N_c)$, that is,

$$A(q^{-1})y(t) = C(q^{-1})e(t)$$

- Autoregressive model with Exogeneous Input (*ARX*): $C(\cdot) = 1$, or $ARMAX(N_a, N_b, 1)$, that is,

$$A(q^{-1})y(t) = B(q^{-1})u(t) + e(t)$$

The *ARMAX* model is shown in Figure 2.7. *ARMAX* models can easily be used for signal processing purposes, since they are basically digital filters with known deterministic ($u(t)$) and random ($e(t)$) excitations. Consider the following example, of spectral shaping using the *ARMAX* model.

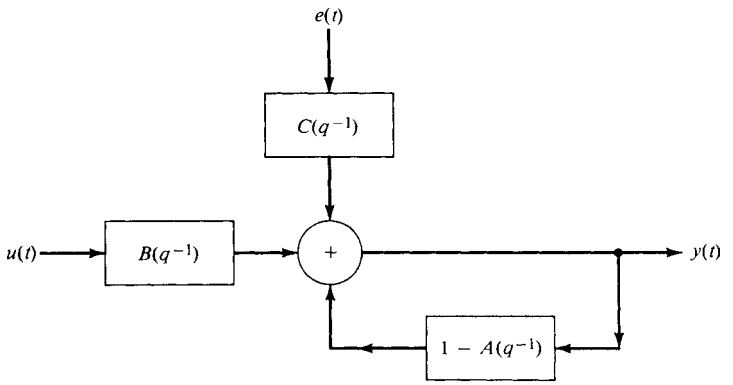



**Figure 2.7.** *ARMAX* input-output model.

***Example 2.14*** Let $a = -0.5$ in the previous example, $\{e(t)\}$ is $\mathcal{N}(0, 1)$. We are asked to generate the output sequence $y(t)$. We see that this can be accomplished using $ARMAX(1, 0, 0)$, an $AR$ model, that is,

$$H(z) = \frac{Y(z)}{E(z)} = \frac{1}{1 + 0.5z^{-1}}$$

Cross-multiplying and substituting the backward shift operation for $z$, we obtain the difference equation

$$y(t) = -0.5y(t-1) + e(t)$$

We use *MATLAB* to simulate these sequences and the results are shown in Figure 2.8. Note we show the estimated covariance and spectrum for this process as well.

So we see that the *ARMAX* can prove to be quite useful for simulation purposes. In fact, as we shall see subsequently, this model provides the basis for the parametric or modern approach to signal processing.

Since the *ARMAX* model is used to characterize a random signal, we are interested in its statistical properties. The mean value of the output is easily determined by

$$A(q^{-1})E\{y(t)\} = B(q^{-1})E\{u(t)\} + C(q^{-1})E\{e(t)\}$$

or

$$A(q^{-1})m_y(t) = B(q^{-1})u(t) + C(q^{-1})m_e(t) \tag{2.96}$$

Because the first term in the $A$-polynomial is unity, we can write the *mean propagation recursion* for the *ARMAX* model as

$$m_y(t) = (1 - A(q^{-1}))m_y(t) + B(q^{-1})u(t) + C(q^{-1})m_e(t) \tag{2.97}$$

$$m_y(t) = -\sum_{i=1}^{N_a} a_i m_y(t-i) + \sum_{i=0}^{N_b} b_i u(t-i) + \sum_{i=0}^{N_c} c_i m_e(t-i) \tag{2.98}$$

We note that the mean of the *ARMAX* model is propagated using a recursive digital filter requiring $N_a$, $N_b$, $N_c$ past input and output values.

The corresponding variance of the *ARMAX* model is more complex. First, we note that the mean must be removed, that is,

$$\begin{aligned} y(t) - m_y(t) = &\left[(1 - A(q^{-1}))y(t) + B(q^{-1})u(t) + C(q^{-1})e(t)\right] \\ &- \left[(1 - A(q^{-1}))m_y(t) + B(q^{-1})u(t) + C(q^{-1})m_e(t)\right] \end{aligned} \tag{2.99}$$

or

$$y(t) - m_y(t) = (1 - A(q^{-1}))(y(t) - m_y(t)) + C(q^{-1})(e(t) - m_e(t))$$

or finally

$$A(q^{-1})(y(t) - m_y(t)) = C(q^{-1})(e(t) - m_e(t)) \tag{2.100}$$

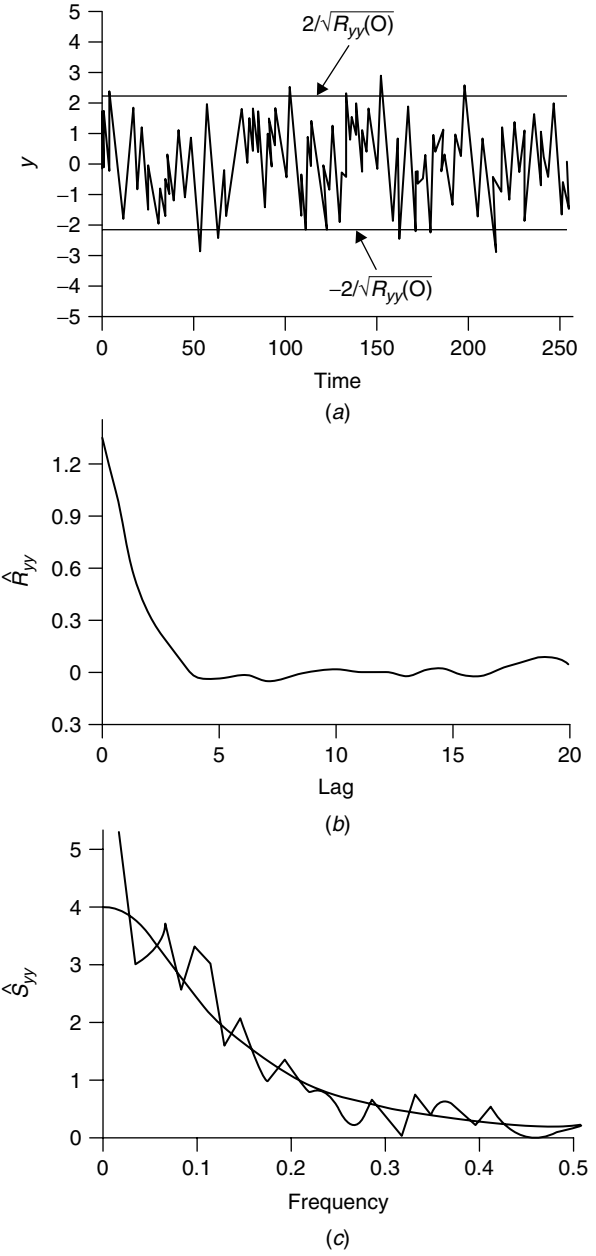



**Figure 2.8.** *ARMAX*(1, 0, 0) simulation for gaussian excitation.

that is, $y - m_y$ is characterized by an $ARMAX(N_a, 0, N_b)$ or equivalently an *ARMA* model.

The covariance of the *ARMAX* model can be calculated utilizing the fact that it is essentially and *IIR* system, that is,

$$\frac{Y(z)}{E(z)} = H(z) = \sum_{t=0}^{\infty} h(t) z^{-t} \tag{2.101}$$

Using this fact and the commutativity of the convolution operator, we have (assuming the mean has been removed)

$$R_{yy}(k) = E\{y(t)y(t+k)\} = E\left\{\sum_{i=0}^{\infty} h(i)e(t-i)\sum_{j=0}^{\infty} h(j+k)e(t-j+k)\right\}$$

or

$$R_{yy}(k) = \sum_{i=0}^{\infty} h(i)h(i+k)E\{e(t-i)e(t-i+k)\} + \sum_{i\neq j}\sum h(i)h(j+k)E\{e(t-i)e(t-j+k)\}$$

The whiteness of $\{e(t)\}$ gives

$$R_{ee}(k) = \begin{cases} R_{ee}, & k = 0 \\ 0, & \text{elsewhere} \end{cases}$$

Therefore, applying this property above, we have the *covariance* of the *ARMAX* model given by

$$R_{yy}(k) = R_{ee}\sum_{i=0}^{\infty} h(t)h(t+k) \qquad \text{for } k \geq 0 \tag{2.102}$$

with corresponding *variance*

$$R_{yy}(0) = R_{ee}\sum_{t=0}^{\infty} h^2(t) \tag{2.103}$$

We note that one property of a stationary signal is that its impulse response is bounded which implies from Eq. (2.102) that the variance is bounded [12]. Clearly, since the variance is characterized by an *ARMA* model ($ARMAX(N_a, 0, N_c)$), then we have

$$A(z^{-1})H(z) = C(z^{-1})E(z), \;\; E(z) = \sqrt{R_{ee}}$$

or taking the inverse $Z$-transform

$$h(t) = (1 - A(q^{-1}))h(t) + C(q^{-1})\delta(t)$$

or

$$h(t) = -\sum_{i=1}^{N_a} a_i h(t-i) + \sum_{i=0}^{N_c} c_i \delta(t), \quad c_0 = 1 \tag{2.104}$$

where $\delta(t)$ is an impulse of weight $\sqrt{R_{ee}}$. So we see that this recursion coupled with Eq. (2.102) provides a method for calculating the variance of an *ARMAX* model. We summarize these results in Table 2.4 and the following example.

**Table 2.4. *ARMAX* Representation**

Output propagation

$$y(t) = (1 - A(q^{-1}))y(t) + B(q^{-1})u(t) + C(q^{-1})e(t)$$

Mean propagation

$$m_y(t) = (1 - A(q^{-1}))m_y(t) + B(q^{-1})u(t) + C(q^{-1})m_e(t)$$

Impulse propagation

$$h(t) = (1 - A(q^{-1}))h(t) + C(q^{-1})\delta(t)$$

Variance/covariance propagation

$$R_{yy}(k) = R_{ee}\sum_{i=0}^{\infty} h(i)h(i+k), \qquad k \geq 0$$

where

$y$ = output or measurement sequence
$u$ = input sequence
$e$ = process (white) noise sequence with variance $R_{ee}$
$h$ = impulse response sequence
$\delta$ = impulse input of amplitude $\sqrt{R_{ee}}$
$m_y$ = mean output or measurement sequence
$m_e$ = mean process noise sequence
$R_{yy}$ = stationary output covariance at lag $k$
$A$ = $N_a$th-order system characteristic (poles) polynomial
$B$ = $N_b$th-order input (zeros) polynomial
$C$ = $N_c$th-order noise (zeros) polynomial

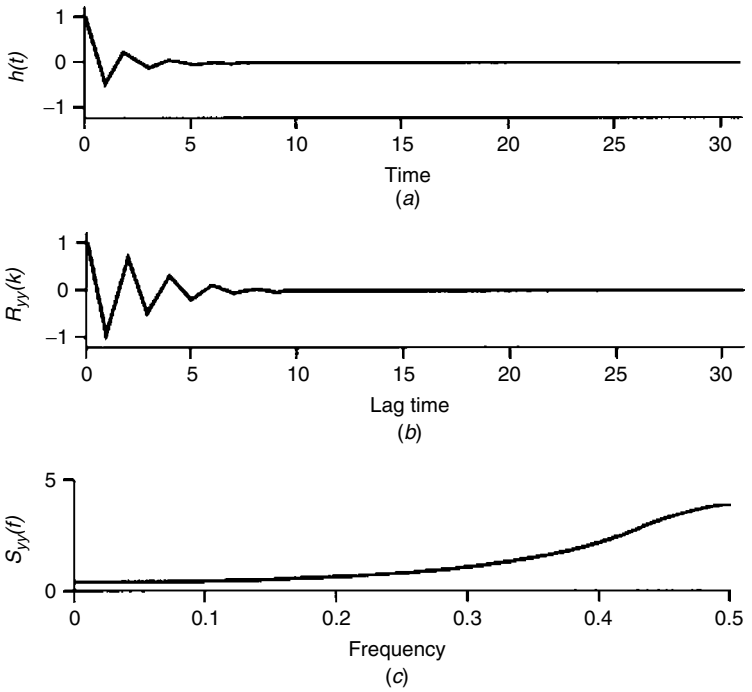



**Figure 2.9.** Autoregressive example. *(a)* Simulated impulse response. *(b)* Simulated covariance. *(c)* Simulated power spectrum.

***Example 2.15*** Suppose that we are given the *AR* model of the previous example with $A(z) = 1 + 0.5z^{-1}$, $R_{ee} = 1$ and would like to calculate the variance, then from Eq. (2.104), we have

$$h(t) = -0.5h(t-1) + \delta(t) \longrightarrow (-0.5)^t$$

and

$$R_{yy}(0) = \sum_{i=0}^{\infty} h^2(i) = [1^2 - 0.5^2 + .25^2 - 1.25^2 + .0875^2 - \cdots] \rightarrow 1.333$$

The covariance is calculated from Eq. (2.102) using *MATLAB* and is shown in Figure 2.9 along with the corresponding power spectrum and impulse response.

Before we leave this discussion let us consider a more complex example to illustrate the use of the *ARMA* model.

***Example 2.16*** Suppose that we would like to simulate an *ARMAX*(2, 1, 1) model with the following difference equation:

$$\left(1 + \frac{3}{4}q^{-1} + \frac{1}{8}q^{-2}\right) y(t) = \left(1 + \frac{1}{8}q^{-1}\right) u(t) + \left(1 + \frac{1}{16}q^{-1}\right) e(t)$$

where $u(t) = \sin\ 2\pi(0.025)t$ and $e \sim \mathcal{N}(1, 0.01)$. Then the corresponding mean propagation equation is

$$\left(1 + \frac{3}{4}q^{-1} + \frac{1}{8}q^{-2}\right) m_y(t) = \left(1 + \frac{1}{8}q^{-1}\right) u(t) + \left(1 + \frac{1}{16}q^{-1}\right) m_e(t)$$

for $m_e(t) = 1\ \forall\ t$ and the impulse propagation model is

$$\left(1 + \frac{3}{4}q^{-1} + \frac{1}{8}q^{-2}\right) h(t) = \left(1 + \frac{1}{16}q^{-1}\right) \sqrt{R_{ee}}\delta(t) \qquad \text{for } \sqrt{R_{ee}} = 0.1$$

and the variance/covariance propagation model is

$$R_{yy}(k) = R_{ee} \sum_{t=0}^{\infty} h(t)h(t+k), \qquad k \geq 0$$

Using *MATLAB* we perform the simulation with $\triangle T = 0.1$ second and the results are shown in Figure 2.10. In the figure we see the simulated process, mean (sinusoidal), and corresponding *ARMA* (mean removed) signals. The statistical relations are also shown with the corresponding impulse response and variance propagation models of Eqs. (2.102) and (2.103) and estimated power spectrum. This completes the example.

It should be noted that for certain special cases of the *ARMAX* model, it is particularly simple to calculate the mean and covariance. For instance, the *MA* model (*ARMAX*(1, 0, $N_c$)) has *mean*

$$m_y(t) = E\{C(q^{-1})e(t)\} = C(q^{-1})m_e(t) \tag{2.105}$$

and *covariance* (directly from Eq. (2.102) with $h \to c$)

$$R_{yy}(k) = R_{ee} \sum_{i=0}^{N_c} c_i c_{i+k} \qquad \text{for } k \geq 0 \tag{2.106}$$

We summarize these results for the *MA* model in Table 2.5.

Another special case of interest is the *AR* (*ARMAX*($N_a$, 0, 1)) model with *mean*

$$m_y(t) = (1 - A(q^{-1}))m_y(t) + m_e(t) \tag{2.107}$$

and *covariance* which is easily derived by direct substitution

$$\begin{aligned} R_{yy}(k) &= E\{y(t)y(t+k)\} = (1 - A(q^{-1}))R_{yy}(k) \\ &= -\sum_{i=1}^{N_a} a_i R_{yy}(k-i) \qquad \text{for } k > 0 \end{aligned} \tag{2.108}$$

In fact the *AR* covariance model of Eq. (2.108) is essentially a recursive (all-pole) digital filter which can be propagated by exciting it with the variance $R_{yy}(0)$ as

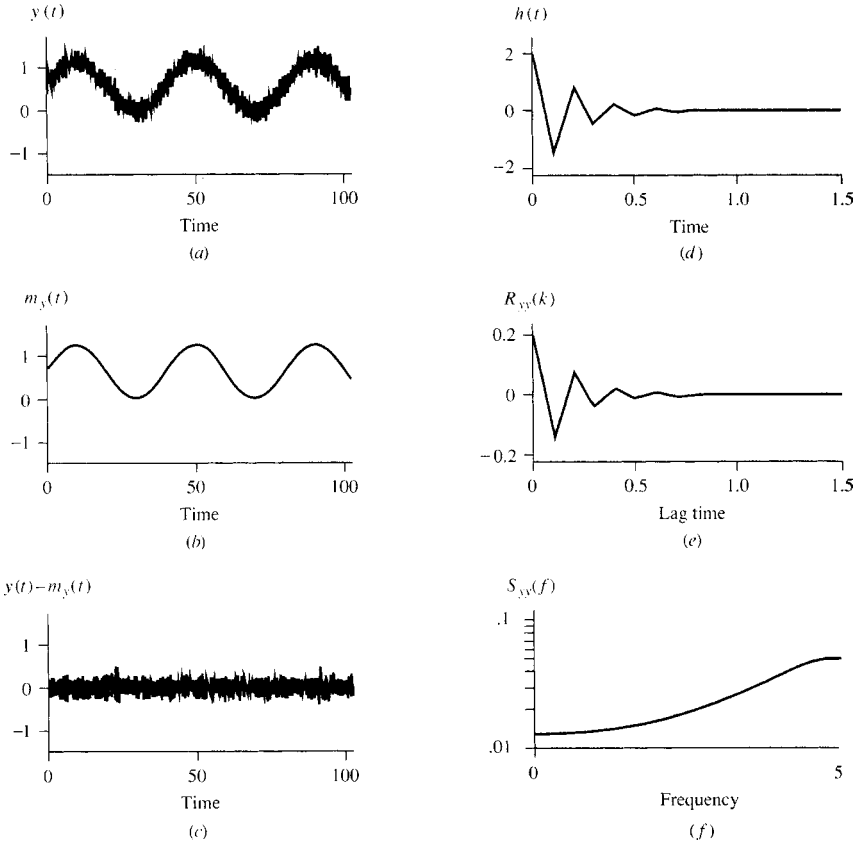



**Figure 2.10.** *ARMAX*(2, 1, 1) simulation. *(a)* Simulated signal. *(b)* Mean propagation. *(c) ARMA* (mean removed) signal. *(d)* Impulse response propagation. *(e)* Covariance propagation. *(f) PSD.*

initial condition. In this case the variance is given by

$$R_{yy}(0) = E\{y^2(t)\} = E\left\{\left(-\sum_{i=1}^{N_a} a_i y(t-i) + e(t)\right) y(t)\right\}$$
$$= -\sum_{i=1}^{N_a} a_i R_{yy}(i) + R_{ee} \tag{2.109}$$

So combining Eqs. (2.108) and (2.109), we have that the *covariance propagation* equations for the *AR* model are given by

$$R_{yy}(k) = \begin{cases} -\sum_{i=1}^{N_a} a_i R_{yy}(i) + R_{ee}, & k = 0 \\ -\sum_{i=1}^{N_a} a_i R_{yy}(k-i), & k > 0 \end{cases}$$

**Table 2.5. *MA* Representation**

Output propagation

$$y(t) = C(q^{-1})e(t)$$

Mean propagation

$$m_y(t) = C(q^{-1})m_e(t)$$

Variance/covariance propagation

$$R_{yy}(k) = R_{ee}\sum_{i=0}^{N_c} c_i c_{i+k}, \qquad k \geq 0$$

where

$y$ = output or measurement sequence
$e$ = process (white) noise sequence with variance $R_{ee}$
$m_y$ = mean output or measurement noise sequence
$m_e$ = mean process noise sequence
$R_{yy}$ = *stationary* output covariance at lag $k$
$C$ = $N_c$th-order noise (zeros) polynomial

We summarize these results for the *AR* model in Table 2.6.

Consider the following example of calculating the statistics of the *AR* model.

***Example 2.17*** Taking the *AR* model of the previous examples, suppose that we would like to determine the corresponding mean and variance using the recursions of Eqs. (2.107) and (2.108) with $A(q^{-1}) = 1 + 0.5q^{-1}$. The mean is

$$m_y(t) = -0.5m_y(t-1)$$

and the covariance is

$$R_{yy}(k) = \begin{cases} -0.5R_{yy}(1) + 1, & k = 0 \\ -0.5R_{yy}(k-1), & k > 0 \end{cases}$$

The variance is obtained directly from these recursions, since

$$R_{yy}(1) = -0.5R_{yy}(0)$$

Therefore

$$R_{yy}(0) = -0.5R_{yy}(1) + 1$$

Substituting for $R_{yy}(1)$, we obtain

$$R_{yy}(0) = -0.5(-0.5R_{yy}(0)) + 1$$

**Table 2.6. *AR* Representation**

Output propagation

$$y(t) = (1 - A(q^{-1}))y(t) + e(t)$$

Mean propagation

$$m_y(t) = (1 - A(q^{-1}))m_y(t) + m_e(t)$$

Variance/covariance propagation

$$R_{yy}(k) = (1 - A(q^{-1}))R_{yy}(k) + R_{ee}\delta(k), \qquad k \geq 0$$

where

$y$ = output or measurement sequence
$e$ = process (white) noise sequence with variance $R_{ee}$
$m_y$ = mean output or measurement sequence
$m_e$ = mean process noise sequence
$R_{yy}$ = *stationary* output covariance at lag $k$
$A$ = $N_a$th-order system characteristic (poles) polynomial
$\delta$ = Kronecker delta function

or

$$R_{yy}(0) = 1.333$$

as before. Using *MATLAB*, we simulated the covariance propagation equations and the results are depicted in Figure 2.11. The corresponding *PSD* is also shown.

This completes the section on *ARMAX* models. In the next section we develop an alternative but equivalent structure—the lattice model.

## 2.8 LATTICE MODELS

Lattice filters find their roots in analogue filter designs and in the design of digital filters as well ([13], [14]). They have also evolved as equivalent representations of random signals. Estimation filters based on the lattice design perform well because of their inherent orthogonality properties ([15]). The (scalar) *lattice model* is defined by the following fundamental propagation equation:

$$\begin{aligned} e_f(t,i) &= e_f(t,i-1) - k_i e_b(t-1,i-1) \\ e_b(t,i) &= e_b(t-1,i-1) - k_i e_f(t,i-1) \end{aligned} \tag{2.110}$$

where $e_f(t,i), e_b(t,i)$ are the respective forward and backward signals of the $i$th lattice section (or stage) at time $t$, and $k_i$ is the *reflection coefficient* of the $i$th section. The lattice model is depicted in Figure 2.12. This model is constructed by

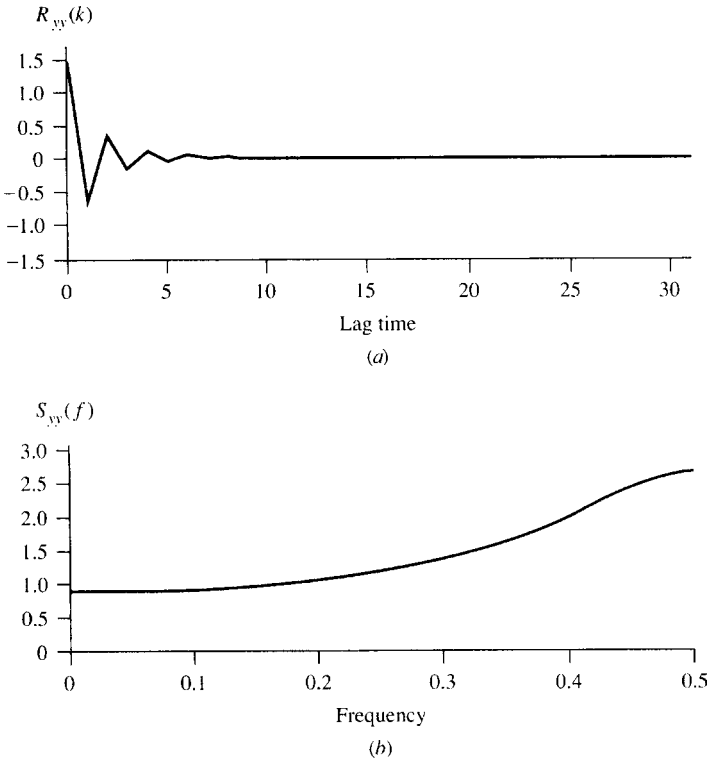



**Figure 2.11.** Autoregressive example. *(a)* Simulated covariance. *(b)* Simulated power spectrum.

a series of connected sections with each section having identical structure as in the figure, so only the values of the reflection coefficients change (see Figure 2.13).

The lattice model derives its physical origins from the facts that it mathematically represents a *wave model* or *wave* propagating through a layered medium. For instance, lattice models are employed to characterize seismic waves propagating; through a layered earth in the exploration of oil ([12]), or an acoustic wave propagating through a tube in speech synthesis ([16]), or an electromagnetic wave

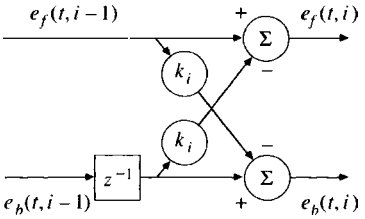



**Figure 2.12.** Lattice model.

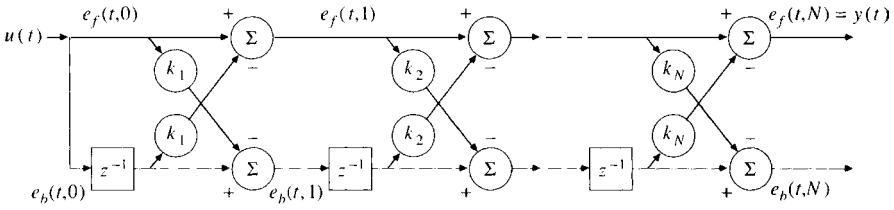



**Figure 2.13.** Feed-forward lattice (all-zero) structure.

propagating in a transmission line. In each of these cases we note that the reflection coefficient, which has magnitude between $\pm 1$, indicates that percentage of the wave transmitted and reflected at a layer boundary (or lattice stage).

Structurally the lattice model possesses some very attractive properties from the implementation viewpoint because of its sectional or stagewise symmetry. Consider a filter of length $N$ consisting of $N$ lattice sections arranged in a feed-forward configuration as shown in Figure 2.13. We first note that the lattice model operation for the $i$th section can be written in matrix form as using the backward shift operator as

$$\begin{bmatrix} e_f(t,i) \\ e_b(t,i) \end{bmatrix} = \begin{bmatrix} 1 & -k_i q^{-1} \\ -k_i & q^{-1} \end{bmatrix} \begin{bmatrix} e_f(t,i-1) \\ e_b(t,i-1) \end{bmatrix} \tag{2.111}$$

Taking the $Z$-transform of Eq. (2.111) gives

$$\begin{bmatrix} E_f(z,i) \\ E_b(z,i) \end{bmatrix} = \begin{bmatrix} 1 & -k_i z^{-1} \\ -k_i & z^{-1} \end{bmatrix} \begin{bmatrix} E_f(z,i-1) \\ E_b(z,i-1) \end{bmatrix} \tag{2.112}$$

or more compactly

$$\underline{E}(z,i) = L(z,i)\underline{E}(z,i-1) \tag{2.113}$$

where $L(z,i)$ can be thought of as a multichannel transfer function matrix[9] between the $i$th and $(i-1)$th section of the lattice. In fact, inverting $L(z,i)$ enables us to determine the inverse relation between the $i$th and the $(i-1)$th stage as

$$\underline{E}(z,i-1) = L^{-1}(z,i)\underline{E}(z,i) \tag{2.114}$$

where

$$L^{-1}(z,i) = \frac{1}{1-k_1^2} \begin{bmatrix} 1 & k_i \\ k_i z & z \end{bmatrix}$$

The lattice transfer function is useful for predicting the response of the lattice at various stages. The transfer function for the $N$-stage lattice can easily be derived

[9]We note that the lattice filter is a two-port network [13], and $L(z,i)$ is the so-called transfer matrix from input to output.

using Eq. (2.113) and recursing backward, that is,

$$\begin{aligned}\underline{E}(z, N) &= L(z, N)\underline{E}(z, N-1) \\ &= L(z, N)[L(z, N-1)\underline{E}(z, N-2)] \\ &= L(z, N)L(z, N-1)\ldots L(z, 1)\underline{E}(z, 0)\end{aligned} \tag{2.115}$$

which gives the stage-wise transfer function. If we start with $(N = 1)$ in Eq. (2.7) we can similarly derive relations for the inverse transfer function. Thus the overall *stage transfer function* of an $N$-stage lattice, and its inverse are given by

$$L(z) := L(z, N)L(z, N-1)\ldots L(z, 1) \tag{2.116}$$

$$L^{-1}(z) := L^{-1}(z, 1)L^{-1}(z, 2)\ldots L^{-1}(z, N) \tag{2.117}$$

These stage transfer functions must be related to our standard input-output models of the previous section. Before we explore these relations, consider the following example.

***Example 2.18*** We have a two-stage lattice $\{k_1\ k_2\}$ and would like to determine the overall transfer function $L(z)$ as well as the inverse transfer function $L^{-1}(z)$. From Eq. (2.116) we see that

$$L(z) = L(z, 2)L(z, 1) = \begin{bmatrix} 1 & -k_2 z^{-1} \\ -k_2 & z^{-1} \end{bmatrix}\begin{bmatrix} 1 & -k_1 z^{-1} \\ -k_1 & z^{-1} \end{bmatrix}$$

or

$$L(z) = \begin{bmatrix} 1 + k_1 k_2 z^{-1} & -k_1 z^{-1} - k_2 z^{-2} \\ -k_2 - k_1 z^{-1} & k_1 k_2 z^{-1} + z^{-2} \end{bmatrix}$$

The inverse transfer function is given by

$$L^{-1}(z) = L^{-1}(z, 1)L^{-1}(z, 2) = \frac{1}{(1-k_1^2)(1-k_2^2)}\begin{bmatrix} 1 + k_1 k_2 z & k_2 + k_1 z \\ k_1 z + k_2 z^2 & k_1 k_2 z + z^2 \end{bmatrix}$$

The overall lattice transfer function is related to the standard *IIR* and *FIR* transfer functions of the previous section. If we examine the *FIR*, *MA* or equivalently all-zero forms of the *ARMAX* model, that is, *ARMAX*$(1, N_b, 0)$ or *ARMAX*$(1, 0, N_c)$, then we can develop the required relations. The *FIR* model is characterized by the input-output difference equation

$$y(t) = B(q^{-1})u(t)$$

and equivalently,

$$H(z) = \frac{Y(z)}{U(z)} = B(z) \tag{2.118}$$

If we use a feed-forward lattice structure, that is,

$$e_f(t, 0) = e_b(t, 0) := u(t) \text{ and } e_f(t, N) := y(t)$$

which is equivalent to exciting both input ports and measuring the output, then Eq. (2.7) becomes

$$\underline{E}(z, N) = L(z, N) \ldots L(z, 1)\underline{E}(z, 0) = L(z) \begin{bmatrix} U(z) \\ U(z) \end{bmatrix} = L(z) \begin{bmatrix} 1 \\ 1 \end{bmatrix} U(z)$$

Alternatively, dividing both sides by $U(z)$, we obtain

$$\begin{bmatrix} \dfrac{Y(z)}{U(z)} \\ \dfrac{Y^R(z)}{U(z)} \end{bmatrix} = \begin{bmatrix} H(z) \\ H^R(z) \end{bmatrix} = \begin{bmatrix} B(z) \\ B^R(z) \end{bmatrix} = L(z) \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} L_{11}(z) + L_{12}(z) \\ L_{21}(z) + L_{22}(z) \end{bmatrix}$$

where $H^R(z)$ is the so-called $N$th-order *reverse polynomial* [14] given by

$$H^R(z) = z^{-N} H(z^{-1})$$

So we see that if we equate the coefficients $\{b_i\}$ of the all-zero transfer function $H(z)$ to the sum of the row elements of $L(z)$ resulting from the iteration of Eq. (2.116), then we obtain the $N$-stage recursion (usually called the Levinson recursion)

$$\begin{aligned} b(1, 1) &= -k_1 \\ b(i, i) &= -k_i \qquad \text{for} \quad i = 2, \ldots, N_b \\ b(j, i) &= b(j, i-1) - k_i b(i-j, i-1) \qquad \text{for} \quad j = 1, \ldots, i-1 \end{aligned}$$

where $H(z, i) = 1 + \sum_{j=1}^{N_b} b(j, i) z^{-j}$ and $H(z, N_b) = 1 + b(1, N_b)z^{-1} + \cdots + b(N_b, N_b)z^{-N_b}$. The *ARMAX* polynomial is equivalent to the final stage.

The all-zero *ARMAX* model can also be transformed to the lattice form by using the inverse filtering operation $L^{-1}(z)$ of Eqs. (2.7) and (2.116), which leads to the following recursion[10]:

$$\begin{aligned} k_i &= -b(i, i) \quad \text{for} \quad i = N_b, \ldots, 1 \\ b(j, i-1) &= \frac{b(j, i) + k_i b(i-j, i)}{1 - k_i^2}, \quad \text{for} \quad j = 1, \ldots, i-1 \end{aligned}$$

Here we again use the coefficient relations

$$\{b_i\} = \{b(i, N_b)\}$$

Consider the following example.

[10]It should be noted that this procedure is equivalent to the Jury or Schur-Cohn stability test for the roots of the polynomial lying inside the unit circle. In fact the necessary and sufficient condition for stability is that $|k_i| < 1$.

***Example 2.19*** Suppose that we are given the $ARMAX(1, 2, 0)$ model with $\{1, b_1, b_2\}$, and we would like to find the corresponding reflection coefficients $\{k_1, k_2\}$ using the recursions of Eq. (2.119) with $N_b = 2$. We have

$$\begin{aligned} i &= 2, \quad k_2 = -b(2,2) = -b_2 \\ b(1,1) &= \frac{b(1,2) + k_2 b(1,2)}{1 - k_2^2} = \frac{b_1(1 + k_2)}{1 - k_2^2} = \frac{b_1(1 - b_2)}{1 - b_2^2} = \frac{b_1}{1 - b_2} \\ k_1 &= -b(1,1) \end{aligned}$$

which completes the recursion.

The inverse of the feed-forward lattice is the feedback lattice shown in Figure 2.14. The feedback lattice is an all-pole transfer function, thus, if we apply an input signal to a feed-forward lattice and the result is applied to a feedback lattice, with identical coefficients, then the original signal will result. From Mason's rule of circuit theory we know that the reflection coefficients parameterize *both* the feedback and feed-forward lattices with appropriate changes in signal flow. Thus the conversion from the reflection coefficients to tapped delay line coefficients is identical for the all-zero or all-pole *ARMAX* models where the signal flow specifies which transfer function is implied. We summarize the recursions in Table 2.7. The final useful transformation of the *ARMAX* model is the pole-zero form, that is, converted from $ARMAX(N_a, N_b, 0)$ or $ARMAX(N_a, 0, N_b)$ to lattice models. The symmetric two multiplier rational lattice, pole-zero transfer function is shown in Figure 2.15 where the tap coefficients are given by $\{g_i\}$ and the pole-zero transfer function is given by

$$H(z) = H(z, N) = \frac{\sum_{j=0}^{N} b(j, N) z^{-j}}{1 + \sum_{j=1}^{N} a(j, N) z^{-j}} = \frac{\sum_{j=0}^{N} b_j z^{-j}}{1 + \sum_{j=1}^{N} a_j z^{-1}} = \frac{B(z)}{A(z)} \quad (2.119)$$

Using procedures similar to the all-zero case (see [14] for details), we have the recursion

$$\begin{aligned} k_i &= -a(i,i) \qquad \text{for } i = N, \ldots, 1 \\ g_i &= b(i,i) \\ a(j, i-1) &= \frac{a(j,i) + k_i a(i-j, i)}{1 - k_i^2} \qquad \text{for } j = 1, \ldots, i-1 \end{aligned}$$

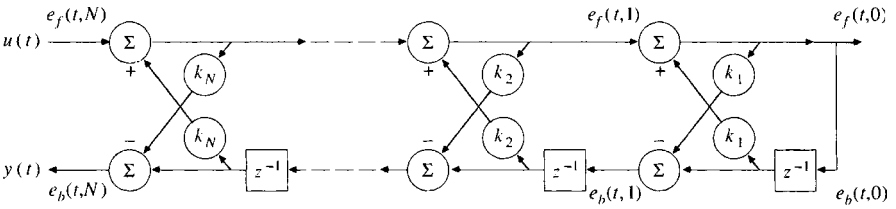



**Figure 2.14.** Feedback lattice (all-pole) structure.

**Table 2.7. *LATTICE* Recursions**

Lattice to all-pole or all-zero model

$$
\begin{aligned}
&b(1,1) = -k_1 \\
&b(i,i) = -k_i, && i = 2, \ldots, N_b \\
&b(j,i) = b(j,i-1) - k_i b(i-j,i-1), && j = 1, \ldots, i-1
\end{aligned}
$$

All-pole or all-zero to lattice model

$$
\begin{aligned}
&k_i = -b(i,i), && i = N_b, \ldots, 1 \\
&b(j,i-1) = \frac{b(j,i) + k_i b(i-j,i)}{1 - k_i^2}, && j = 1, \ldots, i-1
\end{aligned}
$$

where

$$H(z,k) = \sum_{i=0}^{k} b(i,k) z^{-k} \quad \text{and} \quad \{b_i\} = \{b(i,N_b)\}$$

$$
\begin{aligned}
b(j,i-1) &= b(j,i) - g_i a(i-j,i) \\
b(0,i-1) &= b(0,i) + g_i k_i \\
g_0 &= b(0,0)
\end{aligned} \tag{2.120}
$$

Although these coefficients are related in a nonlinear manner, this recursion is invertible so that the rational lattice structure can be converted uniquely to a direct-form, pole-zero filter, and vice versa.

***Example 2.20*** Consider a second-order transfer function with coefficients $\{b_0, b_1, b_2, a_1, a_2\}$, and we would like to calculate the corresponding lattice coefficients

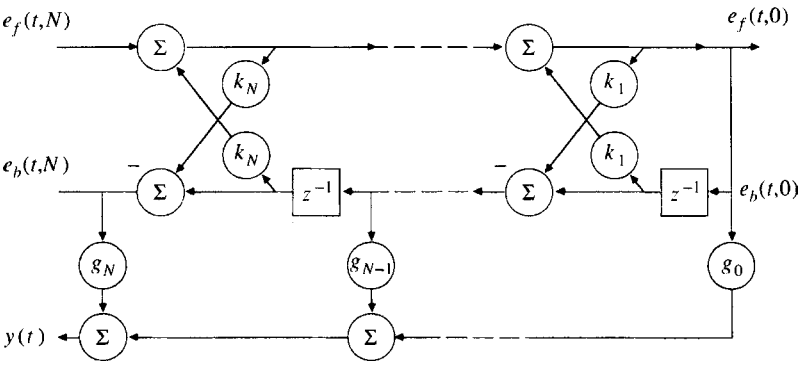



**Figure 2.15.** Rational lattice (pole-zero) structure.

$\{k_1, k_2, g_0, g_1, g_2\}$ using the recursion of Eq. (2.120) with $N = 2$:

$$
\begin{aligned}
i = 2, \quad & k_2 = -a(2,2) = -a_2 \\
& g_2 = b(2,2) = b_2 \\
j = 1, \quad & a(1,1) = \frac{a(1,2) + k_2 a(1,2)}{1 - k_2^2} = \frac{a_1(1+k_2)}{1-k_2^2} = \frac{a_1}{1-k_2} = \frac{a_1}{1+a_2} \\
& b(1,1) = b(1,2) - g_2 a(1,2) = b_1 - b_2 a_1 \\
& b(0,1) = b(0,2) + g_2 k_2 = b_0 - b_2 a_2 \\
i = 1, \quad & k_1 = -a(1,1) \\
& g_1 = b(1,1) \\
& b(0,0) = b(0,1) + g_1 k_1 = b_0 - b_1 \left( \frac{a_1}{1+a_2} \right) - b_2 \left( a_2 + \frac{a_1^2}{1+a_2} \right) = g_0
\end{aligned}
$$

which completes the example. We summarize the recursion in Table 2.8.

So we see that the lattice structure can be transformed to all-pole (*IIR*, *AR*) all-zero (*FIR*, *MA*), and pole-zero (*IIR*, *ARX*, *ARMA*) models, and vice versa. It should also be noted that in the stochastic case where we have various versions of the *ARMAX* model, the signal variance can be calculated directly from the white noise variance, $R_{ee}$, and knowledge of the reflection coefficients, that is ([17]),

$$
R_{yy}(0) = \text{var}\,\{y(t)\} = \frac{R_{ee}}{\prod_{i=1}^{N}(1 - k_i^2)} \tag{2.121}
$$

Compare this result with the recursions developed in the previous section for the variance of an *ARMAX* model.

**Table 2.8. Rational Transfer Function to *LATTICE* Transformation**

$k_i = -a(i,i), \qquad i = N, \ldots, 1$

$g_i = b(i,i)$

$a(j, i-1) = \dfrac{a(j,i) + k_i a(i-j,i)}{1 - k_i^2}, \quad j = 1, \ldots, i-1$

$b(j, i-1) = b(j,i) - g_i a(i-j,i)$

$b(0, i-1) = b(0,i) + g_i k_i$

$g_0 = b(0,0)$

where

$$
H(z,N) = \frac{\sum_{j=0}^{N} b(j,N) z^{-j}}{1 + \sum_{j=1}^{N} a(j,N) z^{-j}}
$$

$$
\{b_j\} = \{b(j,N)\}
$$

## 2.9 EXPONENTIAL (HARMONIC) MODELS

In this section we develop exponential-based models starting from a continuous-time system, since they evolve naturally from physical systems, leading directly to the discrete-time version through sampling theory. By "exponential based" models we mean starting with the complex exponential signal including the real damped sinusoidal representations as well as harmonic models which are all special cases. Once we complete this development, we discuss harmonic models as a special case of the exponential model set providing a good basis for the wave theory models to follow. We define the *continuous-time exponential model* as

$$x_t = \sum_{i=1}^{N_e} A_i e^{-p_i t} = \sum_{i=1}^{N_e} A_i e^{-(\sigma_i + j\omega_i)t} \tag{2.122}$$

for $A_i$ the complex amplitudes, $p_i = \sigma_i + j\omega_i$ the complex poles with respective damping and angular frequencies given by $\{\sigma_i, \omega_i\}$. Taking the Laplace transform of this expression gives the frequency domain representation

$$X(s) = \sum_{i=1}^{N_e} \frac{A_i}{s + p_i} = \sum_{i=1}^{N_e} \frac{A_i}{s + \sigma_i + j\omega_i} \tag{2.123}$$

Clearly, special cases of this model evolve when the poles occur in complex pairs leading to *damped sinusoidal models*, that is,

$$x_t = \sum_{i=1}^{N_e} A_i e^{-p_i t} + A_i^* e^{-p_i^* t} = \sum_{i=1}^{N_e} A_i e^{-(\sigma_i + j\omega_i)t} + A_i^* e^{-(\sigma_i - j\omega_i)t} \tag{2.124}$$

with the frequency domain equivalent

$$X(s) = \sum_{i=1}^{N_e} \frac{A_i}{s + \sigma_i + j\omega_i} + \frac{A_i^*}{s + \sigma_i - j\omega_i} = 2\sum_{i=1}^{N_e} \frac{ReA_i(s + \sigma_i) + ImA_i\omega_i}{(s + \sigma_i)^2 + \omega_i^2} \tag{2.125}$$

which gives the equivalent (real) representation

$$x_t = 2\sum_{i=1}^{N_e} ReA_i e^{-\sigma_i t} \cos\omega_i t + ImA_i e^{-\sigma_i t} \sin\omega_i t \tag{2.126}$$

Letting $\omega_i = 0$ gives the real damped exponential model

$$x_t = 2\sum_{i=1}^{N_e} Re\ A_i e^{-\sigma_i t} \tag{2.127}$$

and taking transforms gives the frequency domain representation

$$X(s) = 2\sum_{i=1}^{N_e} \frac{ReA_i}{s+\sigma_i} \tag{2.128}$$

Each of these representations are used to develop signal models for a wide variety of applications [18] that make them very useful in interpreting the physical signals. Next we develop the discrete-time equivalents.

Now, if we transform the continuous-time model to the discrete-time using the impulse invariant transformation (see Eq. 2.43), $z_i = e^{p_i \triangle T} = e^{(\sigma_i + j\omega_i)\triangle T}$ and $h_t = x_t$ in the model. Thus we have that $h_{k\triangle T} = h_t|_{t=k\triangle T}$. That is, if we sample the impulse response of Eq. (2.122), then we obtain

$$x(k\triangle T) = \sum_{i=1}^{N_e} A_i e^{-p_i k\triangle T} = \sum_{i=1}^{N_e} A_i e^{-(\sigma_i + j\omega_i)k\triangle T} = \sum_{i=1}^{N_e} \left(A_i e^{-\sigma_i k\triangle T}\right)\left(e^{-j\omega_i}\right)^{k\triangle T}$$

Simplifying, we obtain the *discrete-time damped exponential model*

$$x(t_k) = \sum_{i=1}^{N_e} \alpha_i(t_k)\beta_i^{t_k} \tag{2.129}$$

where $t_k = k\triangle T$, $\alpha_i(t_k) := A_i e^{-\sigma_i k\triangle T}$ and $\beta_i := e^{-j\omega_i}$. Thus we see that the generic exponential model evolves by sampling the continuous-time version and using the impulse invariant transformation. With the relationship between the continuous- and discrete-time exponential models established, let us examine some of the special discrete signal models that evolve from these representations. Note that we return to the usual discrete-time index notation of $t$ rather than the sampled index of $t_k$.

Consider the *discrete sinusoidal signal model* given by

$$y(t) = x(t) + n(t) = \sum_{i=1}^{N_s} A_i \sin(\Omega_i t + \phi_i) + n(t) \tag{2.130}$$

where $n$ is zero-mean, white noise with variance $R_{nn}$ and $\phi \sim \mathcal{U}(-\pi, \pi)$. The covariance of this signal can be calculated analytically (see Example 2.8) as

$$R_{yy}(k) = \sum_{i=1}^{N_s} \frac{A_i^2}{2} \cos k\triangle T\Omega_i + R_{nn}\delta(k) \tag{2.131}$$

or

$$R_{yy}(k) = \begin{cases} R_{nn} + \sum_{i=1}^{N_s} P_i, & k = 0 \\ \sum_{i=1}^{N_s} P_i \cos k\triangle T\Omega_i, & k \neq 0 \end{cases} \tag{2.132}$$

where $P_i = A_i^2/2$ and $\triangle T$ is the sampling interval.

The corresponding *PSD* is given by

$$S_{yy}(\Omega) = \triangle T \sum_{i=1}^{N_s} 2\pi P_i \left[\delta(\Omega + \Omega_i) + \delta(\Omega - \Omega_i)\right] + R_{nn} \tag{2.133}$$

So we see that the spectrum consists of $N_s$ spectral lines located at $\Omega_i = 2\pi f_i$ with power $P_i$ on a constant level of $R_{nn}$, and therefore the model is parameterized by $(\{P_i\}, \{\Omega_i\})$ to characterize the sinusoids.

We know from linear systems theory [19] that an individual sinusoid can be characterized by a second-order system or difference equation[11]

$$x(t) = -\alpha_1 x(t-1) - \alpha_2 x(t-2) \tag{2.134}$$

where $\alpha_1 = 2\cos\Omega_1$ and $\alpha_2 = 1$. In general, then $N_s$-sinusoids can be represented by the $2N_s$-order difference equation

$$x(t) = -\sum_{i=1}^{2N_s} \alpha_i x(t-i) \tag{2.135}$$

Equivalently we can use the sinusoidal model directly

$$y(t) = x(t) + n(t) = -\sum_{i=1}^{2N_s} \alpha_i x(t-i) + n(t) \tag{2.136}$$

If we substitute for $x(t-i)$ using Eq. (2.130 above), we have

$$y(t) = -\sum_{i=1}^{2N_s} \alpha_i (y(t-i) - n(t-i)) + n(t) \tag{2.137}$$

or

$$y(t) = -\sum_{i=1}^{2N_s} \alpha_i y(t-i) + \sum_{i=0}^{2N_s} \alpha_i n(t-i) \tag{2.138}$$

for $\alpha_0 = 1$ showing that this is a spectral $2N_s$th-order *ARMA* model.

When the *ARMA* parameters are specified it is well known [1] that they can be used to form the system characteristic equation whose roots are the required

[11]This evolves from the fact that the $Z$-transform of a sinusoid is

$$z[\sin\Omega_0 t] = \frac{z^{-1}\sin\Omega_0}{1 + 2\cos\Omega_0 z^{-1} + z^{-2}}$$

which gives the homogeneous solution.

sinusoidal frequencies, that is

$$z^{2N_s} + \alpha_1 z^{2N_s-1} + \cdots + \alpha_{2N_s} = \prod_{i=1}^{N_s}(z - z_i)(z - z_i^*) = 0 \tag{2.139}$$

where $z_i = e^{j\Omega_i \triangle T}$.

After determining the roots of the characteristic equation, the sinusoidal frequencies are found from the impulse invariant transformation

$$\Omega_i = \frac{1}{\triangle T} \ln z_i \tag{2.140}$$

Next we investigate a more complicated yet equivalent version of the sinusoidal model—the *harmonic model*. The sinusoidal model is a special case of this model. We define the *complex harmonic model* by

$$x(t) = \sum_{i=1}^{N_s} A_i e^{j\Omega_i t + \phi_i} \tag{2.141}$$

where $A_i$ is complex, $A_i = |A_i|e^{j\theta}$; $\phi_i$ is random and uniformly distributed as $\phi \sim \mathcal{U}(-\pi, \pi)$ and $\Omega_i$ is the harmonic frequency. The set of $N_s$-complex amplitudes and harmonic frequencies, $[\{A_i\}, \{|\Omega_i\}]$, $i = 1, \ldots, N_s$ are assumed to be deterministic, but unknown.

The corresponding *measurement model* is characterized by the complex harmonic signal in additive random noise

$$y(t) = x(t) + n(t) \tag{2.142}$$

with $n(t)$, a zero-mean, random noise sequence with variance, $\sigma_n^2$.

Statistically, assuming stationarity, we can characterize this model by its autocorrelation function as

$$R_{yy}(k) = E\{y(t)y^*(t+k)\} = R_{xx}(k) + R_{nn}(k) \tag{2.143}$$

since the signal is uncorrelated with the noise. The autocorrelation of the signal for a single harmonic is found by

$$R_{xx}(k) = E\left\{(A_i e^{j\Omega_i t + \phi_i)} A_i^* e^{-j\Omega_i (t+k) + \phi_i)}\right\} = E\left\{|A_i|^2 e^{-j\Omega_i k}\right\} \tag{2.144}$$

or simply

$$R_{xx}(k) = P_i e^{-j\Omega_i k} \text{ for } P_i := |A_i|^2 \tag{2.145}$$

Taking Fourier transforms, we obtain the corresponding harmonic power spectrum as

$$P_{xx}(\Omega) = P_i \delta(\Omega - \Omega_i) + \sigma_n^2 \tag{2.146}$$

an impulse or spectral line. Expanding this to the $N_s$-harmonics, we obtain the final expressions

$$R_{xx}(k) = \sum_{i=1}^{N_s} P_i e^{-j\Omega_i k} + \sigma_n^2 \delta(k) \tag{2.147}$$

or

$$R_{yy}(k) = \begin{cases} \sum_{i=1}^{N_s} P_i + \sigma_n^2, & k = 0 \\ \sum_{i=1}^{N_s} P_i e^{-j\Omega_i k}, & k \neq 0 \end{cases} \tag{2.148}$$

This completes the discussion and development of exponentially based models. Next we consider the set of models corresponding to propagating waves.

## 2.10 SPATIOTEMPORAL WAVE MODELS

Signals carried by propagating waves yield information about temporal events (time) happening some distance (space) from us. For example, thunder is propagated by acoustic waves received by our ears, while lightening is propagated by electromagnetic waves observed by our eyes. In both of these instances the signal of interest is characterized by variations in space and time, that is, the signal $s(\mathbf{z}; t)$, where $\mathbf{z} := (x, y, z)$, is the spatial location in cartesian coordinates and $t$ is time. Both $\mathbf{z}$ and $t$ can be continuous or discrete. The signal is usually called a *spatio-temporal* or *space-time* signal and methods applied for processing are called space-time signal processing techniques. The major applications of space-time signal processing techniques are in the areas of radar, sonar, seismology, radio astronomy and biomedical imaging. In sonar, seismology and imaging, the spatiotemporal signal is the pressure at a point in space and time propagating through the particular application medium, while in radar and radio astronomy the signal is an electromagnetic wave propagating in the atmosphere.

### 2.10.1 Plane Waves

The space-time signal, $s(\mathbf{z}; t)$, can be analyzed and processed just like any other signal, but we must take into account both its spatial and temporal properties. Typical analysis of a deterministic spatiotemporal signal is concerned with its spectral content leading to the Fourier transform (assuming continuous space-time) given by

$$s(\mathbf{z}; t) \longrightarrow S(\mathbf{k}; \omega)$$

or mathematically

$$S(\mathbf{k}; \omega) = \int\int s(\mathbf{z}; t)\ e^{-j(\omega t - \mathbf{k}\cdot\mathbf{z})}\ d\mathbf{z}\ dt \tag{2.149}$$

$S(\mathbf{k}; \omega)$ is called the *wavenumber-frequency spectrum* of the spatiotemporal signal with $\mathbf{k}$ and $\omega$ the corresponding *wavenumber* and temporal *frequency*.

Wave theory ([20],[21]) evolves from the underlying physics governing a specific phenomenology leading to a particular application (electromagnetics/radar, acoustics/sonar, etc.). Thus signals carried by waves propagating through a medium can be modeled mathematically as satisfying the classical *wave equation* characterized by the partial differential equation (*PDE*)

$$\nabla_z^2\ s(\mathbf{z};t) - \frac{1}{c^2}\nabla_t^2\ s(\mathbf{z};t) = 0 \tag{2.150}$$

for $c$ the propagation velocity (assumed constant) and $\nabla_z^2$ the Laplacian operator defined by $\nabla_z^2 := [\partial^2/\partial x^2\ \ \partial^2/\partial y^2\ \ \partial^2/\partial z^2]$. For certain coordinate systems, the wave equation can be solved using the *separation of variables* technique. If we expand the three-dimensional wave equation above in terms of its spatial (cartesian) coordinates and assume that the solution has the separable form, then we have

$$s(\mathbf{z};t) = s(x,y,z;t) = X(x)Y(y)Z(z)T(t) \tag{2.151}$$

Substituting into the wave equation and performing the appropriate partial derivatives, we obtain

$$\frac{\partial^2 X(x)}{\partial x^2}\ Y(y)Z(z)T(t) + \frac{\partial^2 Y(y)}{\partial y^2}\ X(x)Z(z)T(t) + \frac{\partial^2 Z(z)}{\partial z^2}\ X(x)Y(y)T(t)$$
$$= \frac{1}{c^2}\frac{\partial^2 T(t)}{\partial t^2}\ X(x)Y(y)Z(z) \tag{2.152}$$

Dividing through by Eq. (2.151), the assumed solution in factored form gives

$$\frac{1}{X(x)}\frac{\partial^2 X(x)}{\partial x^2} + \frac{1}{Y(y)}\frac{\partial^2 Y(y)}{\partial y^2} + \frac{1}{Z(z)}\frac{\partial^2 Z(z)}{\partial z^2} = \frac{1}{c^2}\frac{1}{T(t)}\frac{\partial^2 T(t)}{\partial t^2}$$

which implies that the last term is a constant, say $(\omega/c)^2$. Similarly we define a set of "separation constants" $\{\kappa_x^2, \kappa_y^2, \kappa_z^2\}$ for each of the respective spatial terms. Therefore we obtain the set of coupled ordinary differential equations (ODE) by solving for the highest derivative, that is,

$$\frac{d^2T(t)}{dt^2} = -\omega^2 T(t) \quad \text{and therefore} \quad T(t) \propto e^{\pm j\omega t}$$

Similarly we obtain

$$\frac{d^2X(x)}{dx^2} = -\kappa_x^2 X(x) \quad \text{and therefore} \quad X(x) \propto e^{\pm j\kappa_x x}$$
$$\frac{d^2Y(y)}{dy^2} = -\kappa_y^2 Y(y) \quad \text{and therefore} \quad Y(y) \propto e^{\pm j\kappa_y y}$$
$$\frac{d^2Z(z)}{dz^2} = -\kappa_z^2 Z(z) \quad \text{and therefore} \quad Z(z) \propto e^{\pm j\kappa_z z}$$

The ODEs are coupled through the *dispersion relation* that is obtained by substituting the separation constants back into Eq. (2.152) such that

$$\kappa_x^2 + \kappa_y^2 + \kappa_z^2 = \left(\frac{\omega}{c}\right)^2 \tag{2.153}$$

or constraining $z$ to be dependent gives

$$\kappa_z^2 = \left(\frac{\omega}{c}\right)^2 - \kappa_x^2 - \kappa_y^2$$

so that the solution is characterized by only three independent constants, $(\omega, \kappa_x, \kappa_y)$.

Thus the separated solution is of the form

$$s(\mathbf{z}; t) = e^{j(\omega t - \mathbf{k}\cdot\mathbf{z})} \tag{2.154}$$

which can be interpreted as a *plane wave* propagating in the $\mathbf{k}$-direction where the *wavenumber vector* is $\mathbf{k} = [\kappa_x \ \kappa_y \ \kappa_z]$ with $|\mathbf{k}| = \omega/c = 2\pi/\lambda$ for $\lambda$ defined as the *wavelength* [22]. The wavenumber vector is the number of cycles (radians) per length of the plane wave in the propagation direction; therefore it is the *spatial frequency variable* analogous to $\omega$ in the temporal frequency domain. One temporal period of the plane wave is $T = 2\pi/\omega$, which implies that the distance traveled by the wave in one period is given by its wavelength, $\lambda$. The term plane wave arises because at any instant of time the value of $s(\mathbf{k}; t)$ is the same at all points lying in a plane defined by $\mathbf{k} \cdot \mathbf{z} = \kappa_x x + \kappa_y y + \kappa_z z = constant$ and the planes of constant phase are perpendicular to $\mathbf{k}$ [23]. It is also interesting to note that the Fourier transform of a plane wave characterized by $(\mathbf{k}_o, \omega_o)$, that is, $s(\mathbf{z}, \ t) = e^{j(\omega_o t - \mathbf{k}_o\cdot\mathbf{z})}$ is

$$S(\mathbf{k}, \omega) = \iint \left[e^{j(\omega_o t - \mathbf{k}_o\cdot\mathbf{z})}\right] e^{-j(\omega t - \mathbf{k}\cdot\mathbf{z})} \, d\mathbf{z} \, dt = \delta(\mathbf{k} - \mathbf{k}_o)\delta(\omega - \omega_o) \tag{2.155}$$

a 4D delta function in wavenumber-frequency space at $(\mathbf{k} - \mathbf{k}_o, \omega - \omega_o)$ [22]. Thus, the plane wave is the space-time equivalent of a harmonic (sinusoidal) signal in the temporal domain.

The plane wave solution can be expressed in a variety of different forms depending on the set of equivalent parameters selected, that is, Eq. (2.154) can be written as

$$s(\mathbf{z}; t) = e^{j(\omega_o t - \mathbf{k}_o\cdot\mathbf{z})} = e^{j\omega_o(t - \alpha_o\cdot\mathbf{z})} = e^{j\omega_o(t - \tau_o)} \tag{2.156}$$

where $\alpha_o$ is called the *slowness* vector since it is related to the wavenumber by

$$\alpha_o := \frac{\mathbf{k}_o}{\omega_o} \qquad \text{and} \qquad |\alpha_o| = \frac{|\mathbf{k}_o|}{\omega_o} = \frac{1}{c_o} \tag{2.157}$$

and is the reciprocal of the propagation velocity in the propagation direction, $\alpha_o$ [24]. The propagation *time delay* can be expressed in terms of the slowness or

equivalently wavenumber as

$$\tau_o = \alpha_o \cdot \mathbf{z} \qquad \text{or} \qquad \omega_o \tau_o = \mathbf{k}_o \cdot \mathbf{z} \tag{2.158}$$

Consider the following 2D example of a plane wave.

***Example 2.21*** A 2D plane wave propagating in the $XY$-plane makes an angle of incidence of $\theta$ degrees with the vertical axis (see Figure 2.16$a$). Suppose that we would like to determine the components of the direction vector, $\mathbf{k}$. The plane wave is

$$s(\mathbf{k}; t) = e^{j(\omega t - \mathbf{k}\cdot\mathbf{z})} = e^{j\left(\omega t - \begin{bmatrix} \kappa_x & \kappa_y \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}\right)} = e^{j(\omega t - \kappa_x x - \kappa_y y)}$$

but from geometry we have that

$$\kappa_x = |\mathbf{k}| \sin\theta \qquad \text{and} \qquad \kappa_y = |\mathbf{k}| \cos\theta$$

where

$$|\mathbf{k}| = \sqrt{\kappa_x^2 + \kappa_y^2} = \frac{\omega}{c} = \frac{2\pi}{\lambda}$$

Therefore the direction vector is given by

$$\mathbf{k} = [|\mathbf{k}| \sin\theta \quad |\mathbf{k}| \cos\theta]$$

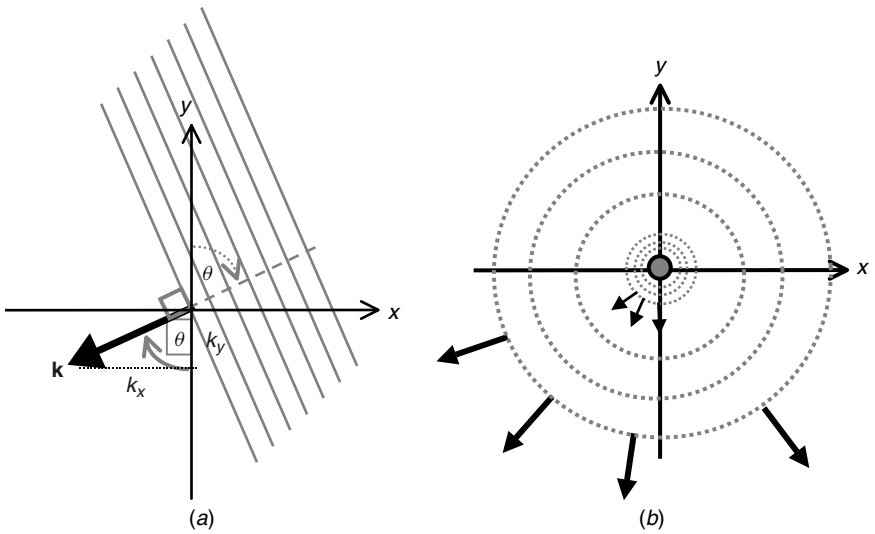



**Figure 2.16.** 2D wavefronts impinging on line array: (*a*) Planar. (*b*) Spherical.

and the plane wave model can be expressed in terms of the angle of incidence as

$$s(\mathbf{k};t) = e^{j(\omega t - x|\mathbf{k}|\sin\theta - y|\mathbf{k}|\cos\theta)} = e^{j\omega\left(t - \frac{x\sin\theta + y\cos\theta}{c}\right)}$$

Note also that to determine the *direction of arrival* (DOA) of a plane wave, we can determine either its wavenumber vector or corresponding angle of incidence, since this example shows their relationship (in 2D) [23].

### 2.10.2 Spherical Waves

Next let us consider the wave equation in spherical coordinates, $(r, \theta, \phi)$. Here the following coordinate transformations are applied to Eq. (2.150):

$$\begin{aligned} x &= r\sin\theta\cos\phi \\ y &= r\sin\theta\sin\phi \\ z &= r\cos\theta \end{aligned} \tag{2.159}$$

yielding the 3D *wave equation* in spherical coordinates

$$\nabla_{\mathbf{r}}^2 s(\mathbf{r};t) = \frac{1}{c^2}\nabla_t^2 s(\mathbf{r};t) \tag{2.160}$$

The Laplacian in this case is given by

$$\nabla_r^2 = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial}{\partial r}\right) + \frac{1}{r^2\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial}{\partial\theta}\right) + \frac{1}{r^2\sin^2\theta}\frac{\partial^2}{\partial\phi^2} \tag{2.161}$$

which can again be solved by the separation of variables, although complicated ([20], [23], [24]). However, if spherical symmetry exists, then $s(\mathbf{r};t)$ does not depend on $\theta$ and $\phi$, and the *spherical wave equation* evolves as

$$\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial}{\partial r}s(\mathbf{r};t)\right) = \frac{1}{c^2}\frac{\partial^2}{\partial t^2}s(\mathbf{r};t) \tag{2.162}$$

Thus admits the solution

$$s(\mathbf{r};t) = \frac{1}{|r|}e^{j(\omega t - \mathbf{k}_r\cdot\mathbf{r})} = \frac{1}{|r|}e^{j\omega\left(t - \frac{|\mathbf{r}|}{c}\right)} \tag{2.163}$$

The relations above can be interpreted as a spherical wave propagating outward from the origin (see Figure 2.16*b*) having a temporal frequency of $\omega$ and a spatial frequency or wavenumber $\mathbf{k}_r$ with corresponding *dispersion relation*

$$|\mathbf{k}_r|^2 = \left(\frac{\omega}{c}\right)^2 \tag{2.164}$$

It is easy to show from the above relations that the distance from $r$ to $r_i$ traveled by the wavefront is related in cartesian coordinates by

$$|r - r_i| = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2} \tag{2.165}$$

Finally, the Fourier transform of a spherical wave at $(\mathbf{k}_o, \omega_o)$ is given by

$$\begin{aligned} S(\mathbf{k}_r; \omega) &= \iint \left[e^{j(\omega_o t - \mathbf{k}_o \cdot \mathbf{r}_o)}\right] e^{-j(\omega t - \mathbf{k}_r \cdot \mathbf{r})}\, d\mathbf{z}\, dt \\ &= \left[\frac{2\pi^2}{j|\mathbf{k}_r|}\delta(|\mathbf{k}_r| - k_o) + \frac{4\pi}{|\mathbf{k}_r|^2 - k_o^2}\right]\delta(\omega - \omega_o)e^{-j\mathbf{k}_r \cdot r_o} \end{aligned} \tag{2.166}$$

Consider the application of an $L$-element sensor array (sonar, radar, seismic, etc.) sampling the propagated field. In this case $\mathbf{z}$ is a set of discrete samples, $\{\mathbf{z}_\ell\}$ for $\ell = 1, \ldots, L$. The signal received by the array, $s(\mathbf{z}; t)$, is characterized by a time delay, that is,

$$\mathbf{y}(t) = \begin{bmatrix} s\,(t - \tau_1(\Theta)) \\ \vdots \\ s\,(t - \tau_L(\Theta)) \end{bmatrix} \tag{2.167}$$

where the set of relative *delay times* $\{\tau_i(\Theta)\}$, $i = 1, \ldots, L$ represent the various distances the wavefield travels to each sensor. If the incident wavefield arrives from a *distant* source, then the associated *far-field* wavefield is classified as *planar* with the parameters, $\Theta$ describing the azimuthal and elevation angles as

$$\tau(\Theta) = \frac{1}{c}[x_i \cos\theta \cos\phi + y_i \sin\theta \cos\phi + z_i \sin\phi]$$

for $(x_i, y_i, z_i)$ the respective coordinates of the $i$th array element and $\theta$, $\phi$ the respective azimuth and elevation of the source. On the other hand, if field arrives from a source that is close, then the associated *near-field* wavefield is classified as *spherical* with the parameters describing its location given by

$$\tau(\Theta) = \sqrt{(x_i - x_s)^2 + (y_i - y_s)^2 + (z_i - z_s)^2}$$

with the corresponding delay times representing the time required for the wavefield to propagate from the source location, $(x_s, y_s, z_s)$ to the sensor element locations. So we see that the overall nature of the wavefield is identified by the particular propagating wave as either planar (far-field) or spherical (near-field), and both can be represented by the parameterized time delays. This idea then leads us to the generic wavefield model described next.

Before we introduce the generic model, however, there are some important concepts that must be developed to understand the approximations employed in the final structure. The first is that of a complex bandpass or *analytic signal* characterized by both real and imaginary parts and a Fourier transform identically zero

for negative frequencies. These signals commonly occur in radar and sonar applications. The complex representation of a bandpass signal is defined in terms of its *Hilbert* transform [1] as

$$\tilde{s}(t) = s(t) + j\overline{s}(t) \tag{2.168}$$

with $\overline{s}$ denoting the Hilbert transform of $s(t)$ defined by

$$\overline{s}(t) := \frac{1}{\pi}\int_{-\infty}^{\infty}\frac{s(\beta)}{t-\beta}d\beta$$

This transform is important when we define a bandpass signal as narrowband. We assume that the bandpass signal has been multiplied or modulated (frequency shifted as in amplitude modulation [1]) to a center frequency, $\omega_o$, that is,

$$s(t) = \alpha(t)\cos(\omega_o t + \phi(t)) \tag{2.169}$$

where both $\alpha$ and $\phi$ are *slowly varying* signals that modulate the amplitude and phase of $s(t)$. Taking the Hilbert transform of this *narrowband signal* representation gives

$$\overline{s}(t) = \alpha(t)\sin(\omega_o t + \phi(t)) \tag{2.170}$$

which is merely a $90^\circ$ *phase shift* of Eq. (2.169). Substituting these relations into Eq. (2.168) gives the complex *narrowband signal* representation

$$\tilde{s}(t) = \alpha(t)[\cos(\omega_o t + \phi(t)) + j\sin(\omega_o t + \phi(t))] = \alpha(t)e^{j(\omega_o t + \phi(t))} \tag{2.171}$$

This relation is the key representation used in many radar, sonar, harmonic, pole and echo estimation problems [25]. Using this characterization of a narrowband signal, we are now ready to develop the generic wavefield model.

### 2.10.3 Spatiotemporal Wave Model

The basic spatiotemporal *wave model* is defined by [25]

$$\tilde{y}_\ell(t) = \sum_{n=1}^{N_s}\alpha_\ell(\mathbf{k}_n)\tilde{s}_n(t - \tau_\ell(\mathbf{k}_n)) + \tilde{\eta}_\ell(t) \tag{2.172}$$

where

$\tilde{s}_n(t)$ = temporal signal of the $n$th wavefront observed at the $\ell$th sensor element;
$\alpha_\ell(\mathbf{k}_n)$ = wavefront amplitude at the $\ell$th sensor arriving from the $\mathbf{k}_n$-direction
$\tau_\ell(\mathbf{k}_n)$ = propagation delay between the $\ell$th sensor and $n$th wavefront arrival
$\tilde{\eta}_\ell(t)$ = additive random noise at the $\ell$th sensor

This model is a general representation of a broadband, space-time wave that can be simplified and made even more useful under the following sequence of assumptions. If we first assume that the process is demodulated to baseband at a center frequency of $\omega_o$ [23] by multiplication (modulation) of a complex exponential, $e^{j\omega_o t}$ followed by low-pass filtering of the upper frequency bands, then the wave model becomes

$$y_\ell(t) = \sum_{n=1}^{N_s} \alpha_\ell(\mathbf{k}_n) s_n\,(t - \tau_\ell(\mathbf{k}_n)) + \eta_\ell(t) \tag{2.173}$$

where $y_\ell$, $s_n$, $\eta$ are the complex, low-pass filtered versions of $\tilde{y}_\ell$, $\tilde{s}_n$, $\tilde{\eta}$.

Next assuming that the low-pass signal, $s(t)$, changes very slowly in time as the wavefront impinges across the array. That is, it is essentially assumed constant over any time interval less than or equal to the maximum array delay ($\triangle T \leq \tau_{\max}$). Under these conditions the signal $s(t)$ is defined as *narrowband* (see above) and therefore

$$s_n\,(t - \tau_\ell(\mathbf{k}_n)) \approx s_n(t) e^{j\omega_o \tau_\ell(\mathbf{k}_n)} \qquad \text{[Narrowband approximation]} \tag{2.174}$$

The generic *narrowband wave model* then becomes

$$y_\ell(t) = \sum_{n=1}^{N_s} \alpha_\ell(\mathbf{k}_n) s_n\,(t)\, e^{j\omega_o \tau_\ell(\mathbf{k}_n)} + \eta_\ell(t) \tag{2.175}$$

If we expand this expression across the array, then we obtain the vector relationships

$$\mathbf{y}(t) = \sum_{n=1}^{N_s} \mathbf{d}(\mathbf{k}_n) s_n\,(t) + \eta(t) \tag{2.176}$$

where we define the *direction* vector, $\mathbf{d}(\mathbf{k}_n) \in \mathcal{C}^{L\times 1}$ with

$$\mathbf{d}'(\mathbf{k}_n) := \left[\alpha_1(\mathbf{k}_n) e^{j\omega_o \tau_1(\mathbf{k}_n)} \cdots \alpha_L(\mathbf{k}_n) e^{j\omega_o \tau_L(\mathbf{k}_n)}\right] \tag{2.177}$$

Finally, expanding this expression over the number of source signals impinging on the array, we obtain the *narrowband, spatiotemporal wave model*

$$\mathbf{y}(t) = \mathbf{D}(\mathbf{k})\mathbf{s}\,(t) + \eta(t) \tag{2.178}$$

for $\mathbf{D}(\mathbf{k}) \in \mathcal{C}^{L\times N_s}$ the *direction matrix* defined by

$$\mathbf{D}(\mathbf{k}) = \begin{bmatrix} \alpha_1(\mathbf{k}_1) e^{j\omega_o \tau_1(\mathbf{k}_1)} & \cdots & \alpha_1(\mathbf{k}_{N_s}) e^{j\omega_o \tau_1(\mathbf{k}_{N_s})} \\ \vdots & & \vdots \\ \alpha_L(\mathbf{k}_1) e^{j\omega_o \tau_L(\mathbf{k}_1)} & \cdots & \alpha_L(\mathbf{k}_{N_s}) e^{j\omega_o \tau_L(\mathbf{k}_{N_s})} \end{bmatrix} \tag{2.179}$$

Assuming that the processes are zero-mean, wide-sense stationary, we take expectations and obtain the corresponding $L \times L$ *measurement covariance matrix*

$$\mathbf{R}_{yy} = E\{\mathbf{y}(t)\mathbf{y}^{\dagger}(t)\} = \mathbf{D}(\mathbf{k})\mathbf{R}_{ss}\mathbf{D}^{\dagger}(\mathbf{k}) + \mathbf{R}_{\eta\eta} \tag{2.180}$$

where $\mathbf{R}_{ss}$ is the $N_s \times N_s$ *signal covariance matrix* and $\mathbf{R}_{\eta\eta}$ is the $L \times L$ *noise covariance matrix*. Note also that this model captures both the plane wave and spherical wave cases by the appropriate definition of the time delay and attenuation, that is,

$$\tau_{\text{planar}}(\mathbf{k}) = \mathbf{k} \cdot \mathbf{z} \qquad \text{or} \qquad \tau_{\text{spherical}}(\mathbf{k}) = \mathbf{k}_r \cdot \mathbf{r}$$

$$\alpha_{\text{planar}}(\mathbf{k}) \quad = 1 \qquad \text{or} \qquad \alpha_{\text{spherical}}(\mathbf{k}) = \frac{1}{|\mathbf{r}|}$$

Consider the following example of developing the model for both cases.

***Example 2.22*** Using the narrowband model of Eq. (2.178), develop the output model of a three-element sensor array in two dimensions for two planar and then two spherical wave sources. The direction matrix for each case is different, since the time delays and attenuations are parameters. Starting with the planar case, we have that

$$\mathbf{D}_{\text{planar}}(\mathbf{k}) = \begin{bmatrix} e^{j\mathbf{k}_1\cdot\mathbf{z}_1} & e^{j\mathbf{k}_2\cdot\mathbf{z}_1} \\ e^{j\mathbf{k}_1\cdot\mathbf{z}_2} & e^{j\mathbf{k}_2\cdot\mathbf{z}_2} \\ e^{j\mathbf{k}_1\cdot\mathbf{z}_3} & e^{j\mathbf{k}_2\cdot\mathbf{z}_3} \end{bmatrix}$$

Alternatively, using the fact that in 2D the wavenumber vector is $\mathbf{k}_n = [|\mathbf{k}| \sin\theta_n \; |\mathbf{k}| \cos\theta_n]$, we have

$$\mathbf{D}_{\text{planar}}(\mathbf{k}) = \begin{bmatrix} e^{j(|\mathbf{k}|\sin\theta_1 x_1 + |\mathbf{k}|\cos\theta_1 y_1)} & e^{j(|\mathbf{k}|\sin\theta_2 x_1 + |\mathbf{k}|\cos\theta_2 y_1)} \\ e^{j(|\mathbf{k}|\sin\theta_1 x_2 + |\mathbf{k}|\cos\theta_1 y_2)} & e^{j(|\mathbf{k}|\sin\theta_2 x_2 + |\mathbf{k}|\cos\theta_2 y_2)} \\ e^{j(|\mathbf{k}|\sin\theta_1 x_3 + |\mathbf{k}|\cos\theta_1 y_3)} & e^{j(|\mathbf{k}|\sin\theta_2 x_3 + |\mathbf{k}|\cos\theta_2 y_3)} \end{bmatrix}$$

For a spherical wavefront we have that the wavenumber vector is $\mathbf{k}_r(n) = [\mathbf{k}_r^x(n) \;\; \mathbf{k}_r^y(n)]$. Therefore

$$\mathbf{D}_{\text{spherical}}(\mathbf{k}) = \begin{bmatrix} \dfrac{1}{|\mathbf{r}_1|} e^{j\mathbf{k}_r^x(1)\cdot\mathbf{r}_1} & \dfrac{1}{|\mathbf{r}_1|} e^{j\mathbf{k}_r^y(2)\cdot\mathbf{r}_1} \\ \dfrac{1}{|\mathbf{r}_2|} e^{j\mathbf{k}_r^x(1)\cdot\mathbf{r}_2} & \dfrac{1}{|\mathbf{r}_2|} e^{j\mathbf{k}_r^y(2)\cdot\mathbf{r}_2} \\ \dfrac{1}{|\mathbf{r}_3|} e^{j\mathbf{k}_r^x(1)\cdot\mathbf{r}_3} & \dfrac{1}{|\mathbf{r}_3|} e^{j\mathbf{k}_r^y(2)\cdot\mathbf{r}_3} \end{bmatrix}$$

which can be expanded as in the planar case with $\mathbf{r}_i = \sqrt{(x_\ell - x_i)^2 + (y_\ell - y_i)^2}$ for $\ell = 1, 2, 3$.

Note that the narrowband model of Eq. (2.178) can also be expressed equivalently in the temporal frequency domain by taking Fourier transforms to obtain the relation

$$\mathbf{Y}(\omega_k) = \mathbf{D}(\mathbf{k})\mathbf{S}(\omega_k) + \mathbf{N}(\omega_k) \tag{2.181}$$

which gives the corresponding *spectral covariance matrix* [23]

$$\mathbf{R}_{yy}(\omega_k) = E\{\mathbf{Y}(\omega_k)\mathbf{Y}^{\dagger}(\omega_k)\} = \mathbf{D}(\mathbf{k})\mathbf{R}_{ss}(\omega_k)\mathbf{D}^{\dagger}(\mathbf{k}) + \mathbf{R}_{NN}(\omega_k) \tag{2.182}$$

This completes the section on the wave models, which will be expanded and used in later chapters.

## 2.11 STATE-SPACE MODELS

In addition to the *ARMAX*, *lattice* and *wave* models of the previous section, we introduce the state-space model as an alternative representation of stochastic processes. As we stated previously, state-space models are easily generalized to multichannel, nonstationary, and in fact nonlinear processes. The state-space models are very popular for model-based signal processing primarily because most physical phenomena modeled by mathematical relations naturally occur in state-space form (see [26] for details).

### 2.11.1 Continuous State-Space Models

Intuitively, the state-space model of a dynamic system is probably the most natural and common form of mathematical representation of physical systems available. In other words, when we begin to develop a mathematical model of a physical process and write down a set of differential equations to describe the underlying dynamics, we have just performed the "first" step in obtaining the state-space representation. For ease of development, let us assume that we have a set of $n$th-order ordinary differential equations (ODE) with constant coefficients in continuous-time ($t \in \mathcal{R}$), that is,

$$\frac{d^n}{dt^n}z_t + a_{n-1}\frac{d^{n-1}}{dt^{n-1}}z_t + \cdots + a_1\frac{d}{dt}z_t + a_0 z_t = u_t$$

where $u_t$ is the input or excitation function.

The usual method of solving these equations, computationally, is to rewrite them into an equivalent set of $n$ first-order differential equations by defining a vector with components corresponding to the differentials, that is, $x_i := d^i z_t/dt^i$ for $i = 0, \ldots, n-1$ or

$$x_t := \left[ z_t \;\; \frac{dz_t}{dt} \;\; \cdots \;\; \frac{d^{n-1}z_t}{dt^{n-1}} \right]'$$

Taking the derivative of $x_t$ for each component, we obtain

$$\begin{aligned}
\dot{x}_1 &= \frac{dz_t}{dt} = x_2 \\
\dot{x}_2 &= \frac{d^2 z_t}{dt^2} = x_3 \\
&\vdots \\
\dot{x}_n &= \frac{d^n z_t}{dt^n} = -a_{n-1}\frac{d^{n-1} z_t}{dt^{n-1}} + \cdots - a_1 \frac{dz_t}{dt} - a_0 z_t + u_t
\end{aligned}$$

Collecting all of these equations and writing them into vector-matrix form, we obtain

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 \\ -a_0 & -a_1 & \cdots & -a_{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} u_t$$

or in compact form

$$\dot{x}_t = Ax_t + Bu_t$$

for $x_t \in \mathcal{R}^{n\times 1}$, $u_t \in \mathcal{R}^{1\times 1}$, $A \in \mathcal{R}^{n\times n}$, and $B \in \mathcal{R}^{n\times 1}$. Here $x_t$ is defined as the *state vector*, $u_t$ the input, $A$ is the system or process matrix and $B$ is the input matrix. So we see from this simplistic view, the state vector is "naturally" obtained when we convert an $n$th order ODE to a set of $n$ first-order equations.

Next we develop a set of relations to "measure" the state variables when they are accessible (e.g., voltage across a capacitor or displacement of a swaying bridge). States are usually called the "internal" or "hidden" variables of a system because only a selected set are available for measurement. Think of a complex integrated circuit (board) with internal currents and voltages through or across individual components that cannot be directly measured. Therefore we must define the relationship between those states that are being measured and the measurement itself.

We define the *measurement vector*, $y_t$, as the linear transformation of the states (e.g., gains in an amplifier) by

$$y_t = Cx_t$$

where $y_t \in R^{p\times 1}$ is the output or measurement vector and $C \in R^{p\times n}$ is the corresponding output or measurement matrix. For our differential equation example above, we measure $z_t$ to obtain

$$y_t = [1 \quad 0 \quad \cdots \quad 0]\, x_t$$

However, let us not be deceived by its simplicity, the state-space representation of a dynamic system truly finds its niche by enabling the modeler to characterize the underlying process and "think" directly in terms of the physical variables of the system under investigation. To see this let us consider the state-space model of a continuous-time ($t \in R$) mechanical system in the following example and then generalize our results.

***Example 2.23*** Suppose that we are monitoring a building for potential damage from seismic activity. The measurement instrument chosen is an accelerometer to obtain the desired displacement data. Assuming a perfect measurement, we have

$$y_t = K_a d_t$$

for $y$ the measured displacement, $d$ the true displacement, and $K_a$ the conversion constant. Physically we know that the building must satisfy the laws of motion governing a mechanical system; therefore we represent it by a single degree of freedom system

$$m\ddot{d}_t + c\dot{d}_t + kd_t = f_e$$

with $m, c, k$ the corresponding mass, damping and spring, constant of appropriate units. Let us define the state vector as $x_t := \begin{bmatrix} d_t & \dot{d}_t \end{bmatrix}'$ and input (earthquake) $u_t = f_e$. Writing these relations, we have

$$\begin{aligned} \dot{x}_1 &= \dot{d}_t = x_2 \\ \dot{x}_2 &= \ddot{d}_t = -\frac{c}{m}\dot{d}_t - \frac{k}{m}d_t + \frac{1}{m}f_e = -\frac{c}{m}x_2 - \frac{k}{m}x_1 + \frac{1}{m}u_t \end{aligned} \tag{2.183}$$

or in vector-matrix form

$$\dot{x}_t = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} x_t + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u_t = Ax_t + Bu_t$$

with corresponding measurement model

$$y_t = K_a \begin{bmatrix} 1 & 0 \end{bmatrix} x_t$$

So we see in this physical system that our state variables are the displacement and velocity enabling us to "think" directly in terms of the structural system—a distinct advantage of state-space modeling.

With this motivation in mind, let us proceed to investigate state-space representation in a more general form to at least "touch" on its inherent richness. We start with continuous-time systems and then proceed to the discrete-time representation (the primary focus of this text) in order to show how to convert physical continuous to discrete models. We begin by formally defining the concept of state [27].

The *state* of a system at time $t$ is the "minimum" set of variables (*state variables*) along with the *input* sufficient to uniquely specify the dynamic system behavior for all $t$ over the interval $t \in [t, \infty)$. The *state vector* is the collection of state variables into a single vector. The idea of a *minimal set* of state variables is critical and all techniques to define them must assure that the smallest number of "independent" states have been defined. If not, there are some serious system theoretic issues that could arise [28], [29].

Let us consider a general formulation of a *nonlinear dynamic system*, including the output (measurement) model in state-space form (continuous-time)[12]

$$\begin{aligned} \dot{x}_t &= f(x_t, u_t) = a(x_t) + b(u_t) \\ y_t &= g(x_t, u_t) = c(x_t) + d(u_t) \end{aligned}$$

for $x_t$, $y_t$ and $u_t$ the respective $N_x$-state, $N_y$-output, and $N_u$-input vectors with corresponding system (process), input, output (measurement), and feed through functions. The $N_x$-dimensional system and input functions are defined by $a(\cdot)$ and $b(\cdot)$, while the $N_y$-dimensional output and feed through functions are given by $c(\cdot)$ and $d(\cdot)$.

In order to specify the solution of the $N_x$th-order differential equations completely, we must specify the above noted functions along with a set of $N_x$-initial conditions at time $t_0$ and the input for all $t \geq t_0$. Here $N_x$ is the "minimal" set of state variables.

If we constrain the state-space representation to be linear in the states, then we obtain the generic continuous-time, *linear time-varying state-space* model given by

$$\begin{aligned} \dot{x}_t &= A_t x_t + B_t u_t \\ y_t &= C_t x_t + D_t u_t \end{aligned} \tag{2.184}$$

where $x_t \in \mathcal{R}^{N_x \times 1}$, $u_t \in \mathcal{R}^{N_u \times 1}$, $y_t \in \mathcal{R}^{N_y \times 1}$ and the respective system, input, output, and feed-through matrices are $A \in \mathcal{R}^{N_x \times N_x}$, $B \in \mathcal{R}^{N_x \times N_u}$, $C \in \mathcal{R}^{N_y \times N_x}$, and $D \in \mathcal{R}^{N_y \times N_u}$.

The interesting property of the state-space representation is to realize that these models represent a complete generic form for almost any physical system. That is, if we have an RLC-circuit or a MCK-mechanical system, their dynamics are governed by the identical set of differential equations, but their coefficients are different. Of course, the physical meaning of the states are different, but this is the idea behind state-space—*many physical systems* can be captured by this generic form of differential equations, even though the systems are quite physically different! Systems theory, which is essentially the study of dynamic systems, is based on the study of state-space models and is rich with theoretical results exposing the underlying properties of the dynamic system under investigation. This is one of the major reasons why state-space models are employed in signal processing especially

[12]We separate $x_t$ and $u_t$ for future models, but this is not really necessary.

when the system is *multivariable*, that is, having multiple inputs and multiple outputs. Next we develop the relationship between the state-space representation and input-output relations—the transfer function.

For this development we constrain the state-space representation above to be a *linear time-invariant* (*LTI*) *state-space* model given by

$$\begin{aligned} \dot{x}_t &= A_c x_t + B_c u_t \\ y_t &= C_c x_t + D_c u_t \end{aligned} \tag{2.185}$$

where $A_t \longrightarrow A_c$, $B_t \longrightarrow B_c$, $C_t \longrightarrow C_c$ and $D_t \longrightarrow D_c$ their time-invariant counterparts with the subscript, $c$, annotating continuous-time matrices.

This *LTI* model corresponds the constant coefficient differential equation solutions which are solved using Laplace transforms. So, taking the Laplace transform of these equations, we have that

$$sX(s) - x_{t_0} = A_c X(s) + B_c U(s)$$

Then, solving for $X(s)$, we have

$$X(s) = (sI - A_c)^{-1} x_{t_o} + (sI - A_c)^{-1} B_c U(s) \tag{2.186}$$

where $I \in \mathcal{R}^{N_x \times N_x}$ is the identity matrix. The corresponding output is

$$Y(s) = C_c X(s) + D_c U(s) \tag{2.187}$$

Next, combining these relations, we obtain

$$Y(s) = \left[C_c(sI - A_c)^{-1} B_c + D_c\right] U(s) + C_c(sI - A_c)^{-1} x_{t_o} \tag{2.188}$$

From the definition of transfer function ($IC = 0$), we have the desired result

$$H(s) = C_c(sI - A_c)^{-1} B_c + D_c \tag{2.189}$$

Taking inverse Laplace transforms of this equation gives us the corresponding *impulse-response matrix* of the *LTI*-system as [27]

$$H(t, \tau) = C_c e^{A_c(t-\tau)} B_c + D_c \delta(t - \tau) \qquad \text{for } t \geq \tau \tag{2.190}$$

So we see that the state-space representation enables us to express the input-output relations in terms of the internal variables or states. Note also that this is a multivariable representation as compared to the usual single input–single output (scalar) systems models (*ARMAX*, *AR*, etc.) discussed in the previous sections.

Now that we have the multivariable transfer function representation of our *LTI* system, we can solve the state equations directly using inverse transforms to obtain the time-domain solutions. We first simplify the notation by defining the Laplace

transform of the state transition matrix or the so-called resolvant matrix of systems theory [27], [29] as

$$\Phi(s) := (sI - A_c)^{-1} \tag{2.191}$$

Therefore we can rewrite the transfer function matrix as

$$H(s) = C_c\Phi(s)B_c + D_c \tag{2.192}$$

and the corresponding state-input transfer matrix by

$$X(s) = \Phi(s)x_{t_0} + \Phi(s)B_cU(s) \tag{2.193}$$

Taking the inverse Laplace transformation gives the time-domain solution

$$x_t = \mathcal{L}^{-1}[X(s)] = \Phi_{t,t_0}x_{t_0} + \Phi_{t,t_0}B_c * u_{t_0}$$

or

$$x_t = \underbrace{\Phi_{t,t_0}x_{t_0}}_{\text{Zero}-\text{input}} + \underbrace{\int_{t_0}^{t} \Phi_{t,\alpha}B_cu_\alpha d\alpha}_{\text{Zero}-\text{state}} \tag{2.194}$$

with corresponding output solution

$$y_t = \underbrace{C_c\Phi_{t,t_0}x_{t_0}}_{\text{Zero}-\text{input response}} + \underbrace{\int_{t_0}^{t} C_c\Phi_{t,\alpha}B_cu_\alpha d\alpha}_{\text{Zero}-\text{state response}} \tag{2.195}$$

The *state transition matrix*, $\Phi_{t,t_0}$, is the critical component in the solution of the state equations. Ignoring the input (zero-state) of the *LTI* state-space system, we have the set of vector-matrix state equations (homogeneous)

$$\dot{x}_t = A_cx_t \tag{2.196}$$

It is well known from linear algebra [27] that this equation has the matrix exponential as its solution

$$x_t = \Phi_{t,t_0}x_{t_0} = e^{A_c(t-t_0)}x_{t_0} \tag{2.197}$$

The meaning of "transition" is now clearer, since knowledge of the transition matrix, $\Phi_{t,t_0}$, enables us to calculate the transition of the state vector at time $t$ for any $t \geq t_0$. The Laplace transform of these equations verifies the previous definition, that is,

$$X(s) = (sI - A_c)^{-1}x_{t_0} = \Phi(s)x_{t_0}$$

Therefore

$$e^{A_ct} = \mathcal{L}^{-1}\left[(sI - A_c)^{-1}\right] \tag{2.198}$$

We have that the *state transition matrix* for a *LTI* is

$$\Phi_{t,t_0} = e^{A_c(t-t_0)}, \qquad t \geq t_0 \tag{2.199}$$

In general, the state transition matrix satisfies the following *properties* [27],[28]:

1. $\Phi_{t,t_0}$ is uniquely defined for $t, t_0 \in [0, \infty)$ (unique).
2. $\Phi_{t,t} = I$ (identity)
3. $\Phi_t$ satisfies the matrix differential equation

$$\dot{\Phi}_{t,t_0} = A_t \Phi_{t,t_0}, \quad \Phi_{t_0,t_0} = I, \ \ t \geq t_0 \tag{2.200}$$

4. $\Phi_{t_k,t_0} = \Phi_{t_k,t_{k-1}} \times \Phi_{t_{k-1},t_{k-2}} \times \cdots \times \Phi_{t_1,t_0}$ (semi-group)
5. $\Phi^{-1}_{t_k,t_{k-1}} = \Phi_{t_{k-1},t_k}$ (inverse)

Thus the transition matrix plays a pivotal role in *LTI* systems theory for the analysis and prediction of the response of linear, linear time-varying and nonlinear systems [28]. For instance, the poles of a *LTI* govern such important properties of stability and response time. The poles are the *roots* of the *characteristic* (polynomial) *equation* of $A_c$, which is found by solving for the roots of the determinant of the resolvant, that is,

$$|\Phi(s)| = |(sI - A_c)|_{s=p_i} = 0 \tag{2.201}$$

Stability is determined by assuring that all of the poles lie within the left half of the $S$-plane. The poles of the system determine it response as

$$y_t = \sum_{i=1}^{N_x} K_i e^{-p_i t} \tag{2.202}$$

where $A_c$ is diagonal in this case given by $A_c = \text{diagonal}\left[p_1, p_2, \ldots, p_{N_x}\right]$, the eigenvalues of $A_c$.[13] Next we consider the discrete-time state-space model, which is the emphasis of this text.

### 2.11.2 Discrete State-Space Models

In this subsection we discuss discrete state-space models. These models evolve in two distinct ways: naturally from the problem or from sampling a continuous-time dynamical system. An example of a natural discrete system is the dynamics of balancing our own checkbook. Here the state is the evolving balance at $x(t), \ t \in$

[13]There are other system theoretic properties in model-based signal processing that are important and will be discussed in subsequent chapters.

$\mathcal{I}$, given the past balance at $x(t-1)$ and the amount of the last check, $u(t-1)$, $u(0)=0$, that is,

$$x(t) = x(t-1) - u(t-1), \qquad u(0) = 0,\ t > 0$$

There is "no information" between time samples, so this model represents a discrete-time system that evolves naturally from the underlying problem. On the other hand, if we have a physical system governed by continuous-time dynamics, then we "sample" it at given time instants. We must preserve the dynamics to assure ourselves that the continuous-time signal can be reconstructed from the discrete (Nyquist sampling theorem). Examining the physical case of a parallel $RC$-circuit, we have that the output voltage is related to the input current through Kirchoff's law to give

$$C\dot{e}_t + \frac{1}{R}e_t - i_t = 0$$

with output voltmeter measurement

$$v_t = Ke_t, \qquad t \in \mathcal{R}$$

Here we must sample the system with an analog-to-digital (A/D) converter, $t \longrightarrow t_k = k\triangle T$ with $\triangle T$ the sampling interval selected to, at least, satisfy Nyquist sampling leading to a set of difference equations using the *first-difference* approximation,

$$\dot{e}_t \approx \frac{e(t) - e(t-1)}{\triangle T}$$

to obtain

$$e(t) = \left(1 - \frac{\triangle T}{RC}\right)e(t-1) + \frac{\triangle T}{C}i(t-1)$$

$$v(t) = Ke(t), \quad t \in \mathcal{I}$$

So we see that discrete-time dynamical systems can evolve from a wide variety of problems both naturally (checkbook) or physically ($RC$-circuit). In this text we are primarily interested in physical systems (physics-based models), so we will concentrate on sampled systems reducing them to a discrete-time state-space model.

If we extend the first-difference approximation to the general *LTI* continuous-time state-space model, then we have that

$$\dot{x}_t \approx \frac{x(t) - x(t-1)}{\triangle T} \approx A_c x(t-1) + B_c u(t-1)$$
$$y_t \approx y(t) = C_c x(t) + D_c u(t)$$

Solving for $x(t)$, we obtain

$$x(t) = (I + A_c\triangle T)\,x(t-1) + B_c\triangle T u(t-1)$$
$$y(t) = C_c x(t) + D_c u(t) \tag{2.203}$$

Recognizing that the first-difference approximation is equivalent to a first-order Taylor series approximation of $A_c$ gives the discrete system, input, output, and feed-through matrices:

$$\begin{aligned} A &\approx I + A_c \triangle T + O(\triangle T^2) \\ B &\approx B_c \triangle T \\ C &\approx C_c \\ D &\approx D_c \end{aligned} \tag{2.204}$$

Another way to develop a sampled data model is to sample the solution of the continuous-time state-space model of Eq. (2.194) $t \longrightarrow t_k$ over the interval $[t_k, t_{k-1}]$; therefore

$$x_{t_k} = \Phi_{t_k, t_{k-1}} x_{t_{k-1}} + \int_{t_{k-1}}^{t_k} \Phi_{t_k, \alpha} B_c u_\alpha d\alpha \tag{2.205}$$

If we assume that the input, $u_{t_{k-1}}$ is *piecewise constant* over the interval $[t_k, t_{k-1}]$, then we can write

$$x_{t_k} = \Phi_{t_k, t_{k-1}} x_{t_{k-1}} + \Gamma_{t_k} u_{t_{k-1}} \tag{2.206}$$

where

$$\Gamma_{t_k} = \int_{t_{k-1}}^{t_k} \Phi_{t_k, \alpha} B_c d\alpha \tag{2.207}$$

which is the discrete-time, *sampled data* description of the original continuous state-space model. This simple derivation enables us to understand in some sense how we progress from a physical differential system to a sampled discrete system. Let us now return to the natural discrete representation and develop its properties replacing the Laplace transform with the $Z$-transform and then return to show the equivalence with the sample data system.

We define the *nonlinear discrete-time state-space representation* by its process or system model

$$x(t) = f\,[x(t-1), u(t-1)] = a\,[x(t-1)] + b\,[u(t-1)] \tag{2.208}$$

and corresponding output or measurement model by

$$y(t) = g\,[x(t), u(t)] = c\,[x(t)] + d\,[u(t)] \tag{2.209}$$

where $x(t), u(t), y(t)$ are the respective discrete-time, $N_x$-state, $N_u$-input, and $N_y$-output vectors with corresponding system (process), input, output and feed-through functions: the $N_x$-dimensional system and input functions, $a[\cdot]$, $b[\cdot]$, and the $N_y$-dimensional output and feed-through functions, $c[\cdot]$, $d[\cdot]$.

The discrete *linear time-varying state-space representation* follows directly and is given by the *system* or *process model* as

$$x(t) = A(t-1)x(t-1) + B(t-1)u(t-1) \tag{2.210}$$

and the corresponding discrete *output* or *measurement model* as

$$y(t) = C(t)x(t) + D(t)u(t) \tag{2.211}$$

where $x, u, y$ are the respective $N_x-$state, $N_u$-input, $N_y$-output, and $A, B, C, D$ are the $(N_x \times N_x)$-system, $(N_x \times N_u)$-input, $(N_y \times N_x)$-output, and $(N_y \times N_u)$-feed-through matrices.

The state-space representation for linear, time-invariant, discrete systems is characterized by constant system, input, output and feed through matrices, that is,

$$A(t) = A, \quad B(t) = B, \quad \text{and} \quad C(t) = C, \quad D(t) = D$$

and is given by

$$\begin{aligned} x(t) &= Ax(t-1) + Bu(t-1) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{2.212}$$

Time-invariant state-space systems can also be represented in *input-output* or *transfer function* form using $Z$-transforms to give

$$H(z) = C(zI - A)^{-1}B + D \tag{2.213}$$

Also taking inverse $Z$-transforms, we obtain the discrete impulse response matrix as

$$H(t,k) = CA^{t-k}B + D \qquad \text{for} \quad t \geq k \tag{2.214}$$

The solution to the state-difference equations can easily be derived by induction [29] or using the transfer function approach of the previous subsection. In any case it is given by the relations

$$x(t) = \Phi(t,k)x(k) + \sum_{i=k+1}^{t} \Phi(t,i)B(i)u(i) \qquad \text{for} \quad t > k \tag{2.215}$$

where $\Phi(t,k)$ is the *discrete state-transition* matrix. For time-varying systems, it can be shown that the state-transition matrix[14] satisfies (by induction)

$$\Phi(t,k) = A(t-1) \times A(t-2) \times \cdots \times A(k)$$

[14]Recall that for a sampled-data system, the state-transition matrix is $\Phi(t,t_k) = e^{A_t - t_k}$, where $A_t$ is the continuous-time system matrix.

while for time-invariant systems the state-transition matrix is given by

$$\Phi(t,k) = A^{t-k} \qquad \text{for } t > k$$

The discrete-state transition matrix possesses analogous properties to its continuous-time counterpart, that is,

1. $\Phi(t,k)$ is uniquely defined (unique)
2. $\Phi(t,t) = I$ (identity)
3. $\Phi(t,k)$ satisfies the matrix difference equation:

$$\Phi(t,k) = A(t-1)\Phi(t-1,k), \quad \Phi(k,k) = I, \qquad t \geq k+1 \quad (2.216)$$

4. $\Phi(t,k) = \Phi(t,t-1) \times \Phi(t-1,t-2) \times \cdots \times \Phi(k+1,k)$ (semi-group)
5. $\Phi^{-1}(t,k) = \Phi(k,t)$ (inverse)

As in the continuous case, the discrete-time transition matrix has the same importance in discrete systems theory which we discuss next.

### 2.11.3 Discrete Systems Theory

In this subsection we investigate the discrete state-space model from the systems theory viewpoint. There are certain properties that a dynamic system must possess in order to assure a consistent representation of the dynamics under investigation. For instance, it can be shown [28] that a necessary requirement of a measurement system is that it is *observable*, that is, measurements of available variables or parameters of interest provide enough information to reconstruct the internal variables or states. Mathematically, a system is said to be *completely observable*, if for any initial state, say $x(0)$, in the state space, there exists a finite $t > 0$ such that knowledge of the input $u(t)$ and the output $y(t)$ is sufficient to specify $x(0)$ uniquely. Recall that the deterministic linear *state-space* representation of a discrete system is defined by the following set of equations:

State model: $x(t) = A(t-1)x(t-1) + B(t-1)u(t-1)$
with corresponding measurement system or output defined by
Measurement model: $y(t) = C(t)x(t)$

Using this representation, the simplest example of an observable system is one in which each state is measured directly; therefore the measurement matrix $C$ is a $N_x \times N_x$ matrix. Thus, from the measurement system model, we have that in order to reconstruct $x(t)$ from its measurements $y(t)$, then $C$ must be invertible. In this case the system is said to be completely observable; however, if $C$ is not invertible, then the system is said to be *unobservable*. The next level of complexity involves the solution to this same problem when $C$ is a $N_y \times N_x$ matrix, then a pseudoinverse must be performed instead [27], [28]. We begin by asking the

necessary conditions for its solution. However, in the general case the solution gets more involved because we are not just interested in reconstructing $x(t)$, but $x(t)$ over all finite values of $t$. Therefore we must include the state model, that is, the dynamics as well.

With this motivation in mind, we now formally define the concept of observability. The solution to the state representation is governed by the state-transition matrix, $\Phi(t,0)$. Recall here that the state equation is [29]

$$x(t) = \Phi(t,0)x(0) + \sum_{k=0}^{t-1} \Phi(t,k)B(k)u(k)$$

Therefore pre-multiplying by the measurement matrix yields the output relations

$$y(t) = C(t)\Phi(t,0)x(0) + \sum_{k=0}^{t-1} C(t)\Phi(t,k)B(k)u(k) \tag{2.217}$$

or rearranging yields the definition

$$\tilde{y}(t) := y(t) - \sum_{k=0}^{t-1} C(t)\Phi(t,k)B(k)u(k) = C(t)\Phi(t,0)x(0) \tag{2.218}$$

The problem is to solve this resulting equation for the initial state. Therefore after multiplying both sides by $\Phi' C'$, we can infer the solution from the relation

$$\Phi'(t,0)C'(t)C(t)\Phi(t,0)x(0) = \Phi'(t,0)C(t)\tilde{y}(t)$$

Thus the observability question now becomes under what conditions can this equation uniquely be solved for $x(0)$? Equivalently we are asking if the null space of $C(t)\Phi(t,0)$ is $\mathbf{0} \in \mathcal{R}^{N_x \times 1}$. It has been shown [28], [30] that the following $N_x \times N_x$ *observability Gramian* has the identical null space, that is,

$$\mathcal{O}(0,t) := \sum_{k=0}^{t-1} \Phi'(k,0)C'(k)C(k)\Phi(k,0) \tag{2.219}$$

which is equivalent to determining that $\mathcal{O}(0,t)$ is nonsingular or rank $N_x$. Further assuming that the system is *LTI*, then this leads to the $NN_y \times N_x$ *observability matrix* [30] given by

$$\mathcal{O}(N) := \begin{bmatrix} C \\ --- \\ \vdots \\ --- \\ CA^{N-1} \end{bmatrix} \tag{2.220}$$

It can be shown that a necessary and sufficient condition for a system to be completely observable is that the *rank* of $\mathcal{O}$ or $\rho[\mathcal{O}(N)]$ must be $N_x$. Thus, for the *LTI* case, checking that all of the measurements contain the essential information to reconstruct the states for a linear time-invariant system reduces to that of checking the rank of the observability matrix. Although this is a useful mathematical concept, it is primarily used as a rule-of-thumb in the analysis of complicated measurement systems.

Analogous to the system theoretic property of observability is that of controllability, which is concerned with the effect of the input on the states of the dynamic system. A discrete system is said to be *completely controllable* if for any $x(t)$, $x(0) \in \mathcal{R}^{N_x \times 1}$ there exists an input sequence, $\{u(t)\}$, $t = 0, \ldots, t-1$ such that the solution to the state equations with initial condition $x(0)$ is $x(t)$ for some finite $t$. Following the same approach as for observability, we obtain that the *controllability Gramian* defined by

$$\mathcal{C}(0,t) := \sum_{k=0}^{t-1} \Phi(0,k) B(k) B'(k) \Phi'(0,k) \tag{2.221}$$

is nonsingular or $\rho[\mathcal{C}(0,t)] = N_x$.

Again, for the *LTI* system the $N_x \times NN_u$ *controllability matrix* defined by

$$\mathcal{C}(N) := \left[ \; B \mid AB \mid \cdots \mid A^{N-1}B \; \right] \tag{2.222}$$

must satisfy the rank condition, $\rho[\mathcal{C}] = N_x$ to be completely controllable [30].

If we continue with the *LTI* system description, we know from $Z$-transform theory that the discrete transfer function can be represented by an infinite power series, that is,

$$H(z) = C(zI - A)^{-1}B = \sum_{k=1}^{\infty} H(k) z^{-k} \qquad \text{for} \quad H(k) = CA^{k-1}B \tag{2.223}$$

where $H(k)$ is the $N_y \times N_u$ unit impulse response matrix or Markov sequence and $(A, B, C)$ are defined as the Markov parameters. The problem of determining the *internal description* $(A, B, C)$ from the *external description* ($H(z)$ or $\{H(k)\}$) is called the *realization problem*. Out of all possible realizations, $(A, B, C)$, having the same Markov parameters, those of smallest dimension are defined as *minimal realizations*. It will subsequently be shown that the dimension of the minimal realization is identical to the degree of the characteristic polynomial (actually the minimal polynomial for multivariable systems) or equivalently the degree of the transfer function (number of system poles).

In order to develop these relations, we define the $(N \times N_y N_u) \times (N \times N_y N_u)$ *Hankel matrix* by

$$\mathcal{H}(N) := \begin{bmatrix} H(1) & H(2) & \cdots & H(N) \\ H(2) & H(3) & \cdots & H(N+1) \\ \vdots & \vdots & \cdots & \vdots \\ H(N) & H(N+1) & \cdots & H(2N) \end{bmatrix} \tag{2.224}$$

Suppose that the dimension of the system is $N_x$. Then the Hankel matrix can be constructed such that $N = N_x$ using $2N_x$ impulse response matrices, which tells us the minimum number of terms we require to extract the Markov parameters. Also the $\rho[\mathcal{H}(N)] = N_x$, which is the dimension of the *minimal realization*. If we did not know the dimension of the system, then we would let (in theory) $N \longrightarrow \infty$ and determine the rank of $\mathcal{H}(\infty)$. Therefore the minimal dimension of an "unknown" system is the rank of the Hankel matrix. In order for a system to be *minimal*, it must be *completely controllable* and *completely observable*. This can be seen from the fact that the Hankel matrix factors as

$$\mathcal{H}(N) = \begin{bmatrix} CB & \cdots & CA^{N-1}B \\ \vdots & & \vdots \\ CA^{N-1}B & \cdots & CA^{2N-2}B \end{bmatrix} = \begin{bmatrix} C \\ \vdots \\ CA^{N-1} \end{bmatrix} \begin{bmatrix} B|\cdots|A^{N-1}B \end{bmatrix} \tag{2.225}$$

or more simply

$$\mathcal{H}(N) = \mathcal{O}(N) \times \mathcal{C}(N) \tag{2.226}$$

Since $\mathcal{H}(N) = \mathcal{O}(N)\mathcal{C}(N)$, it follows that the $\rho[\mathcal{H}(N)] = min[\rho(\mathcal{O}(N)), \rho(\mathcal{C}(N))] = N_x$. We see therefore that the properties of controllability and observability are carefully woven into that of minimality and that testing the rank of the Hankel matrix yields the dimensionality of the underlying dynamic system. This fact will prove crucial when we must "identify" a system from noisy measurement data. This completes the subsection on discrete systems theory. It should be noted that all of these properties exist for continuous-time systems (see [28] for details) as well with the continuous counterparts replacing the discrete.

### 2.11.4 Gauss-Markov (State-Space) Models

Here we investigate the case when random inputs are applied to a discrete state-space system with random initial conditions. If the excitation is a random signal, then the state is also random. Restricting the input to be deterministic $u(t-1)$ and the noise to be zero-mean, white, random, and gaussian $w(t-1)$, the *Gauss-Markov model* evolves as

$$x(t) = A(t-1)x(t-1) + B(t-1)u(t-1) + W(t-1)w(t-1) \tag{2.227}$$

where $w \sim \mathcal{N}(0, R_{ww})$ and $x(0) \sim \mathcal{N}(\overline{x}(0), P(0))$.

The solution to the Gauss-Markov equations can be obtained easily by induction to give

$$x(t) = \Phi(t,k)x(k) + \sum_{i=k}^{t-1} \Phi(t, i+1)B(i)u(i) + \sum_{i=k}^{t-1} \Phi(t, i+1)W(i)w(i) \tag{2.228}$$

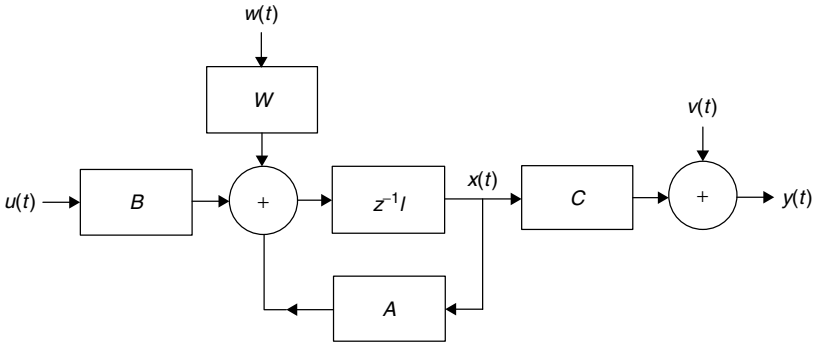



**Figure 2.17.** Gauss-Markov model of a discrete process.

which is Markov depending only on the previous state, and since $x(t)$ is merely a linear transformation of gaussian processes, it is also gaussian. Thus we can represent a Gauss-Markov process easily using the state-space models.

When the *measurement* model is also included, we have

$$y(t) = C(t)x(t) + v(t) \tag{2.229}$$

where $v \sim \mathcal{N}(0, R_{vv})$. The model is shown diagrammatically in Figure 2.17.

Since the Gauss-Markov model of Eq. (2.227) is characterized by a gaussian distribution, it is completely specified statistically by its mean and variance. Therefore, if we take the expectation of Eqs. (2.227) and (2.229), respectively, we obtain the *state mean vector* $m_x$ as

$$m_x(t) = A(t-1)m_x(t-1) + B(t-1)u(t-1) \tag{2.230}$$

and the *measurement mean vector* $m_y$ as

$$m_y(t) = C(t)m_x(t) \tag{2.231}$$

The *state variance* $P(t) := \text{var}\{x(t)\}$ is given by the discrete Lyapunov equation[15]:

$$P(t) = A(t-1)P(t-1)A'(t-1) + W(t-1)R_{ww}(t-1)W'(t-1) \tag{2.232}$$

The measurement variance, $R_{yy}(t) := \text{var}\{y(t)\}$ is

$$R_{yy}(t) = C(t)P(t)C'(t) + R_{vv}(t) \tag{2.233}$$

Similarly it can be shown [4] that the *state covariance* propagates according to the following equations:

$$P(t,k) = \begin{cases} \Phi(t,k)P(k), & \text{for } t \geq k \\ P(t)\Phi'(t,k), & \text{for } t \leq k \end{cases} \tag{2.234}$$

[15]We use the shorthand notation $P(k) := P_{xx}(k,k) = \text{cov}\{x(k), x(k)\} = \text{var}\{x(k)\}$ throughout this text.

**Table 2.9. Gauss-Markov Representation**

State propagation

$$x(t) = A(t-1)x(t-1) + B(t-1)u(t-1) + W(t-1)w(t-1)$$

State mean propagation

$$m_x(t) = A(t-1)m_x(t-1) + B(t-1)u(t-1)$$

State variance/covariance propagation

$$P(t) = A(t-1)P(t-1)A'(t-1) + W(t-1)R_{ww}(t-1)W'(t-1)$$

$$P(t,k) = \begin{cases} \Phi(t,k)P(k) & t \geq k \\ P(t)\Phi'(t,k) & t \leq k \end{cases}$$

Measurement propagation

$$y(t) = C(t)x(t) + v(t)$$

Measurement mean propagation

$$m_y(t) = C(t)m_x(t)$$

Measurement variance/covariance propagation

$$R_{yy}(t) = C(t)P(t)C'(t) + R_{vv}(t)$$

$$R_{yy}(t,k) = C(t)P(t)C'(t) + R_{vv}(t,k)$$

We summarize the Gauss-Markov and corresponding statistical models in Table 2.9.

If we restrict the Gauss-Markov model to the stationary case, then

$$A(t) = A$$

$$B(t) = B$$

$$C(t) = C$$

$$W(t) = W$$

$$R_{ww}(t) = R_{ww}$$

$$R_{vv}(t) = R_{vv}$$

and the variance equations become

$$P(t) = AP(t-1)A' + WR_{ww}W'$$

and

$$R_{yy}(t) = CP(t)C' + R_{vv} \tag{2.235}$$

At steady-state ($t \to \infty$) we have

$$P(t) = P(t-1) = \cdots = P_{ss} := P$$

Therefore the measurement covariance relations become

$$R_{yy}(0) = CPC' + R_{vv} \qquad \text{for lag} \quad k = 0 \tag{2.236}$$

It is possible to show that

$$R_{yy}(k) = CA^{|k|}PC' \qquad \text{for} \quad k \neq 0 \tag{2.237}$$

The measurement power spectrum is easily obtained by taking the $Z$-transform of this equation to obtain

$$S_{yy}(z) = CS_{xx}(z)C' + S_{vv}(z) \tag{2.238}$$

where

$$S_{xx}(z) = T(z)S_{ww}(z)T'(z^{-1}) \qquad \text{for} \quad T(z) = (zI - A)^{-1}W$$

with

$$S_{ww}(z) = R_{ww} \quad \text{and} \quad S_{vv}(z) = R_{vv}$$

Thus, using $H(z) = CT(z)$, we have the spectrum given by

$$S_{yy}(z) = H(z)R_{ww}H'(z^{-1}) + R_{vv} \tag{2.239}$$

So we see that the Gauss-Markov state-space model enables us to have a more general representation than the *ARMAX* or *lattice* models. In fact we are able to easily handle the multichannel and nonstationary statistical cases within this framework. Generalizations are also possible with the vector *ARMA* models, but the forms become quite complicated and require some knowledge of multivariable systems theory and canonical forms (see [27] for details). Before we leave this subject, let us reapproach the input-output example given previously with Gauss-Markov models. Consider the following example that indicates the equivalence between input-output and state-space models.

***Example 2.24*** Suppose that we are given the *ARMA* relations, that is,

$$y(t) = -ay(t-1) + e(t-1)$$

The corresponding state-space representation is obtained as

$$x(t) = -ax(t-1) + w(t-1) \quad \text{and} \quad y(t) = x(t)$$

Using these models, we obtain the variance equation

$$P(t) = a^2 P(t-1) + R_{ww}$$

However, $P(t) = P$ for all t, so

$$P = \frac{R_{ww}}{1-a^2}$$

Therefore with $R_{vv} = 0$

$$R_{yy}(k) = CA^{|k|}PC' = \frac{a^{|k|}R_{ww}}{1-a^2} \quad \text{and} \quad R_{yy}(0) = CPC' + R_{vv} = \frac{R_{ww}}{1-a^2}$$

Choosing $R_{ww} = 1 - a^2$ gives $R_{yy}(k) = a^{|k|}$ and

$$S_{yy}(z) = H(z)R_{ww}H'(z^{-1}) + R_{vv} = \frac{1}{1-az^{-1}}R_{ww}\frac{1}{1-az}$$

as before. So we conclude that for stationary processes these models are equivalent. If we let $a = -0.75$, $x(0) \sim \mathcal{N}(1, 2.3)$, $w \sim \mathcal{N}(0, 1)$, and $v \sim \mathcal{N}(0, 4)$, then the Gauss-Markov model is given by

$$x(t) = 0.75x(t-1) + w(t-1) \quad \textit{and} \quad y(t) = x(t) + v(t)$$

The corresponding statistics are given by the mean relations

$$\begin{aligned} m_x(t) &= 0.75m_x(t-1), & m_x(0) &= 1 \\ m_y(t) &= m_x(t), & m_y(0) &= m_x(0) \end{aligned}$$

The variance equations are

$$P(t) = 0.5625P(t-1) + 1 \quad \text{and} \quad R_{yy}(t) = P(t) + 4$$

We apply the simulator available in *SSPACK_PC* [31] to develop a 100-point simulation. The results are shown in Figure 2.18*a* through *c*. In Figure 2.18*a* and Figure 2.18*b* we see the mean and simulated states with corresponding confidence interval about the mean, that is,

$$[m_x(t) \pm 1.96\sqrt{P(t)}]$$

and

$$P = \frac{R_{ww}}{1-a^2} = 2.286$$

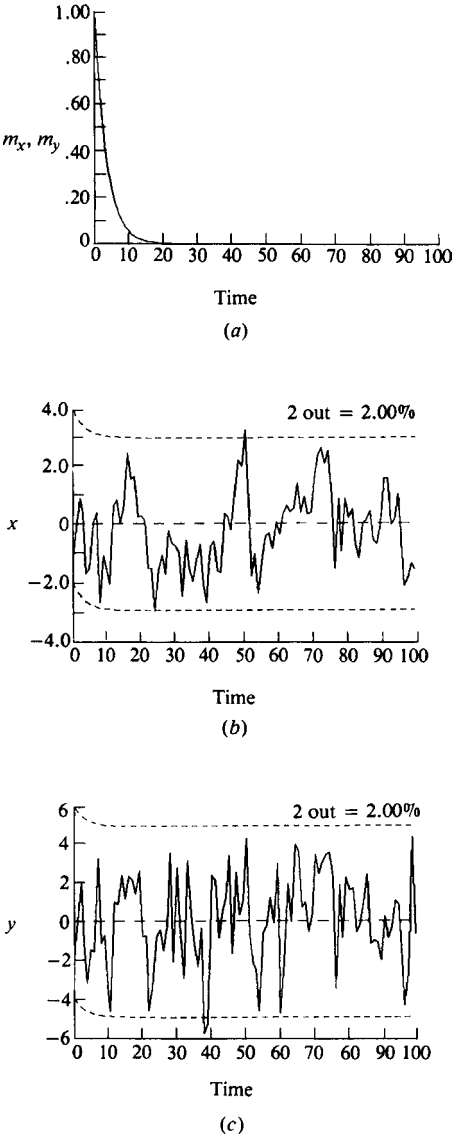



**Figure 2.18.** Gauss-Markov simulation of first-order process.

We expect 95% of the samples to lie within $(m_x \to 0) \pm 3.03$. From the figure we see that only 2 of the 100 samples exceed the bound, indicating a statistically acceptable simulation. We observe similar results for the simulated measurements. The steady-state variance is given by

$$R_{yy} = P + R_{vv} = 2.286 + 4 = 6.286$$

Therefore we expect 95% of the measurement samples to lie within $(m_y \to 0)$ $\pm 5.01$ at steady state. This completes the example.

In the next subsection we develop a special state-space model.

### 2.11.5 Innovations (State-Space) Models

In this subsection we briefly develop the innovations model that is related to the Gauss-Markov representation just discussed. The significance of this model will be developed throughout the text, but we take the opportunity now to show its relationship to the basic Gauss-Markov representation. We start by extending the original Gauss-Markov representation to the correlated process and measurement noise case and then show how the innovations model is a special case of this structure.

The *standard* Gauss-Markov model for *correlated process and measurement noise* is given by

$$\begin{aligned} x(t) &= Ax(t-1) + Bu(t-1) + W(t-1)w^*(t-1) \\ y(t) &= Cx(t) + v^*(t) \end{aligned} \tag{2.240}$$

where $R^*(t,k) := R^*\delta(t-k)$ and

$$R^* := \left[\begin{array}{c|c} R_{w^*w^*} & R_{w^*v^*} \\ \hline R_{v^*w^*} & R_{v^*v^*} \end{array}\right] = \left[\begin{array}{c|c} WR_{ww}W' & WR_{wv} \\ \hline R_{vw}W' & R_{vv} \end{array}\right]$$

Here we observe that in the standard Gauss-Markov model, the $(N_x + N_v) \times (N_x + N_v)$ block covariance matrix, $R^*$, is full with cross-covariance matrices $R_{w^*v^*}$ on its off-diagonals. The standard model assumes that they are null (uncorrelated). To simulate a system with correlated $w(t)$ and $v(t)$ is more complicated using this form of the Gauss-Markov model because $R^*$ must first be factored such that

$$R^* = \left[\begin{array}{c} R_1^* \\ R_2^* \end{array}\right] \left[\begin{array}{cc} R_1^{*\prime} & R_2^{*\prime} \end{array}\right] \tag{2.241}$$

where $R_i^*$ are matrix square roots [4], [32]. Once the factorization is performed, then the correlated noise is synthesized "coloring" the uncorrelated noise sources, $w(t)$ and $v(t)$, as

$$\left[\begin{array}{c} w^*(t) \\ v^*(t) \end{array}\right] = \left[\begin{array}{c} R_1^{*\prime}w(t) \\ R_2^{*\prime}v(t) \end{array}\right] \tag{2.242}$$

The innovations model is a constrained version of the correlated Gauss-Markov characterization. If we assume that $\{e(t)\}$ is a zero-mean, white, gaussian sequence, that is, $e \sim \mathcal{N}(0, R_{ee})$, then a particular Gauss-Markov model evolves defined as the *innovations model* [11],[26] by

$$\begin{aligned} x(t) &= A(t-1)x(t-1) + B(t-1)u(t-1) + K(t-1)e(t-1) \\ y(t) &= C(t)x(t) + D(t)u(t) + e(t) \end{aligned} \tag{2.243}$$

where $e(t)$ is the $N_y$-dimensional innovations vector and $K(t-1)$ is the $(N_x \times N_y)$ weighting matrix.[16] For the time invariant case, we have

$$R^*_{ee} := \text{cov}\left(\begin{bmatrix} Ke(t) \\ e(t) \end{bmatrix}, \begin{bmatrix} Ke(k) \\ e(k) \end{bmatrix}\right) = \left[\begin{array}{c|c} KR_{ee}K' & KR_{ee} \\ \hline R_{ee}K' & R_{ee} \end{array}\right]\delta(t-k)$$

It is important to note that the innovations model has implications in Wiener-Kalman filtering (spectral factorization) because $R^*_{ee}$ and can be represented in *factored* or *square-root* form ($R := \sqrt{R}\sqrt{R}'$) directly in terms of the gain and innovation covariance matrix, that is,

$$R^*_{ee} := \begin{bmatrix} K\sqrt{R_{ee}} \\ \sqrt{R_{ee}} \end{bmatrix}\begin{bmatrix} \sqrt{R_{ee}}K' & \sqrt{R_{ee}} \end{bmatrix}\delta(t-k) \tag{2.244}$$

Comparing the innovations model to the Gauss-Markov model, we see that they both are equivalent corresponding to the case where $w$ and $v$ are correlated. This completes the discussion of the innovations model. Next we show the equivalence of the various model sets to this family of state-space representations.

## 2.12 STATE-SPACE, *ARMAX* (*AR*, *MA*, *ARMA*, LATTICE) EQUIVALENCE MODELS

In this section we show the equivalence between the *ARMAX*, lattice and state-space models (for scalar processes). That is, we show how to obtain the state-space model given the *ARMAX* or lattice models by inspection. We choose particular coordinate systems in the state-space (canonical forms) and obtain a relationship between entries of the state-space system to coefficients of the *ARMAX* model. An example is presented that shows how these models can be applied to realize a random signal. First we consider the *ARMAX* to state-space transformation.

Recall from Eq. (2.95) that the general difference equation form of the *ARMAX* model is given by

$$y(t) = -\sum_{i=1}^{N_a} a_i y(t-i) + \sum_{i=0}^{N_b} b_i u(t-i) + \sum_{i=0}^{N_c} c_i e(t-i) \tag{2.245}$$

Equivalently in the frequency domain is

$$\begin{aligned} Y(z) &= \left(\frac{b_o + b_1 z^{-1} + \cdots + b_{N_b} z^{-N_b}}{1 + a_1 z^{-1} + \cdots + a_{N_a} z^{-N_a}}\right) U(z) \\ &\quad + \left(\frac{c_o + c_1 z^{-1} + \cdots + c_{N_c} z^{-N_c}}{1 + a_1 z^{-1} + \cdots + a_{N_a} z^{-N_a}}\right) E(z) \end{aligned} \tag{2.246}$$

Here $N_a \geq N_b$ and $N_c$ and $\{e(t)\}$ is a zero-mean white sequence with spectrum given by $R_{ee}$.

[16] Actually $K$ is called the *Kalman gain matrix*, which will be discussed in detail when we develop the model-based processor in a subsequent chapter.

It is straightforward but tedious to show (see [6]) that the *ARMAX* model can be represented in *observer canonical form*:

$$\begin{aligned} x(t) &= A_0 x(t-1) + B_0 u(t-1) + W_0 e(t-1) \\ y(t) &= C_0' x(t) + b_0 u(t) + c_0 e(t) \end{aligned} \tag{2.247}$$

where $x, u, e,$ and $y$ are the $N_a$-state vector, scalar input, noise, and output with

$$A_0 := \left[\begin{array}{c|c} 0 & -a_{N_a} \\ \hline I_{N_a-1} & \begin{array}{c} \vdots \\ -a_1 \end{array} \end{array}\right]$$

$$B_0 := \left[\begin{array}{c} -a_{N_a} b_0 \\ \vdots \\ -a_{N_b+1} b_0 \\ \hline b_{N_b} - a_{N_b} b_0 \\ \vdots \\ b_1 - a_1 b_0 \end{array}\right]$$

$$W_0 := \left[\begin{array}{c} -a_{N_a} c_0 \\ \vdots \\ -a_{N_c+1} c_0 \\ \hline c_{N_c} - a_{N_c} c_0 \\ \vdots \\ c_1 - a_1 c_0 \end{array}\right]$$

$$C_0' := [0 \ \cdots \ 0 \ 1]$$

Noting this structure, we see that each of the matrix or vector elements $\{A_{i,N_a}, B_i, W_i, C_i\}$ $i = 1, \ldots, N_a$ can be determined from the relations

$$\begin{aligned} A_{i,N_a} &= -a_i, \qquad i = 1, \ldots, N_a \\ B_i &= b_i - a_i b_0 \\ W_i &= c_i - a_i c_0 \\ C_i &= \delta(N_a - i) \end{aligned} \tag{2.248}$$

where

$$\begin{aligned} b_i &= 0 \qquad \text{for } i > N_b \\ c_i &= 0 \qquad \text{for } i > N_c \\ \delta(i-j) &\qquad \text{is the Kronecker delta} \end{aligned}$$

Consider the following example to illustrate these relations:

***Example 2.25*** Let $N_a = 3$, $N_b = 2$, and $N_c = 1$. Then the corresponding *ARMAX* model is

$$\begin{aligned} y(t) &= -a_1 y(t-1) - a_2 y(t-2) - a_3 y(t-3) + b_0 u(t) \\ &\quad + b_1 u(t-1) + b_2 u(t-2) + c_0 e(t) + c_1 e(t-1) \end{aligned} \tag{2.249}$$

Using the observer canonical form of Eq. (2.247), we have

$$x(t) = \left[\begin{array}{ccc} 0 & 0 & -a_3 \\ \hline 1 & 0 & -a_2 \\ 0 & 1 & -a_1 \end{array}\right] x(t-1) + \left[\begin{array}{c} -a_3 b_0 \\ \hline b_2 - a_2 b_0 \\ b_1 - a_1 b_0 \end{array}\right] u(t-1)$$

$$+ \left[\begin{array}{c} -a_3 c_0 \\ -a_2 c_0 \\ \hline c_1 - a_1 c_0 \end{array}\right] e(t-1)$$

$$y(t) = [0\ 0\ 1]x(t) + b_0 u(t) + c_0 e(t)$$

This completes the example.

It is important to realize that if we assume that $\{e(t)\}$ is gaussian, then the *ARMAX* model is equivalent to the innovations representation of the previous section, that is,

$$\begin{aligned} x(t) &= Ax(t-1) + Bu(t-1) + We(t-1) \\ y(t) &= C'x(t) + b_0 u(t) + c_0 e(t) \end{aligned} \tag{2.250}$$

where in this case, $K \to W$, $D \to b_0$, and $1 \to c_0$. Also the corresponding covariance matrix becomes

$$R^*_{ee} := \operatorname{cov}\left(\left[\begin{array}{c} We(t) \\ c_0 e(t) \end{array}\right], \left[\begin{array}{c} We(k) \\ c_0 e(k) \end{array}\right]\right) = \left[\begin{array}{c|c} W\underline{R}_{ee}W' & W\underline{R}_{ee}c_0 \\ \hline c_0\underline{R}_{ee}W' & c_0\underline{R}_{ee}c_0 \end{array}\right]\delta(t-k)$$

This completes the discussion on the equivalence of *ARMAX* to state-space.

Next let us develop the state-space-equivalent model for the *moving average MA* discussed previously

$$y(t) = \sum_{i=1}^{N_c} c_i e(t-i) \quad \text{or} \quad Y(z) = C(z)E(z) = (1 + c_1 z^{-1} + \cdots + c_{N_c} z^{-N_c})E(z)$$

We define the state variable as

$$x_i(t-1) := e(t-i-1), \qquad i = 1, \ldots, N_c \tag{2.251}$$

Therefore

$$x_i(t) = e(t-i) = x_{i-1}(t-1), \qquad i = 1, \ldots, N_c \tag{2.252}$$

Expanding this expression, we obtain

$$\begin{aligned} x_1(t) &= e(t-1) \\ x_2(t) &= e(t-2) = x_1(t-1) \\ x_3(t) &= e(t-3) = x_2(t-1) \\ &\vdots \quad\quad \vdots \quad\quad \vdots \\ x_{Nc}(t) &= e(t-N_c) = x_{N_c-1}(t-1) \end{aligned} \tag{2.253}$$

or in vector-matrix form

$$\begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_{N_c}(t) \end{bmatrix} = \left[\begin{array}{ccc|c} 0 & \cdots & 0 & 0 \\ \hline 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 \end{array}\right] \begin{bmatrix} x_1(t-1) \\ x_2(t-1) \\ \vdots \\ x_{N_c}(t-1) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} e(t-1)$$

$$y(t) = \begin{bmatrix} c_1 & c_2 & \cdots & c_{N_c} \end{bmatrix} \begin{bmatrix} x_1(t-1) \\ x_2(t-1) \\ \vdots \\ x_{N_c}(t-1) \end{bmatrix} + c_0 e(t) \tag{2.254}$$

Thus the general form for the *moving average* (*MA*) state space is given by

$$\mathbf{x}(t) = \left[\begin{array}{ccc|c} 0 & \cdots & 0 & 0 \\ \hline & \mathbf{I}_{N_c-1} & & \vdots \\ & & & 0 \end{array}\right] \mathbf{x}(t-1) + \mathbf{b}e(t-1)$$

$$y(t) = \mathbf{c}'\mathbf{x}(t) + b_0 e(t) \tag{2.255}$$

with $N_x = N_c$, $\mathbf{b}, \mathbf{c} \in R^{N_x \times 1}$.

***Example 2.26*** Suppose that we have the following *MA* model

$$y(t) = c_0 e(t) + c_1 e(t-1) + c_2 e(t-2)$$

and we would like to construct the equivalent state-space representation. Then we have $N_x = 2$. Therefore

$$\mathbf{x}(t) = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \mathbf{x}(t-1) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} e(t-1)$$

$$y(t) = [c_1 \quad c_2]\mathbf{x}(t) + c_0 e(t)$$

Consider another representation for the *AR* model (all-pole) given by

$$\sum_{i=1}^{N_a} a_i y(t-i) = \sigma e(t) \quad \text{or} \quad Y(z) = \frac{\sigma E(z)}{1 + a_1 z^{-1} + \cdots + a_{N_a} z^{-N_a}} \tag{2.256}$$

Here the state vector is defined by $x_i(t-1) = y(t-i-1)$, and therefore $x_i(t) = y(t-i) = x_{i+1}(t-1)$; $i = 1, \ldots, N_a - 1$ with $x_{N_a}(t) = y(t)$. Expanding over $i$, we obtain the vector-matrix state-space model

$$\begin{bmatrix} x_1(t) \\ x_2(t) \\ \\ \vdots \\ \\ x_{N_a}(t) \end{bmatrix} = \left[\begin{array}{c|ccccc} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ \hline -a_{N_a} & -a_{N_a-1} & -a_{N_a-2} & \cdots & a_1 \end{array}\right] \times \begin{bmatrix} x_1(t-1) \\ x_2(t-1) \\ \\ \vdots \\ \\ x_{N_a}(t-1) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \\ \vdots \\ \\ \sigma \end{bmatrix} e(t-1)$$

$$y(t) = [0 \;\; 0 \;\; \cdots \;\; 1] \begin{bmatrix} x_1(t-1) \\ x_2(t-1) \\ \vdots \\ x_{N_a}(t-1) \end{bmatrix} \tag{2.257}$$

In general, we have the *AR* (*all-pole*) state-space model

$$\mathbf{x}(t) = \left[\begin{array}{c|ccccc} 0 & & & & \\ \vdots & & & \mathbf{I}_{N_a-1} & \\ 0 & & & & \\ \hline -a_{N_a} & -a_{N_a-1} & -a_{N_a-2} & \cdots & a_1 \end{array}\right] \mathbf{x}(t-1) + \mathbf{b}e(t-1)$$

$$y(t) = \mathbf{c}'\mathbf{x}(t) \tag{2.258}$$

with $N_x = N_a$, $\mathbf{b}, \mathbf{c} \in R^{N_x \times 1}$. Consider the following example to demonstrate this form:

***Example 2.27*** Given the *AR* model with $N_a = 2$ and $\sigma = \sqrt{2}$, find the equivalent state-space form. We have

$$y(t) = -a_1 y(t-1) - a_2 y(t-2) + \sqrt{2}e(t)$$

Therefore

$$x_1(t-1) = y(t-2)$$
$$x_2(t-1) = y(t-1)$$

which gives

$$x_1(t) = y(t-1) = x_2(t-1)$$
$$x_2(t) = y(t) = -a_1 y(t-1) - a_2 y(t-2) = -a_1 x_2(t-1) - a_2 x_1(t-1)$$

More succinctly we can write

$$\mathbf{x}(t) = \begin{bmatrix} 0 & 1 \\ -a_2 & -a_1 \end{bmatrix} \mathbf{x}(t-1) + \begin{bmatrix} 0 \\ \sqrt{2} \end{bmatrix} e(t-1)$$
$$y(t) = [0 \quad 1]\mathbf{x}(t)$$

Another useful state-space representation is the *normal form* that evolves if we perform a partial fraction expansion of a rational discrete transfer function model (*ARMA*) to obtain

$$h(t) = \sum_{i=1}^{N_p} R_i \, (p_i)^{-t} \quad \text{or} \quad H(z^{-1}) = \frac{Y(z^{-1})}{E(z^{-1})} = \sum_{i=1}^{N_p} \frac{R_i}{1 - p_i z^{-1}} \tag{2.259}$$

for $\{R_i, p_i\}; i = 1, \ldots, N_p$ the set of residues and poles of $H(z^{-1})$. Note that the normal form model is the decoupled or parallel system representation based on the following set of relations:

$$y_i(t) - p_i y_i(t-1) = e(t), \qquad i = 1, \ldots, N_p$$

Defining the state variable as $x_i(t) := y_i(t)$, then equivalently we have

$$x_i(t) - p_i x_i(t-1) = e(t), \qquad i = 1, \ldots, N_p \tag{2.260}$$

Therefore the output is given by

$$y(t) = \sum_{i=1}^{N_p} R_i y_i(t) = \sum_{i=1}^{N_p} R_i x_i(t), \qquad i = 1, \ldots, N_p \tag{2.261}$$

Expanding these relations over $i$, we obtain

$$\begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_{N_p}(t) \end{bmatrix} = \begin{bmatrix} p_1 & 0 & \cdots & 0 \\ 0 & p_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & p_{N_p} \end{bmatrix} \begin{bmatrix} x_1(t-1) \\ x_2(t-1) \\ \vdots \\ x_{N_p}(t-1) \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} e(t-1)$$

$$y(t) = \begin{bmatrix} R_1 & R_2 & \cdots & R_{N_p} \end{bmatrix} \begin{bmatrix} x_1(t-1) \\ x_2(t-1) \\ \vdots \\ x_{N_p}(t-1) \end{bmatrix} \tag{2.262}$$

Thus the general decoupled form of the *normal state-space model* is given by

$$\begin{aligned} \mathbf{x}(t) &= \begin{bmatrix} p_1 & 0 & \cdots & 0 \\ 0 & p_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & p_{N_p} \end{bmatrix} \mathbf{x}(t-1) + \mathbf{b}e(t-1) \\ y(t) &= \mathbf{c}'\mathbf{x}(t) \end{aligned} \tag{2.263}$$

for $\mathbf{b} \in \mathcal{R}^{N_p \times 1}$ with $\mathbf{b} = \mathbf{1}$, a $N_p$-vector of unit elements. Here $\mathbf{c} \in \mathcal{R}^{1 \times N_p}$ and $\mathbf{c}' = \begin{bmatrix} R_1 & R_2 & \cdots & R_{N_p} \end{bmatrix}$.

***Example 2.28*** Consider the following set of parameters and model with $N_x = N_p = 3$ and

$$\begin{aligned} y_i(t) &= p_i y_i(t-1) + e(t-1) \\ y(t) &= \sum_{i=1}^{3} R_i y_i(t) \end{aligned}$$

Using the *normal state-space form* structure above, we obtain, by inspection,

$$\begin{aligned} \mathbf{x}(t) &= \begin{bmatrix} p_1 & 0 & \cdots & 0 \\ 0 & p_2 & \cdots & 0 \\ 0 & 0 & \cdots & p_3 \end{bmatrix} \mathbf{x}(t-1) + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} e(t-1) \\ y(t) &= [R_1 \quad R_2 \quad R_3]\, \mathbf{x}(t) \end{aligned} \tag{2.264}$$

Next we consider lattice transformations to state space. Recall from Eq. (2.110) that the general lattice recursion is given by

$$\begin{aligned} e_f(t,i) &= e_f(t,i-1) - k_i e_b(t-1,i-1) \\ e_b(t,i) &= e_b(t-i,i-1) - k_i e_f(t,i-1) \end{aligned} \tag{2.265}$$

It is possible to show that (see [33]) that an $N$-stage lattice model can be represented in *state-space feed-forward lattice form*:

$$\begin{aligned} x(t) &= A_{ff}x(t-1) + B_{ff}u(t-1) \\ y(t) &= C'_{ff}x(t) + u(t) \end{aligned} \tag{2.266}$$

where

$$x'(t) = [e_b(t-1,0)\ldots e_b(t-1,N-1)]$$
$$y(t) = e_f(t,N)$$

and

$$A_{ff} = \begin{bmatrix} 0 & & \cdots & & 0 \\ 1 & & & & \\ k_1k_2 & 1 & & & \\ \vdots & \vdots & & \ddots & \\ k_1k_{N-1} & k_2k_{N-1} & \ldots & k_{N-2}k_{N-1} & 1 \end{bmatrix}, \quad B_{ff} = \begin{bmatrix} 1 \\ -k_1 \\ \vdots \\ -k_{N-1} \end{bmatrix}$$
$$C'_{ff} = [-k_1 \ \ldots \ -k_N]$$

This *MA* model of the feed-forward lattice structure leads to the following relations:

$$A_{ij} = \begin{cases} 1, & i = j+1 \\ k_{i-j}k_i, & i > j \\ 0, & i \le j \end{cases}$$
$$B_{i+1} = -k_i$$
$$i = 1, \ldots, N-1$$
$$B_1 = 1$$
$$C_i = -k_i, \qquad i = 1, \ldots, N$$

The corresponding *state-space feedback lattice form* is obtained from the feed-forward lattice form using the matrix inversion lemma ([34]) as

$$A_{fb} = A_{ff} - B_{ff}C'_{ff}$$
$$B_{fb} = B_{ff}$$
$$C'_{fb} = -C'_{ff}$$

where

$$A_{fb} = \begin{bmatrix} k_1 & k_2 & \cdots & k_N \\ 1-k_1^2 & -k_1k_2 & \cdots & -k_1k_N \\ & \ddots & & \vdots \\ 0 & \cdots & 1-k_{N-1}^2 & -k_{N-1}k_N \end{bmatrix}$$

Finally, the *state-space rational lattice form* is simply a combination of the feed-forward and feedback lattice results, that is,

$$A_R = A_{fb}$$
$$B_R = B_{fb} \tag{2.267}$$

and

$$C'_R = [-g_1 \ \ldots \ -g_N]$$

where $\{g_i\}$ are the coefficients of the rational lattice recursion.

Let us consider an example to illustrate these forms.

***Example 2.29*** Assume that we have a three-stage lattice with $\{k_1,\ k_2,\ k_3\}$, and we would like to construct the corresponding feed-forward and feedback lattice state-space forms. Starting with the feed-forward form and Eq. (2.267), we have

$$A_{ff} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ k_1k_2 & 1 & 0 \end{bmatrix}, \quad B_{ff} = \begin{bmatrix} 1 \\ -k_1 \\ -k_2 \end{bmatrix}, \quad C'_{ff} = [k_1\ k_2\ k_3]$$

The feedback lattice form is determined by inspection from Eq. (2.29) as

$$A_{fb} = \begin{bmatrix} k_1 & k_2 & k_3 \\ 1-k_1^2 & -k_1k_2 & -k_1k_3 \\ 0 & 1-k_2^2 & -k_3^2 \end{bmatrix}, \quad B_{fb} = \begin{bmatrix} 1 \\ -k_1 \\ -k_2 \end{bmatrix}, \quad C'_{ff} = [k_1\ k_2\ k_3]$$

If we further assume that we have a rational lattice with coefficients $\{g_1,\ g_2,\ g_3\}$, then we obtain the rational lattice form as

$$A_R = A_{fb}, \quad B_R = B_{fb}, \quad C'_R = [-g_1\ -g_2\ -g_3]$$

This completes the section on the equivalence of *ARMAX* and lattice models to state-space forms, we summarize the relations in Table 2.10 and Table 2.11.

## 2.13 STATE-SPACE AND WAVE MODEL EQUIVALENCE

In this section we develop an equivalent state-space representation for a plane wave model. It will become obvious that the spherical wave model is equivalent and easily derived from these results. Observing the structure of the ordinary differential equations resulting after solving the wave equation using separation of variables, we see that they have a general homogeneous form given by

$$\frac{d^2W}{dw^2} + \alpha^2 W = 0 \tag{2.268}$$

Defining the state vector as $\mathbf{w} := [W \quad \dot{W}]'$ results in the generic *state-space wave model*

$$\dot{\mathbf{w}} = \begin{bmatrix} 0 & 1 \\ -\alpha^2 & 0 \end{bmatrix} \mathbf{w} \qquad \text{for} \quad \mathbf{w} \in \mathcal{C}^{N_w \times 1} \tag{2.269}$$

**Table 2.10. State-Space-to-*ARMAX* Equivalence Relations**

*ARMAX* to state space

$A_{iN_a} = -a_i, \qquad i = 1, \cdots, N_a$
$B_i = b_i - a_i b_0$
$W_i = c_i - a_i c_0$
$C_i = \delta(N_a - i)$
$b_i = 0, \qquad i > N_b$
$c_i = 0, \qquad i > N_c$
$\delta(i - j) =$ Kronecker delta
and

$$A_0 = \left[\begin{array}{c|c} 0 & A_{iN_a} \\ \hline I & \begin{array}{c} \vdots \\ A_{N_aN_a} \end{array} \end{array}\right]$$

$$B_0 = \begin{bmatrix} B_1 \\ \vdots \\ B_{N_a} \end{bmatrix}$$

$$W_0 = \begin{bmatrix} W_1 \\ \vdots \\ W_{N_a} \end{bmatrix}$$

$$C_0' = [0 \ \ldots \ 1]$$

Since this representation is complex, we can rewrite it in real form in terms of its real and imaginary parts $(W_R, W_I)$ as

$$\frac{d}{dw}\begin{bmatrix} W_R \\ \dot{W}_R \\ -- \\ W_I \\ \dot{W}_I \end{bmatrix} = \left[\begin{array}{cc|cc} 0 & 1 & 0 & 0 \\ -\alpha^2 & 0 & 0 & 0 \\ -- & -- & -- & -- \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\alpha^2 & 0 \end{array}\right]\begin{bmatrix} W_R \\ \dot{W}_R \\ -- \\ W_I \\ \dot{W}_I \end{bmatrix}$$

$$\mathbf{w}(0) = \begin{bmatrix} 1 \\ 0 \\ -- \\ 0 \\ -\alpha \end{bmatrix} \tag{2.270}$$

Assuming that $\alpha = \kappa_x, \kappa_y$ , or $\kappa_z$, we can represent any of the spatial coordinate solutions. Further letting $\alpha = \omega$, we obtain the corresponding temporal solution as

**Table 2.11. State-Space-to-*LATTICE* Equivalence Relations**

Feed-forward lattice to state-space

$$A_{ij} = \begin{cases} k_{i-1}k_i, & i > j \quad i = 1, \ldots, N \\ 1, & i = j+1 \\ 0, & i \le j \end{cases}$$

$$B_{i+1} = -k_i, \quad B_1 = 1, \qquad i = 1, \ldots, N-1$$

$$C_i = -k_i,$$

where

$$A_{ff} = \begin{bmatrix} 0 & \ldots & 0 \\ 1 & & 0 \\ & \ddots & \vdots \\ k_1k_2 & & \\ k_1k_{N-1} & \ldots & 1 \end{bmatrix}, \quad B_{ff} = \begin{bmatrix} 1 \\ -k_1 \\ \vdots \\ -k_{N-1} \end{bmatrix}, \quad C'_{ff} = [-k_1 \cdots - k_N]$$

Feedback lattice to state-space

$$A_{fb} = \begin{bmatrix} k_1 & k_2 & \ldots & k_N \\ 1-k_1^2 & -k_1k_2 & \ldots & -k_1k_N \\ \vdots & \ddots & & \vdots \\ 0 & \ldots\ 1-k_{N-1}^2 & \vdots & -k_{N-1}k_N \end{bmatrix}, \quad B_{fb} = B_{ff}, \quad C'_{fb} = -C'_{ff}$$

Rational lattice to state-space

$$A_R = A_{fb}, \quad B_R = B_{fb}, \quad C'_R = [-g_1 \ \ldots \ -g_N]$$

where $\{g_i\}$ are given in Table 2.8.

well. Note that the initial conditions will be different for each. If we specify all three coordinates, $(x, y, z)$, then the complex measurement is given by Eq. (2.151):

For simplicity, let us assume our model consists of one spatial coordinate (in $z$) and one temporal (in $t$). Then the measurement model for this system from the separation of variables is

$$y(z; t) = Z(z)T(t) \tag{2.271}$$

each in the generic form above. Note that the measurement model is nonlinear. However, by fixing $T(t)$, that is, assuming a narrowband solution, we have

$$T(t) = e^{-j\omega t} \tag{2.272}$$

Then a variety of simplified relationships are easily developed. Note also that we can fix $Z(z)$ as well by solving the corresponding eigen-equation to obtain

$$Z(z) = e^{j\kappa_z z} \tag{2.273}$$

Let us examine the various cases and see how the complex measurement evolves:

1. Both the state spaces (spatial and temporal) are used so that

$$y(z;t) = Z(z)T(t) = \left[Z_R(z) + jZ_I(z)\right]\left[T_R(t) + jT_I(t)\right] \tag{2.274}$$

2. Fix $T(t)$; then

$$y(z;t) = Z(z)e^{-j\omega t} = \left[Z_R(z) + jZ_I(z)\right]e^{-j\omega t} \tag{2.275}$$

3. Fix $Z(z)$; then

$$y(z;t) = e^{j\kappa_z z}T(t) = e^{j\kappa_z z}\left[T_R(t) + jT_I(t)\right] \tag{2.276}$$

4. Fix both $z$ and $t$,

$$y(z;t) = e^{-j(\omega t - \kappa_z z)} \tag{2.277}$$

An array samples the wavefield at specific (discrete) sensor locations that lead to discrete wave models, that is, $z \longrightarrow z_\ell$. So the measurement model at the $\ell$th-element is given by

$$y(z_\ell;t) = Z(z_\ell)T(t) \tag{2.278}$$

Note that within this framework using the spatial state-space model, the measurement is at the $\ell$th-element. Therefore each sensor is synthesized "sequentially," that is, one at a time. The measurements merely represent a set of spatial data, $\{y(z_\ell;t)\}$, $\ell = 1, \ldots, L$, that the state-space model will process. Thus the dimensionality of the array does not explicitly appear in this representation, since the data is processed sequentially in state-space. This could be a huge savings in operations for large dimensional arrays.

Thus we see that it is quite easy to represent the plane wave model in state-space form and that the system of wave equations can be discretized using finite difference methods (forward, backward, central differences). Let us demonstrate this approach with a simple example. Similarly it is possible to develop relationships for the spherical wavefronts that are essentially equivalent to the plane wave case with the complex exponentials replaced by the range-weighted exponentials.

***Example 2.30*** Let us assume that we have a one-dimensional plane wave propagating in the $z$-direction and we would like to represent it in state-space form.

Once developed, we would then like to sample the wave with a uniformly sampled $L$-element array and discretize the temporal measurements at each sensor. We assume case 2 with $T(t)$ fixed; therefore we only require a spatial state-space model. First, we develop the continuous spatiotemporal model from the plane wave representation of

$$y(z;t) = Z(z)e^{-j\omega t}$$

We use the real form of the model and begin by defining the spatial state vector, $\mathbf{x}(z) := [Z_R(z)\dot{Z}_R(z) \mid Z_I(z)\dot{Z}_I(z)]'$. Then

$$\frac{d\mathbf{x}(z)}{dz} = \left[\begin{array}{cc|cc} 0 & 1 & 0 & 0 \\ -\kappa_z^2 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \\ 0 & 0 & -\kappa_z^2 & 0 \end{array}\right] \mathbf{x}(z)$$

Next we sample with an $L$-element array, but in the spatial state-space framework the array sensors are processed sequentially one at a time. Therefore

$$\mathbf{y}(z_\ell;t_k) = \left[\begin{array}{cc|cc} 1 & 0 & j & 0 \end{array}\right] \mathbf{x}(z_\ell) \times e^{-j\omega t_k}, \qquad \ell = 1,\ldots,L$$

This completes the example.

## 2.14 SUMMARY

In this chapter we discussed the evolution of random signals as models of phenomenological events ranging from coin-flipping to a linear dynamical system. We also discussed special processes (e.g., gaussian, Markov) and properties of random signals. Under the assumption of stationarity, we developed spectral representations and the concept of simulating a stationary process with given covariance by driving a linear system with white noise. We introduced the basic models to be employed throughout the text, the autoregressive moving average model with exogeneous inputs (*ARMAX*), the lattice, exponential (harmonic), wave and the Gauss-Markov (state-space) model. We showed the equivalence of these models and pointed out how to transform from one to the other. We also introduced the innovations model as a special Gauss-Markov (state-space) model and showed how it simplifies the simulation of correlated noise sources.

## *MATLAB* NOTES

*MATLAB* and its accompanying *Signal Processing Toolbox* can be used to simulate both deterministic as well as random signals and systems. Besides processing, a full compliment of signal simulation (white noise [**rand**], gaussian white noise [**randn**], etc.), it provides the capability of synthesizing linear systems through its

convolution (**conv**) command as well as its filter (**filter**) command, which enables the simulation of *IIR* or *pole-zero* and *FIR* or *all-zero* using the *ARMAX*-based as well as deterministic *transfer function* polynomial representations. Frequency domain analysis of deterministic signals or systems follow from the fast Fourier transform commands (**fft**) and (**freq***) commands (* being a wildcard) again assuming polynomial representations. Stochastic signals and systems can be analyzed using the auto/cross covariance (**xcov**) or correlation (**xcorr**) commands while the power spectrum is estimated using a variety of *PSD* commands ranging from the classical periodogram (**periodogram**) to the improved windowed periodogram of Welch (**pwelch**) or (**psd**). Modern model-based methods abound as well, which we discuss in the Chapter 4 notes. Transformations from continuous-to-discrete scalar systems are available through the impulse invariant transformation (**impinvar**). Model equivalence transformations are available primarily from commands consisting of the lattice-to-transfer function (**latc2tf**), state-space-to-transfer function (**ss2tf**), and zero-pole-to-transfer function (**zp2tf**) and autoregressive-to-reflection coefficient (lattice) (**ar2rc**) and polynomial-to-reflection coefficients (**poly2rc**). There are wide variety of these conversions available.

The Gauss-Markov models are available in the third party toolbox, *SSPACK_PC* (see Appendix A) consisting of the canonical conversions (*ARMAX*, innovations, etc.). The other equivalence transformations discussed in this chapter are easily programmed in *MATLAB*. The *System Identification Toolbox* in *MATLAB* also provides a set of simulation commands that are used to simulate random signals and systems.

## REFERENCES


1. A. Oppenheim and R. Schafer, *Discrete-Time Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1989.
2. R. Churchill, *Complex Variables and Applications*, New York: McGraw-Hill, 1974.
3. A. Papoulis, *Probability, Random Variables and Stochastic Processes*, New York: McGraw-Hill, 1965.
4. A. Jazwinski, *Stochastic Processes and Filtering Theory*, New York: Academic Press, 1970.
5. R. Hogg and A. Craig, *Introduction to Mathematical Statistics*, New York: Macmillan, 1970.
6. S. Tretter, *Introduction to Discrete-Time Signal Processing*, New York: Wiley, 1976.
7. K. Astrom, *Introduction to Stochastic Control Theory*, New York: Academic Press, 1970.
8. F. Bauer, "A direct iterative process for the Hurwitz decomposition of a polynomial," *Arch. Elect. Ubertragung*, **9**, 285–290 1955.
9. T. Kailath and J. Rissanen, "Partial realization of random systems," *Automatica*, **8**, 389–396 1972.
10. J. Rissanen, "Algorithm for the triangular decomposition of block Hankel and Toeplitz matrices with application to factorizing positive matrix polynomials," *Math. Comput.*, **17**, 147–157 1973.

11. G. Goodwin and K. Sin, *Adaptive Filtering, Prediction and Control*, Englewood Cliffs, NJ: New Prentice-Hall, 1984.
12. E. Robinson and M. Silvia, *Digital Foundations of Time Series Analysis*, Vol. 1, San Francisco: Holden-Day, 1979.
13. S. Mitra and R. Sherwood, "Digital ladder networks," *IEEE Trans. Audio Electroacoust.*, **21**, 30–36 1973.
14. A. Gray and J. Markel, "Digital lattice and ladder filter synthesis," *IEEE Trans. Audio Electroacoust.*, **21**, 491–500 1975.
15. B. Friedlander, "Lattice filters for adaptive processing," *Proc. IEEE*, **70**, 829–867 1982.
16. J. Markel and A. Gray, *Linear Prediction of Speech*, New York: Springer-Verlag, 1976.
17. S. Orfanidis, *Optimum Signal Processing*, New York: Macmillan, 1988.
18. M. Hayes, *Statistical Digital Signal Processing and Modeling*, New York: Wiley, 1996.
19. S. M. Kay, *Modern Spectral Estimation: Theory and Applications*, Englewood Cliffs, NJ: Prentice-Hall, 1988.
20. K. Aki and P. Richards, *Quantitative Seismology: Theory and Methods*, New York: Freeman, 1980.
21. E. Robinson and M. Silvia, *Digital Foundations of Time Series Analysis: Wave Equation and Space-Time Processing*, Vol. 2 (San Francisco: Holden-Day, 1981).
22. D. Dudgeon and R. Mersereau, *Multidimensional Digital Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1984.
23. D. Johnson and D. Dudgeon, *Array Signal Processing: Concepts and Techniques*, Englewood Cliffs, NJ: Prentice-Hall, 1993.
24. E. Robinson, T. Durrani, and L. Peardon, *Geophysical Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1986.
25. M. Wax, *Detection and Estimation of Superimposed Signals*, PhD dissertation, Stanford University, Standford CA, 1985.
26. J. Candy, *Signal Processing: The Model-Based Approach*, New York: McGraw-Hill, 1986.
27. T. Kailath, *Linear Systems*, Englewood Cliffs, NJ: Prentice-Hall, 1980.
28. F. Szidarovszky and A. Bahill, *Linear Systems Theory*, Boca Raton, FL: CRC Press, 1980.
29. R. DeCarlo, *Linear Systems: A State Variable Approach*, Englewood Cliffs, NJ: Prentice-Hall, 1989.
30. C. Chen, *Introduction to Linear System Theory*, New York: Holt, Rhinehart, and Winston, 1984.
31. J. Candy and P. Candy, "SSPACK_PC: A model-based signal processing package on personal computers," *DSP Appl.*, **2**(3), 33–42 1993 (see **http://www.techni-soft.net** for more details).
32. G. Bierman, *Factorization Methods for Discrete Sequential Estimation*, New York: Academic Press, 1977.
33. D. Lee, *Canonical Ladder Form Realizations and Fast Estimation Algorithms*, PhD dissertation, Stanford Univ., 1980.
34. T. Kailath, *Lectures on Kalman and Wiener Filtering Theory*, New York: Springer-Verlag, 1981.

## PROBLEMS

**2.1** Suppose that we are given the following system:

$$y(t) = e^{-t}x(t+1)$$

Determine if it is

**(a)** Time invariant
**(b)** Causal
**(c)** Linear

**2.2** Calculate the response of a linear time-invariant (*LTI*) system with impulse response

$$h(t) = \begin{cases} \alpha^t, & 0 \le t \le N-1 \\ 0, & \textit{elsewhere} \end{cases}$$

when excited by the pulse

$$x(t) = \mu(t) - \mu(t-N)$$

**2.3** Suppose that we are given a digital filter

$$y(t) = -ay(t-1) + bx(t) \qquad \text{for } a < 0,\ b > 0$$

and the values of the impulse response

$$h(1) = \frac{5}{2}, \quad h(2) = \frac{5}{4}$$

Find the values of $a$, $b$, and $h(0)$.

**2.4** Find the impulse response of the system

$$y(t) = -4y(t-1) - \sin\Omega_0 y(t-2) + \ln\Omega_0 x(t-1) \qquad \text{for } \Omega_0 > 0$$

**2.5** We are given a causal *LTI* system with transfer function

$$T(z) = \frac{4}{1 - 5/2z^{-1} + z^{-2}}$$

**(a)** Find the unit impulse response of the system.
**(b)** Find the unit step response.
**(c)** Is the system stable?
**(d)** Find the inverse $Z$-transform of $T(z)$ by division. Calculate $h(t)$, $t = 0, 1, 2$ from *(a)* and compare.
**(e)** Sketch the frequency response (magnitude only) of $T(z)$.

**2.6** Calculate the frequency response of the three-point averager given by the difference equation

$$y(t) = \frac{1}{3}\sum_{i=0}^{2} x(t-i)$$

**2.7** Suppose that we are given the following *LTI* system characterized by the following difference equation

$$y(t) + \frac{3}{4}y(t-1) + \frac{1}{8}y(t-2) = x(t) + \frac{1}{3}x(t-1)$$

**(a)** Find the frequency response.

**(b)** Find the impulse response.

**(c)** Find the response to the excitation

$$x(t) = \left(\frac{1}{3}\right)^t \mu(t)$$

and zero initial conditions.

**2.8** Calculate the frequency response of the following signal, $x(t) = (1.8)^t \mu(t)$.

**2.9** Given the two digital filters, $F_1$ and $F_2$ characterized by difference equations

$$F_1: \quad y_1(t) - \frac{1}{4}y_1(t-1) = x_1(t) + \frac{1}{2}x_1(t-1), \quad y_1(-1) = 4$$

$$F_2: \quad y_2(t) - \frac{1}{8}y_2(t-1) = 4x_2(t), \quad y_2(-1) = -8$$

Find the overall performance of the cascaded filters if they are connected in series. Find the overall transfer function and unit impulse response.

**2.10** Given the discrete *LTI* system characterized by

$$y(t) + \frac{5}{36}y(t-1) = \frac{1}{36}y(t-2) = x(t-1) - \frac{1}{2}x(t-2)$$

with zero initial conditions. Find

**(a)** Impulse response.

**(b)** Frequency response.

**(c)** Simulate the system using *MATLAB* and plot the corresponding frequency response.

**2.11** A digital filter designed to operate in a microprocessor need not be causal. Suppose that we are given the system and REV means time reverse the input $x(t) \rightarrow x(-t)$:

**(a)** Find $h'(t)$, the overall impulse response.

**(b)** Find the overall frequency response, $H'(\Omega)$.

(*Note*: This is a technique to remove the phase effect of a filter.)

**2.12** Let $h(t)$ be the unit impulse response of a causal *FIR* filter with $h(t)$ real. Then the frequency response can be represented in the form

$$H(\Omega) = \hat{H}(\Omega)e^{j\phi(\Omega)} \qquad \text{for } \hat{H} \text{ real}$$

Find $\phi(\Omega)$ for $0 \leq \Omega \leq \pi$ when $h(t) = h(N-1-t)$, $N$ even.

**2.13** Show that an *FIR* filter is always stable.

**2.14** Suppose that the stochastic process $\{y(t)\}$ is generated by

$$y(t) = a\ \exp(-t) + ct, \qquad a, b \text{ random.}$$

Then

**(a)** What is the mean of the process?

**(b)** What is the corresponding covariance?

**(c)** Is the process stationary, if $E\{a\} = E\{b\} = 0$, and $E\{ab\} = 0$.

**2.15** Suppose that $x, y, z$ are gaussian random variables with corresponding means $m_x, m_y, m_z$ and variances $R_{xx}, R_{yy}, R_{zz}$. Show that:

**(a)** If $y = ax + b$, $a, b$ constants, then $y \sim N(am_x + b, a^2 R_{xx})$.

**(b)** If $x$ and $y$ are uncorrelated, then they are independent.

**(c)** If $x(i)$ are gaussian with mean $m(i)$ and variance $R_{xx}(i)$, then for

$$y = \sum_i K_i x(i), \quad y \sim N\left(\sum_i K_i m(i), \sum_i K_i^2 R_{xx}(i)\right)$$

**(d)** If $x$ and $y$ are jointly (conditionally) gaussian, then

$$E\{x|y\} = m_x + R_{xy}R_{yy}^{-1}(y + m_y)$$
$$R_{x|y} = R_{xx} + R_{xy}R_{yy}^{-1}R_{yx}$$

**(e)** The random variable $x = E\{x|y\}$ is orthogonal to $y$.

**(f)** If $y$ and $z$ are independent, then

$$E\{x|y,z\} = E\{x|y\} + E\{x|z\} - m_x.$$

**(g)** If $y$ and $z$ are not independent, show that

$$E\{x|y,z\} = E\{x|y,e\} = E\{x|y\} + E\{x|e\} - m_x$$

for $e = z - E\{x|y\}$.

**2.16** Assume that $y(t)$ is a zero mean, ergodic process with covariance $R_{yy}(k)$. Calculate the corresponding power spectra, $S_{yy}(z)$, if

**(a)** $R_{yy}(k) = Ca^{|k|}$.
**(b)** $R_{yy}(k) = C\ \cos(w|k|),\ |k| < \pi/2$.
**(c)** $R_{yy}(k) = C\ \exp(-a^{|k|})$.

**2.17** Verify the covariance-spectral density pairs of Table 2.1 for the discrete process:

**(a)** Bandlimited white noise
**(b)** Triangular

**2.18** Let the impulse response of a linear system with random input $u(t)$ be given by $h(t)$. Then show that

**(a)** $R_{yy}(k) = \sum_{m=0}^{\infty} \sum_{i=0}^{\infty} h(m)h(i)R_{uu}(k+i-m)$ and $S_{yy}(z) = H(z)H(z^{-1})S_{uu}(z)$.
**(b)** $R_{yy}(k) = \sum_{m=0}^{\infty} h(m)R_{yu}(k-m)$ and $S_{yy}(z) = H(z)S_{yu}(z)$.
**(c)** $R_{uy}(k) = \sum_{m=0}^{\infty} h(m)R_{uu}(k-m)$ and$S_{uy}(z) = H(z)S_{uu}(z)$.

**2.19** Derive the *sum decomposition* relation,

$$S_{yy}(z) = S_{yy}^{+}(z) + S_{yy}^{-}(z) - R_{yy}(0)$$

**2.20** Develop a *MATLAB* program to simulate the *ARMA* process

$$y(t) = -ay(t-1) + e(t)$$

where $a = 0.75,\ e \sim \mathcal{N}(0, 0.1)$ for 100 data points.

**(a)** Calculate the analytic covariance $R_{yy}(k)$.
**(b)** Determine an expression to "recursively" calculate, $R_{yy}(k)$.
**(c)** Plot the simulated results and construct the $\pm 2\sqrt{R_{yy}(0)}$ bounds.
**(d)** Do 95% of the samples fall within these bounds?

**2.21** Develop the digital filter to simulate a sequence, $y(t)$ with covariance $R_{yy}(k) = 4e^{-3|k|}$. Perform the simulation using *MATLAB*.

**2.22** Suppose that we are given a linear system characterized by transfer function

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

which is excited by discrete exponentially correlated noise

$$R_{xx}(k) = (1/2)^{|k|}$$

**(a)** Determine the output *PSD*, $S_{yy}(z)$.

**(b)** Determine the output covariance, $R_{yy}(k)$.

**(c)** Determine the cross-spectrum, $S_{yx}(z)$.

**(d)** Determine the cross-covariance, $R_{yx}(k)$.

**2.23** Suppose that we are given a causal *LTI* system characterized by its impulse response, $h(t)$. If this system is excited by zero-mean, unit variance white noise, then

**(a)** Determine the output variance, $R_{yy}(0)$.

**(b)** Determine the covariance, $R_{yy}(k)$ for $k > 0$.

**(c)** Suppose that the system transfer function is given by

$$H(z) = \frac{1 + b_0 z^{-1}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

Find a method to recursively calculate $h(t)$ and therefore $R_{yy}(0)$.

**2.24** Given the covariance function

$$R_{yy}(k) = e^{-1/2|k|} \cos \pi |k|$$

find the digital filter when driven by unit variance white noise produces a sequence $\{y(t)\}$ with these statistics.

**2.25** Suppose that we have a process characterized by difference equation

$$y(t) = x(t) + \frac{1}{2}x(t-1) + \frac{1}{3}x(t-2)$$

**(a)** Determine a recursion for the output covariance, $R_{yy}(k)$.

**(b)** If $x(t)$ is white with variance $\sigma_{xx}^2$, determine $R_{yy}(k)$.

**(c)** Determine the output *PSD*, $S_{yy}(z)$.

**2.26** Suppose that we are given a linear system characterized by the difference equation

$$y(t) - \frac{1}{5}y(t-1) = \frac{1}{\sqrt{3}}x(t)$$

The system is excited by (1) White gaussian noise, $x \sim \mathcal{N}(0, 3)$ and (2) exponentially correlated noise, $R_{ee}(k) = (1/2)^{|k|}$. In both cases find:

**(a)** Output *PSD*, $S_{yy}(z)$

**(b)** Output covariance, $R_{yy}(k)$

**(c)** Cross-spectrum, $S_{ye}(k)$

**(d)** Cross covariance, $R_{ye}(k)$

**2.27** Suppose that we have a *MA* process (two-point averager)

$$y(t) = \frac{e(t) + e(t-1)}{2}$$

**(a)** Develop an expression for $S_{yy}(z)$ when $e$ is white with variance $R_{ee}$.
**(b)** Let $z = \exp\{j\Omega\}$, and sketch the "response" of $S_{yy}(e^{j\Omega})$.
**(c)** Calculate an expression for the covariance $R_{yy}(k)$ in closed and recursive form.

**2.28** Suppose that we are given a zero-mean process with covariance

$$R_{yy}(k) = 10 \exp(-0.5|k|)$$

**(a)** Determine the digital filter which when driven by white noise will yield a sequence with the covariance above.
**(b)** Develop a computer program to generate $y(t)$ for 100 points.
**(c)** Plot the results, and determine of 95% of the samples fall within $\pm 2\sqrt{R_{yy}(0)}$.

**2.29** Suppose that we have the following transfer functions:

**(a)** $H_1(z) = 1 - 1/8z^{-1}$
**(b)** $H_2(z) = \dfrac{1}{1 - 3/4z^{-1} + 1/8z^{-2}}$
**(c)** $H_3(z) = \dfrac{1 - 1/8z^{-1}}{1 - 3/4z^{-1} + 1/8z^{-2}}$

Find the corresponding lattice model for each case, that is, all-zero, all-pole, and pole-zero.

**2.30** Suppose that we are given the factored power spectrum $S_{yy}(z) = H(z)H(z^{-1})$ with

$$H(z) = \frac{1 + \beta_1 z^{-1} + \beta_2 z^{-2}}{1 + \alpha_1 z^{-1} + \alpha_2 z^{-2}}$$

**(a)** Develop the *ARMAX* model for the process.
**(b)** Develop the corresponding Gauss-Markov model for *both* the standard and innovations representation of the process.

**2.31** Suppose that we are given the following Gauss-Markov model

$$\begin{aligned} x(t) &= \frac{1}{3}x(t-1) + \frac{1}{2}w(t-1) \\ y(t) &= 5x(t) + v(t) \\ w &\sim \mathcal{N}(0,3), \quad v \sim \mathcal{N}(0,2) \end{aligned}$$

**(a)** Calculate the state power spectrum, $S_{xx}(z)$.
**(b)** Calculate the measurement power spectrum, $S_{yy}(z)$.
**(c)** Calculate the state covariance recursion, $P(t)$.
**(d)** Calculate the steady-state covariance, $P(t) = \cdots = P = P_{ss}$.
**(e)** Calculate the output covariance recursion, $R_{yy}(t)$.
**(f)** Calculate the steady-state output covariance, $R_{yy}$.

**2.32** Suppose that we are given the Gauss-Markov process characterized by the state equations

$$x(t) = 0.97x(t-1) + u(t-1) + w(t-1)$$

for $u(t)$ a step of amplitude 0.03, $w \sim \mathcal{N}(0, 10^{-4})$, and $x(0) \sim \mathcal{N}(2.5, 10^{-12})$.

**(a)** Calculate the covariance of $x$, that is, $P(t) = \text{cov}(x(t))$.
**(b)** Since the process is stationary, we know that

$$P(t+k) = P(t+k-1) = \cdots = P(0) = P$$

What is the steady-state covariance $P$ of this process?
**(c)** Develop a *MATLAB* program to simulate this process.
**(d)** Plot the process $x(t)$ with the corresponding confidence limits $\pm 2\sqrt{P(t)}$ for 100 data points. Do 95% of the samples lie within the bounds?

**2.33** Suppose that the process in the problem above is measured using an instrument with uncertainty $v \sim \mathcal{N}(0, 4)$ such that

$$y(t) = 2x(t) + v(t)$$

**(a)** Calculate the output covariance $R_{yy}(k)$.
**(b)** Develop a *MATLAB* program to simulate the output using the results the previous problem.
**(c)** Plot the process $y(t)$ with the corresponding confidence limits $\pm 2\sqrt{R_{yy}(0)}$ for 100 data points. Do 95% of the samples lie within the bounds?

**2.34** Suppose that we are given the *ARMAX* model

$$\begin{aligned} y(t) = &-0.5y(t-1) - 0.7y(t-2) + u(t) + 0.3u(t-1) \\ &+ e(t) + 0.2e(t-1) + 0.4e(t-2) \end{aligned}$$

**(a)** What is the corresponding innovations model in state-space form for $e \sim \mathcal{N}(0, 10)$?
**(b)** Calculate the corresponding covariance matrix $R_{ee}^*$.

**2.35** Given the following *ARMAX* model

$$A(q^{-1})y(t) = B(q^{-1})u(t) + \frac{C(q^{-1})}{D(q^{-1}}\epsilon(t)$$

for $q^{-1}$ the backward shift (delay) operator such that

$$\begin{aligned} A(q^{-1}) &= 1 + 1.5q^{-1} + 0.7q^{-2} \\ B(q^{-1}) &= 1 + 0.5q^{-1} \\ C(q^{-1}) &= 1 + 0.7q^{-1} \\ D(q^{-1}) &= 1 + 05q^{-1} \end{aligned}$$

**(a)** Find the pulse transfer representation of this process ($C = D = 0$). Convert it to the following equivalent *pole-zero* and *normal* state-space forms. Is the system controllable? Is it observable? Show your calculations.

**(b)** Find the pole-zero or *ARX* representation of this process ($C = 1$, $D = 0$). Convert it to the equivalent state-space form.

**(c)** Find the pole-zero or *ARMAX* representation of this process ($D = 0$). Convert it to the equivalent state-space form.

**(d)** Find the all-zero or *FIR* representation of this process ($A = 1, C = D = 0$). Convert it to the equivalent state-space form.

**(e)** Find the all-pole or *IIR* representation of this process($B = 0, C = 0, D = 0$). Convert it to the equivalent state-space form.

**(f)** Find the all-zero or *MA* representation of this process($A = 1, B = 0, D = 0$)). Convert it to the equivalent state-space form.

**(g)** Using the full model above with $A, B, C, D$ polynomials, is it possible to find and equivalent Gauss-Markov representation? If so, find it and convert to the equivalent state-space form. (*Hint*: Consider the $C/D$ polynomials to be a coloring filter with input $\epsilon(t)$ and output $e(t)$.)

# 3

# ESTIMATION THEORY

## 3.1 INTRODUCTION

In the previous chapter we saw that a discrete random signal can be completely characterized by the equivalent probabilistic concept of a stochastic process. The filtering of random signals is referred to as *estimation* and the particular algorithm is called a *signal estimator* or just *estimator*. The process of *estimation* is concerned with the design of a rule or algorithm, the estimator, to extract useful signal information from random data.

Perhaps a slightly more formal description of the estimation problem evolves from its inherent structure ([1], [2], [3]) as depicted in Figure 3.1. The *source* or origin of the problem provides the parameter ($\Theta$) random or deterministic, but unknown, as a point in parameter space. Next the *probabilistic transition mechanism* ($\Pr(Y|\Theta)$) separates the parameter from the measurements $Y$. It "knows" the true parameter value and generates samples in the *measurement space* according to the underlying probability law. Finally the *estimation rule*, or simply *estimator* ($\Theta(Y)$), is the result providing the "best" (in some sense) parameter value. The estimator is based upon knowledge of a priori distributions (in the random parameter case) and of the inherent conditional probabilities contained in the transition mechanism.

There are many different estimators and algorithms. Suppose that we are given two estimators, asked to evaluate their performance to decide which one is superior. We must have a reasonable measure to make this decision. Thus we must develop techniques to investigate various properties of estimators as well as a means to decide how well they perform. Sometimes we are interested in estimators that are

*Model-Based Signal Processing*, by James V. Candy

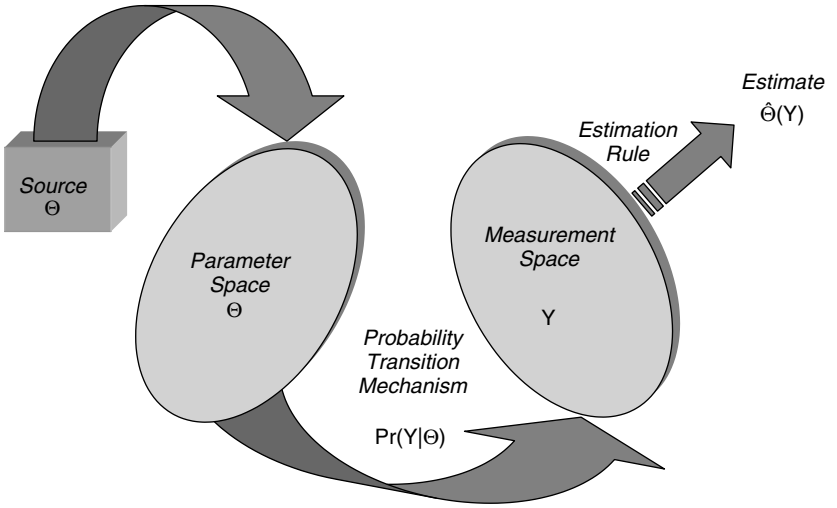



**Figure 3.1.** The estimation problem: Source, probabilistic transition mechanism, measurement space, and estimate.

not optimal for various reasons (simplicity, implementation ease, etc.) and we would like to judge how well they perform compared to the optimal. In this section we investigate desirable properties of estimators, criteria to evaluate their performance, and investigate some of the most popular schemes.

### 3.1.1 Estimator Properties

Thus the need for an agreed-upon rule(s) to measure estimator performance is necessary. The two primary statistical measures employed are the estimator mean (accuracy) and the variance (precision). These measures lead to desirable estimator properties; that is, we want estimators that are accurate or unbiased and precise. More formally, an *unbiased estimator* is one whose expected value is identical to the parameter being estimated. Suppose that we wish to estimate the random parameter, $\Theta$, then the estimator $\hat{\Theta}$[1] is *unconditionally unbiased* if

$$E\{\hat{\Theta}\} = E\{\Theta\} \qquad \forall\ \Theta \tag{3.1}$$

If $\Theta$ is a function of the measurements $Y$, then it is equivalent to have

$$E_{\hat{\Theta}Y}\{\hat{\Theta}(Y)\} = E\{\Theta\} \qquad \forall\ \Theta \tag{3.2}$$

If the estimator is conditioned on $\Theta$, then we desire a *conditionally unbiased* estimator

$$E\{\hat{\Theta}|\Theta = \theta\} = E\{\Theta\} \tag{3.3}$$

[1]Recall that we use the caret symbol (ˆ) to annotate the estimator.

A *biased* estimator, then is given by

$$E\{\hat{\Theta}\} = E\{\Theta\} + E\{B(\Theta)\} \tag{3.4}$$

where $B(\cdot)$ is the bias, which may be known or unknown. Known biases are easily removed. Note also that when the parameter $\Theta$ is *not* random but merely *unknown*, then the expectation operation results in the parameter itself ($E\{\Theta\} = \Theta$).

An estimator is *consistent* if the estimate improves as the number of measurements increase. Equivalently it is said to *converge in probability*, if

$$\lim_{t\to\infty} \Pr\left(\left[\Theta - \hat{\Theta}(Y_t)\right] = 0\right) = 1 \tag{3.5}$$

It is said to be *mean-squared convergent* if

$$\lim_{t\to\infty} E\left\{\left[\Theta - \hat{\Theta}(Y_t)\right]'\left[\Theta - \hat{\Theta}(Y_t)\right]\right\} = 0 \tag{3.6}$$

The estimator, $\hat{\Theta}$, is *asymptotically efficient*, if its corresponding estimation error defined by $\left(\tilde{\Theta} := \Theta - \hat{\Theta}(Y_t)\right)$ has the smallest (minimum) error variance of all possible estimators. That is, for any other estimator, say $\tilde{\Theta}^*$, then the following condition holds

$$\text{var}(\tilde{\Theta}^*) > \text{var}(\tilde{\Theta}) \tag{3.7}$$

Finally an estimator is called *sufficient* if it possesses all of the statistical information contained in the set of measurements regarding the parameter to be estimated. More formally, a statistic, $\hat{\Theta}$ is sufficient for $\Theta$ if the conditional probability, $\Pr(Y|\hat{\Theta})$, does not depend on $\Theta$. The underlying theory of this result is based on being able to factor the probability such that $\Pr(Y, \Theta) = f(\hat{\Theta}, \Theta)g(Y)$ (Neyman-Fisher factorization theorem) as discussed in more detail in [4] or [5]. These properties are desirable in an estimator and are determined in its evaluation.

### 3.1.2 Estimator Performance

The quality of an estimator is usually measured in terms of its *estimation error*,

$$\tilde{\Theta} = \Theta - \hat{\Theta} \tag{3.8}$$

A common measure of estimator quality is called the Cramer-Rao (*CRB*) bound. The *CRB* offers a means of assessing estimator quality prior to processing the measured data. We restrict discussion of the *CRB* to the case of unbiased estimates, $\hat{\Theta}$, of a "nonrandom" parameter $\Theta$. The bound is easily extended to more complex cases for biased estimates as well as random parameters ([1], [6]). The Cramer-Rao

bound[2] for any unbiased estimate $\hat{\Theta}$ of $\Theta$ based on the measurement $Y$ is given by

$$R_{\tilde{\Theta}|\Theta} = \text{cov}(\Theta - \hat{\Theta}(Y)|\Theta = \theta) \geq \mathcal{I}^{-1} \tag{3.9}$$

where $\mathcal{I}$ the $N_\theta \times N_\theta$ *information matrix* defined by

$$\mathcal{I} := -E_y \left\{ \nabla_\Theta \left( \nabla_\Theta \ln \Pr(Y|\Theta) \right)' \right\} \tag{3.10}$$

with the *gradient vector* $\nabla_\Theta \in R^{N_\theta \times 1}$ defined by

$$\nabla_\Theta := \left[ \frac{\partial}{\partial \Theta_1} \cdots \frac{\partial}{\partial \Theta_{N_\theta}} \right]' \tag{3.11}$$

Any estimator satisfying the *CRB* bound with equality is called *efficient*. The bound is easily calculated using the *chain rule* from vector calculus [7] defined by

$$\nabla_\Theta (a'b) := (\nabla_\Theta a')b + (\nabla_\Theta b')a, \qquad a, b \in \mathcal{R}^{N_\theta \times 1} \tag{3.12}$$

where $a, b$ are functions of $\Theta$. Consider the following example illustrating the calculation of the *CRB* bound.

***Example 3.1*** Suppose that we would like to estimate a nonrandom but unknown parameter, $\Theta$, from a measurement $y$ contaminated by additive gaussian noise, that is,

$$y = \Theta + v$$

where $v \sim \mathcal{N}(0, R_{vv})$ and $\Theta$ is unknown. Thus we have that

$$E\{Y|\Theta\} = E\{\Theta + v|\Theta\} = \Theta$$

and

$$\text{var}(Y|\Theta) = E\{(y - E\{Y|\Theta\})^2|\Theta\} = E\{v^2|\Theta\} = R_{vv}$$

This gives

$$\Pr(Y|\Theta) \sim \mathcal{N}(\Theta, R_{vv})$$

and therefore

$$\ln \Pr(Y|\Theta) = -\frac{1}{2} \ln 2\pi \sqrt{R_{vv}} - \frac{1}{2} \frac{(y - \Theta)^2}{R_{vv}}$$

Differentiating according to the chain rule of Eq. (3.12) and taking the expectation, we obtain

$$\mathcal{I} = -E \left\{ \frac{\partial^2}{\partial \Theta^2} \ln \Pr(Y|\Theta) \right\} = -E \left\{ \frac{\partial}{\partial \Theta} \frac{(y - \Theta)}{R_{vv}} (-1) \right\} = \frac{1}{R_{vv}}$$

[2]We choose the matrix-vector version, since parameter estimators are typically vector estimates.

Therefore the *CRB* bound is

$$R_{\tilde{\Theta}|\Theta} \geq R_{vv}$$

The utility of the *CRB* is twofold: (1) it enables us to measure estimator quality because it indicates the "best" (minimum error covariance) that any estimator can achieve, and (2) it allows us to decide whether or not the designed estimator is efficient, a desirable statistical property.

In summary, the properties of an estimator can be calculated prior to estimation (in some cases) and these properties can be used to answer the question: How well does this estimator perform? In subsequent sections we investigate some popular estimators and their associated performance.

## 3.2 MINIMUM VARIANCE (*MV*) ESTIMATION

In this section we review the concepts of minimum variance estimation and use it as a foundation to develop both Bayesian and likelihood estimators. The general techniques developed will be applied to develop various model-based processors in subsequent sections.

Suppose that we are asked to obtain an estimate of a $N_\theta$-parameter vector $\Theta$ from a set of noisy measurements characterized by the $N_y$-measurement vector $\mathbf{Y}$. We would like to construct an estimate that is best or optimal in some sense. The most natural criterion to consider is one that minimizes the error between the true parameter and its estimate based on the measured data. The *error variance criterion* defined by

$$J(\Theta) = E_\Theta \left\{ [\Theta - \hat{\Theta}(Y)]'[\Theta - \hat{\Theta}(Y)] | Y \right\} \tag{3.13}$$

where

$\Theta$ = the true random $N_\theta$-vector

$Y$ = the measured random $N_y$-vector (data)

$\hat{\Theta}$ = the estimate of $\Theta$ given $Y$

whose minimization leads to the *minimum variance estimator* [8]. Thus, if we minimize $J(\Theta)$ using the chain rule of Eq. (3.12), then we have that

$$\begin{aligned}
\nabla_{\hat{\Theta}} J(\Theta) &= E_\theta \{ \nabla_{\hat{\Theta}} \left( \Theta - \hat{\Theta}(Y) \right)' \left( \Theta - \hat{\Theta}(Y) \right) | Y \} \\
&= E_\theta \{ - \left( \Theta - \hat{\Theta}(Y) \right) - \left( \Theta - \hat{\Theta}(Y) \right) | Y \} \\
&= -2 \left[ E_\theta \{ \Theta - \hat{\Theta}(Y) | Y \} \right]
\end{aligned}$$

Performing the conditional expectation operation gives

$$\nabla_{\hat{\Theta}} J(\Theta) = -2 \left[ E_\theta \{ \Theta | Y \} - \hat{\Theta}(Y) \right] \tag{3.14}$$

Setting this equation to zero and solving yields the minimum variance estimate as

$$\hat{\Theta}_{MV} = \hat{\Theta}(Y) = E_\theta\{\Theta|Y\} \tag{3.15}$$

We check for the global minimum by examining the sign (positive) of the second derivative, since

$$\nabla_{\hat{\Theta}} \left(\nabla_{\hat{\Theta}} J(\Theta)\right)' = 2I \tag{3.16}$$

indicating the unique minimum. Thus the minimum variance estimator is the *conditional mean*. The associated error variance is determined by substituting for $\hat{\Theta}(Y)$ into Eq. (3.13), giving

$$J_{MV} = R_{\tilde{\Theta}|Y} = E_\theta\{(\Theta - E\{\Theta|Y\})'(\Theta - E\{\Theta|Y\})\,|Y\} = E_\theta\{\Theta'\Theta|Y\} - E_\theta^2\{\Theta|Y\} \tag{3.17}$$

The minimum variance estimator can be summarized as follows:

Criterion: $J = E\{\tilde{\Theta}'\tilde{\Theta}\}$
Model: $\Pr(\Theta|Y)$
Algorithm: $\hat{\Theta}_{MV} = E_\theta\{\Theta|Y\}$
Quality: $R_{\tilde{\Theta}|Y}$

Note also that this estimator is linear, unconditionally and conditionally unbiased and possesses general orthogonality properties. To see this, we investigate the estimation error defined by

$$\tilde{\Theta} = \Theta - \hat{\Theta}(Y) \tag{3.18}$$

Taking expectations, we have the fact that the estimator is unconditionally unbiased

$$E_\theta\{\tilde{\Theta}\} = E_\theta\{\Theta - \hat{\Theta}(Y)\} = E_\theta\{\Theta\} - E_\theta\{E_\theta\{\Theta|Y\}\} = E_\theta\{\Theta\} - E_\theta\{\Theta\} = 0 \tag{3.19}$$

as well as conditionally unbiased, since

$$\begin{aligned} E_\theta\{\tilde{\Theta}|Y\} &= E_\theta\{\Theta - \hat{\Theta}(Y)|Y\} = E_\theta\{\Theta|Y\} - E_\theta\{E_\theta\{\Theta|Y\}|Y\} \\ &= E_\theta\{\Theta|Y\} - E_\theta\{\Theta|Y\} = 0 \end{aligned} \tag{3.20}$$

Another important property of the minimum variance estimator is that the estimation error is orthogonal to *any* function, say $f(\cdot)$, of the data vector $Y$ [9], that is,

$$E_{\theta Y}\{f(Y)\tilde{\Theta}'\} = 0 \tag{3.21}$$

Also

$$E_\theta\{f(Y)\tilde{\Theta}'|Y\} = 0 \tag{3.22}$$

This is the well-known orthogonality condition. To see this, we substitute for the error

$$\begin{aligned} E_\theta\{f(Y)\tilde{\Theta}'|Y\} &= E_\theta\{f(Y)\left(\Theta - \hat{\Theta}(Y)\right)'|Y\} \\ &= f(Y)E_\theta\{\left(\Theta - \hat{\Theta}(Y)\right)'|Y\} \\ &= f(Y)\left(E_\theta\{\Theta|Y\} - \hat{\Theta}(Y)'\right) = 0 \end{aligned}$$

Taking the expectation over $Y$ proves the unconditional result as well. Thus the estimation error is orthogonal to *any* function of $Y$, a fact that is used in proofs throughout the literature for both linear and nonlinear estimators.

Let us now investigate the special case of the *linear* minimum variance estimator. The *estimation error* is *orthogonal* to all past data $Y$, that is,

$$E_{\Theta Y}\{Y\tilde{\Theta}'\} = 0 \tag{3.23}$$

or as before

$$E_\Theta\{Y\tilde{\Theta}'|Y\} = 0 \tag{3.24}$$

This is the well-known minimum variance estimator results in the linear case [2]. For a linear function of the parameter, we have that

$$y = C\Theta + v \tag{3.25}$$

where $y, v, \in \mathcal{R}^{N_y \times 1}$, $\Theta \in \mathcal{R}^{N_\theta \times 1}$, $C \in \mathcal{R}^{N_y \times N_\theta}$ and $v$ is zero-mean, white with $R_{vv}$. The mean-squared error criterion

$$J(\Theta) = E\{\tilde{\Theta}'\tilde{\Theta}\} \tag{3.26}$$

is minimized to determine the estimate. The minimization results in the orthogonality condition of Eq. (3.24), which we write as

$$E\{y\tilde{\Theta}'\} = E\{y\Theta'\} - E\{y\hat{\Theta}'_{MV}\} = 0 \tag{3.27}$$

for $\hat{\Theta}_{MV} = K_{MV}y$, a linear function of the data vector. Substituting for $y$ and $\hat{\Theta}$ in this equation gives

$$K_{MV} = R_{\Theta\Theta}C'(CR_{\Theta\Theta}C' + R_{vv})^{-1} \tag{3.28}$$

The corresponding quality is obtained as

$$R_{\tilde{\Theta}\tilde{\Theta}} = (R^{-1}_{\Theta\Theta} + C'R^{-1}_{vv}C)^{-1} \tag{3.29}$$

So we see that the minimum variance estimator for the linear case is as follows:

Criterion: $J = E\{\tilde{\Theta}'\tilde{\Theta}\}$
Model: $y = C\Theta$
Algorithm: $\hat{\Theta}_{MV} = K_{MV}y$
Quality: $R_{\tilde{\Theta}\tilde{\Theta}}$

It is also interesting to note that the fundamental Wiener result [7] is easily obtained from the orthogonality condition of Eq. (3.24), that is,

$$E\{y\tilde{\Theta}'\} = E\{y\Theta'\} - E\{yy'\}K'_{MV} = R_{y\Theta} - R_{yy}K'_{MV} = 0 \tag{3.30}$$

which is called the *discrete Wiener-Hopf equation*. Solving for $K_{MV}$, we obtain the Wiener solution for a linear (batch) estimation scheme, that is,

$$K_{MV} = R_{\Theta y}R_{yy}^{-1} \tag{3.31}$$

We also mention in passing that *least-squares estimation* is similar to that of minimum variance except that no statistical information is assumed known about the process. That is, the least-squares estimator, $\hat{y}_{LS}$ minimizes the squared error criterion

$$\min_{\hat{y}} J = \tilde{y}'\tilde{y} \tag{3.32}$$

for $\tilde{y} = y - \hat{y}_{LS}$.

In contrast to the least-squares approach requiring *no* statistical information, we introduce two popular estimators: the "most probable" or maximum a posteriori (*MAP*) estimator and the maximum likelihood (*ML*) estimator, which is a special case of *MAP*. We show their relationship and analyze their performance.

### 3.2.1 Maximum a Posteriori (*MAP*) Estimation

Suppose that we are trying to estimate a random parameter, say $\Theta$, from data $Y = y$. Then the associated conditional density $\Pr(\Theta|Y = y)$ is called the *posterior density* because the estimate is conditioned "after (*post*) the measurements" have been acquired. Estimators based on the a posteriori density are usually called *Bayesian* because they are constructed from Bayes' rule, since $\Pr(\Theta|Y)$ is difficult to obtain directly. That is,

$$\Pr(\Theta|Y) = \Pr(Y|\Theta)\frac{\Pr(\Theta)}{\Pr(Y)} \tag{3.33}$$

where $\Pr(\Theta)$ is called the *prior density* (before measurement), $\Pr(Y|\Theta)$ is called the likelihood (more likely to be true), and $\Pr(Y)$ is called the *evidence* (scales the posterior to assure its integral is unity). Bayesian methods view the sought after parameter as random possessing a "known" a priori density. As measurements are made,

the *prior* is converted to the *posterior density* function adjusting the parameter estimates. Thus the result of increasing the number of measurements is to improve the a posteriori density resulting in a sharper peak closer to the true parameter.

To solve the estimation problem, the first step requires the determination of the a posteriori density. A logical solution to this problem leads us to find the "most probable" value of $\Pr(\Theta|Y)$ that is, its *maximum*. The *maximum a posteriori* (*MAP*) *estimate* is the value that yields the maximum value of the a posteriori density. The optimization is carried out in the usual manner by

$$\nabla_{\Theta}\Pr(\Theta|Y)\Big|_{\Theta=\hat{\Theta}_{MAP}} = 0 \tag{3.34}$$

Because many problems are based the exponential class of densities, the $\ln\Pr(\Theta|Y)$ is considered instead. Since the logarithm is a monotonic function, the maximum of $\Pr(\Theta|Y)$ and $\ln\Pr(\Theta|Y)$ occur at the same value of $\Theta$. Therefore the *MAP equation* is

$$\nabla_{\Theta}\ln\Pr(\Theta|Y)\Big|_{\Theta=\hat{\Theta}_{MAP}} = 0 \tag{3.35}$$

Now, if we apply Bayes' rule to Eq. (3.33), then

$$\ln\Pr(\Theta|Y) = \ln\Pr(Y|\Theta) + \ln\Pr(\Theta) - \ln\Pr(Y) \tag{3.36}$$

Since $\Pr(Y)$ is not a function of the parameter $\Theta$, the *MAP* equation can be written

$$\nabla_{\Theta}\ln\Pr(\Theta|Y)\Big|_{\Theta=\hat{\Theta}_{MAP}} = \nabla_{\Theta}(\ln\Pr(Y|\Theta) + \ln\Pr(\Theta))\Big|_{\Theta=\hat{\Theta}_{MAP}} = 0 \tag{3.37}$$

Of course, the sign of the second derivative must be checked to ensure that a global maximum is obtained. It can be shown that for a variety of circumstances (cost function convex, symmetrical) the minimum variance estimate of the previous section is in fact identical to the *MAP* estimate [2]. Next we consider the case when the parameter $\Theta$ is not random leading to the maximum likelihood estimator.

### 3.2.2 Maximum Likelihood (*ML*) Estimation

In contrast to the Bayesian approach, the likelihood method views the parameter as deterministic but *unknown*. It produces the "best" estimate as the value which maximizes the probability of the measurements given that the parameter value is "most likely" true. In the estimation problem the measurement data are given along with the underlying structure of the probability density function (as in the Bayesian case), but the parameters of the density are unknown and must be determined from the measurements. Therefore the maximum likelihood estimate can be considered heuristically as that value of the parameter that best "explains" the measured data giving the most likely estimation.

More formally, let $\Theta$ be a vector of unknown parameters, $\Theta \in \mathcal{R}^{N_\theta \times 1}$ and the corresponding set of $N$-independent measurements, $Y(N) := \{\mathbf{y}(1)\cdots\mathbf{y}(N)\}$

for $\mathbf{y} \in \mathcal{R}^{N_y \times 1}$. The *likelihood* of $\Theta$, given the measurements is defined to be proportional to the value of the probability density of the measurements given the parameters, that is,

$$\mathcal{L}(Y(N); \Theta) \propto \Pr(Y(N)|\Theta) = \Pr(\mathbf{y}(1) \cdots \mathbf{y}(N)|\Theta) = \Pi_{i=1}^{N} \Pr(\mathbf{y}(i)|\Theta) \quad (3.38)$$

where $\mathcal{L}$ is the likelihood function and $\Pr(Y|\Theta)$ is the joint probability density functions of the measurements given the unknown parameters. This expression indicates the usefulness of the likelihood function in the sense that in many applications measurements are available and are assumed drawn as a sample from a "known" or assumed known probability density function with unknown parameters (e.g., Poisson with unknown mean). Once we have the measurements (given) and the likelihood function, then we would like to find the best estimate of the parameters. If we search through parameter space over various values of $\Theta$, say $\Theta_i$, then we select the value of $\hat{\Theta}$ that most likely (most probable) specifies the underlying probability function that the measurement sample was drawn from. That is, suppose that we have two estimates, $\hat{\Theta}_i$ and $\hat{\Theta}_j$ for which

$$\Pr(Y|\Theta_i) > \Pr(Y|\Theta_j) \quad (3.39)$$

Thus it is "more likely" that the $Y(N)$ were drawn for parameter value $\hat{\Theta}_i$, then $\hat{\Theta}_j$, since equivalently $\mathcal{L}(Y; \hat{\Theta}_i) > \mathcal{L}(Y; \hat{\Theta}_j)$. Searching over all $\Theta$ and selecting that value of $\Theta$ that is maximum (most probable) leads to the maximum likelihood estimate (*ML*) given by

$$\hat{\Theta}_{ML}(Y) = \arg\max_{\Theta} \Pr(Y|\Theta) \quad (3.40)$$

As noted before for the Bayesian estimator, many problems are characterized by the class of exponential densities making it simpler to use the natural logarithm function. Therefore we define the *log-likelihood function* as

$$\Lambda(Y(N)|\Theta) := \ln \mathcal{L}(Y; \Theta) = \ln \Pr(Y(N)|\Theta) \quad (3.41)$$

Since the logarithm is monotonic, it preserves the maximum of the likelihood providing the identical result, that is,

$$\hat{\Theta}_{ML}(Y) = \arg\max_{\Theta} \ln \Pr(Y|\Theta) \quad (3.42)$$

What makes the *ML* estimator popular is the fact that it enjoys some very desirable properties, which we that we list without proof (see [5] for details):

1. *ML* estimates are *consistent*.
2. *ML* estimates are *asymptotically efficient* with $R_{\tilde{\Theta}|\Theta} = \mathcal{I}^{-1}$.

3. *ML* estimates are *asymptotically gaussian* with $\mathcal{N}(\Theta, R_{\tilde{\Theta}\tilde{\Theta}})$.
4. *ML* estimates are *invariant*, that is, if $\hat{\Theta}_{ML}$, then any function of the *ML* estimate is the *ML* estimate of the function, $\hat{f}_{ML} = f\left(\hat{\Theta}_{ML}\right)$
5. *ML* estimates of the *sufficient statistic* are equivalent to the *ML* estimates over the original data.

These properties are asymptotic and therefore imply that a large amount of data must be available for processing.

Mathematically the relationship between the *MAP* and *ML* estimators is clear even though philosophically they are quite different in construct. If we take the *MAP* equation of Eq. (3.33) and ignore the a priori density $\Pr(\Theta)$ (assume $\Theta$ is unknown but deterministic), then the maximum likelihood estimator is only a special case of *MAP*. Using the same arguments as before, we use the $\ln \Pr(\Theta|Y)$ instead of $\Pr(\Theta|Y)$. We obtain the maximum likelihood estimate by solving the *log-likelihood equation* and checking for the existence of a maximum; that is,

$$\nabla_\Theta \ln \Pr(\Theta|Y)\Big|_{\Theta=\hat{\Theta}_{ML}} = 0 \tag{3.43}$$

where recall that the $\Pr(\Theta|Y)$ is the likelihood of $\Theta$. Of course, to check for a maximum we have that $\nabla_\Theta(\nabla_\Theta \ln \Pr(\Theta|Y)') < 0$. Again applying Bayes' rule as in Eq. (3.33) and ignoring $\Pr(\Theta)$, we have

$$\nabla_\Theta \ln \Pr(\Theta|Y) = \nabla_\Theta \ln \Pr(Y|\Theta)\Big|_{\Theta=\hat{\Theta}_{ML}} \tag{3.44}$$

Consider the following example to demonstrate this relationship between *MAP* and *ML*.

***Example 3.2*** Consider estimating an unknown *constant*, from a noisy measurement as in the previous example. Further assume that the noise is an independent gaussian random variable such that $v \sim \mathcal{N}(0, R_{vv})$ and the measurement model is given by

$$y = \Theta + v \tag{3.45}$$

First, we assume no a priori statistical knowledge of $\Theta$ just that it is an unknown, nonrandom constant. Thus we require the maximum likelihood estimate, since no prior information is assumed about $\Pr(\Theta)$. The associated conditional density is

$$\Pr(Y|\Theta) = \frac{1}{\sqrt{2\pi R_{vv}}} e^{-\frac{1}{2}\frac{(y-\Theta)^2}{R_{vv}}} \tag{3.46}$$

The maximum likelihood estimate of $\Theta$ is found by solving the log-likelihood equation:

$$\nabla_\Theta \ln \Pr(Y|\Theta)\Big|_{\Theta=\hat{\Theta}_{ML}} = 0 \tag{3.47}$$

or

$$\frac{\partial}{\partial\Theta}\ln\Pr(Y|\Theta) = \frac{\partial}{\partial\Theta}\left\{-\frac{1}{2}\ln 2\pi R_{vv} - \frac{1}{2R_{vv}}(y-\Theta)^2\right\}$$
$$= \frac{1}{R_{vv}}(y-\Theta)$$

Setting this expression to zero and solving for $\Theta$, we obtain

$$\hat{\Theta}_{ML} = y \tag{3.48}$$

That is, the best estimate of $\Theta$ in a maximum likelihood sense is the raw data $y$. The corresponding error variance is easily calculated (as before)

$$R_{\tilde{\Theta}|\Theta} = R_{vv} \tag{3.49}$$

Next we model $\Theta$ as a random variable with gaussian prior; that is, $\Theta \sim \mathcal{N}(\overline{\Theta}, R_{\Theta\Theta})$ and desire the maximum a posteriori estimate. The *MAP* equation is

$$\mathcal{J}_{MAP} = \nabla_\Theta(\ln\Pr(Y|\Theta) + \ln\Pr(\Theta))$$
$$\mathcal{J}_{MAP} = \frac{\partial}{\partial\theta}\left\{-\frac{1}{2}\ln 2\pi R_{vv} - \frac{1}{2R_{vv}}(y-\Theta)^2 - \frac{1}{2}\ln 2\pi R_{\Theta\Theta} - \frac{1}{2R_{\Theta\Theta}}(\Theta-\overline{\Theta})^2\right\}$$

or

$$\mathcal{J}_{MAP} = \frac{1}{R_{vv}}(y-\Theta) - \frac{1}{R_{\Theta\Theta}}(\Theta-\overline{\Theta})$$

Setting this expression to zero and solving for $\Theta = \hat{\Theta}_{MAP}$, we obtain

$$\hat{\Theta}_{MAP} = \frac{y + \dfrac{R_{vv}}{R_{\Theta\Theta}}\overline{\Theta}}{1 + \dfrac{R_{vv}}{R_{\Theta\Theta}}}$$

It can be shown that the corresponding error variance is

$$R_{\tilde{\Theta}|\Theta} = \frac{R_{vv}}{1 + \dfrac{R_{vv}}{R_{\Theta\Theta}}}$$

From the results of this example, we see that when the parameter variance is large ($R_{\Theta\Theta} >> R_{vv}$), the *MAP* and *ML* estimates perform equivalently. However, when the variance is small, the *MAP* estimator performs better because the corresponding error variance is smaller. This completes the example.

The main point to note is that the *MAP* estimate provides a mechanism to incorporate the a priori information, while the *ML* estimate does not. Therefore, for

some problems, *MAP* is the efficient estimator. In this example, if $\Theta$ were actually gaussian, then the *ML* solution, which models $\Theta$ as an unknown parameter, is not an efficient estimator, while the *MAP* solution incorporates this information by using $\Pr(\Theta)$.

This completes the introduction to minimum variance, maximum a posteriori, and maximum likelihood estimation, next we consider the nonstatistical approach.

## 3.3 LEAST-SQUARES (*LS*) ESTIMATION

In this section we discuss least-squares (*LS*) estimation techniques and relate them to the minimum variance estimator discussed previously. Weighted *LS* estimators in the strict sense are estimators that minimize the sum-squared error criterion

$$\mathcal{J}_{LS} = \frac{1}{2}\tilde{y}' W^{-1}\tilde{y} \tag{3.50}$$

where $\tilde{y} := y - \hat{y}$, $y \in \mathcal{R}^{N_y \times 1}$, $W \in \mathcal{R}^{N_y \times N_y}$ is the symmetric, positive definite, *weighting matrix*. No probabilistic or statistical description of the estimation problem is assumed. In fact we can consider the solution to the least-squares estimation problem to be a "deterministic" optimization problem.

### 3.3.1 Batch Least Squares

In this section we investigate "batch" solutions to the least-squares estimation problem. We consider the specific problem of estimating an $N_\theta$-parameter vector $\Theta$ from data and assume that the *LS* estimator of $\Theta$ is linearly related to the *LS* estimate of $y$ by

$$\hat{Y}_{LS} = C\hat{\Theta}_{LS} \tag{3.51}$$

where $C \in \mathcal{R}^{N_y \times N_\theta}$ is a full rank matrix, that is, $\rho(C) = N_\theta$, $N_\theta < N_y$. The *LS* estimator is found by minimizing $\mathcal{J}_{LS}$ with respect to $\Theta$, by taking the derivative (gradient), setting the result to zero, and solving for $\Theta = \hat{\Theta}_{LS}$; that is,

$$\nabla_\Theta \mathcal{J}_{LS}\Big|_{\Theta=\hat{\Theta}_{LS}} = \nabla_\Theta \left(\tfrac{1}{2}\tilde{y}' W^{-1}\tilde{y}\right) = \tfrac{1}{2}\nabla_\Theta (y - C\Theta)' W^{-1}(y - C\Theta) = 0 \tag{3.52}$$

Applying the chain rule of Eq. (3.12) with $a' = (y - C\Theta)$ and $b = W^{-1}(y - C\Theta)$, we obtain

$$\nabla_\Theta \mathcal{J}_{LS} = \tfrac{1}{2}\left[-C' W^{-1}(y - C\Theta) - C' W^{-1}(y - C\Theta)\right] = -C' W^{-1}(y - C\Theta) = 0$$

Solving this equation for $\hat{\Theta}$, we obtain the *weighted least-squares estimate*:

$$\hat{\Theta}_{LS} = K_{LS} y = \left(C' W^{-1} C\right)^{-1} C' W^{-1} y \tag{3.53}$$

If we further assume additive measurement noise with variance $R_{vv}$, then

$$y = C\theta + v \tag{3.54}$$

and therefore

$$R_{\tilde{\Theta}\tilde{\Theta}} = \text{cov } \tilde{\Theta} = K_{LS} R_{vv} K'_{LS} \tag{3.55}$$

To investigate "how good" the least-squares estimator is for this problem, we must calculate the *CRB*—assuming some prior statistics for the underlying process. First, we assume that the noise is $v \sim \mathcal{N}(0, R_{vv})$. Then $\Pr(Y|\Theta) \sim \mathcal{N}(C\Theta, R_{vv})$ and

$$\ln \Pr(Y|\Theta) = -\tfrac{1}{2} \ln (2\pi)^N |R_{vv}|^{-1/2} - \tfrac{1}{2}(y - C\Theta)' R_{vv}^{-1}(y - C\Theta)$$

Again, using the chain rule with $a' = (y - C\Theta)'$ and $b = R_{vv}^{-1}(y - C\Theta)$, we have that the information matrix $\mathcal{I}$ is

$$\mathcal{I} = -E\{\nabla_\theta (\nabla_\theta \ln \Pr(Y|\Theta))'\} = -\nabla_\theta (C' R_{vv}^{-1}(y - C\Theta)) = \left(C' R_{vv}^{-1} C\right)$$

Thus the *CRB* is given by

$$R_{\tilde{\Theta}|\Theta} = \left(C' R_{vv}^{-1} C\right)^{-1}$$

Note also in passing that if we solve for the minimum variance estimator (set gradient to zero), then we obtain

$$\hat{\Theta}_{MV} = \left(C' R_{vv}^{-1} C\right)^{-1} C' R_{vv}^{-1} y$$

Comparing this result with the least-squares estimator, we see that the weighted *LS* estimator is identical to the *MV* estimator, if the weighting matrix $W = R_{vv}$, that is, both estimators satisfy the *CRB* with *equality* (efficient) for estimating a random vector from noisy data. To see this substitute, $W = R_{vv}$ for $K_{LS}$ in Eq. (3.55) to obtain

$$R_{\tilde{\theta}\tilde{\theta}} = \left[(C' R_{vv}^{-1} C)^{-1} C' R_{vv}^{-1} R_{vv} R_{vv}^{-1} C (C' R_{vv}^{-1} C)^{-1}\right] = (C' R_{vv}^{-1} C)^{-1} \tag{3.56}$$

This estimator is also considered a "batch" estimator, since it processes all of the $N_y$-measurements ($C$ is $N_y \times N_\theta$) in one batch to obtain the estimate. Other techniques to solve this problem sequentially or recursively yield the identical result but update the estimate only as a new data sample becomes available. Note that the batch approach would require the entire problem to be recalculated each time as the next measurement becomes available, that is, $C$ becomes a $(N_y + 1) \times N_\theta$ matrix.

We summarize the batch *weighted least-squares (WLS) estimator* as follows, *assuming that* $y = C\Theta + v$ and cov $v = R_{vv}$:

Criterion: $J_{WLS} = \frac{1}{2} \tilde{y}' W^{-1} \tilde{y}$

Models: $y = C\Theta$
Algorithm: $\hat{\Theta}_{LS} = K_{LS}y$
Quality: $R_{\tilde{\Theta}\tilde{\Theta}} = K_{LS}R_{vv}K'_{LS}$

Finally we consider the batch weighted *LS* estimator for a common estimation problem—trend removal.

***Example 3.3*** Consider the problem of removing a trend from noisy measurement data. A typical problem application is tracking a flight vehicle using a radar system [10] and compensating for the known ballistic trajectory. The basic motion compensation problem is that of removing the trajectory or trend from the measured data. In general, suppose that we are given $N$ measurements and would like to develop the compensation algorithm. Our underlying scalar measurement model is

$$y(t) = s(t) + \Theta(t) + n(t)$$

where $\Theta(t)$ is the unknown trajectory (trend) and $y$, $s$, and $n$ are the respective measurement, signal and random noise. The *trend estimation* problem is to estimate the trend, $\hat{\Theta}(t)$ and remove it from the data. Generally, a temporal polynomial is used as the model, that is,

$$\Theta(t) = \theta_0 + \theta_1 t + \cdots + \theta_{N_\theta - 1} t^{N_\theta - 1} = \mathbf{c}'(t)\theta$$

where $\mathbf{c}'(t) = [1\ t \cdots t^{N_\theta - 1}]$ and $\theta = [\theta_0\ \theta_1 \cdots \theta_{N_\theta - 1}]'$. The measurement model at a given time, $t$, is therefore

$$y(t) = s(t) + \Theta(t) + n(t) = s(t) + \mathbf{c}'(t)\theta + n(t)$$

If we expand over the $N_y$ measurements, we obtain

$$\mathbf{y} = \mathbf{s} + C(t)\theta + \mathbf{n}$$

with $\mathbf{y} = [y(1) \cdots y(N_y)]'$, $\mathbf{s} = [s(1) \cdots s(N_y)]'$, and $\mathbf{n} = [n(1) \cdots n(N_y)]'$ for the $N_y \times N_\theta$ measurement matrix, $C(t) = [\mathbf{c}(1) \cdots \mathbf{c}(N_\theta)]'$, $\mathbf{c} \in \mathcal{R}^{N_y \times 1}$. For this problem

$$J_{LS} = \tfrac{1}{2} (\mathbf{y} - \hat{\mathbf{y}})' W^{-1} (\mathbf{y} - \hat{\mathbf{y}})$$

and the *WLS* estimator is of the form

$$\hat{Y}_{LS} = C\hat{\Theta}_{LS}$$

From Eq. (3.53) we obtain the *WLS* estimator

$$\hat{\theta}_{LS} = K_{LS}\mathbf{y} = \left(C'W^{-1}C\right)^{-1} C'W^{-1}\mathbf{y}$$

Therefore

$$\hat{\Theta}(t) = \mathbf{c}'(t)\hat{\theta}_{LS}$$

Now, if we assume that $n$ is zero-mean, white with covariance, $R_{nn}$, then the quality of the estimator is

$$R_{\tilde{\theta}\tilde{\theta}} = K_{LS} R_{nn} K'_{LS}$$

Also note (as before) that if we choose the weighting matrix to be $W = R_{nn}$, then we obtain the minimum variance estimate. Once $\hat{\Theta}(t)$ is estimated, it can be removed from the measurement data to obtain

$$z(t) = y(t) - \hat{\Theta}(t) = s(t) + \left[\Theta(t) - \hat{\Theta}(t)\right] + n(t) \approx s(t) + n(t)$$

Now in terms of our motion compensation application [10], our polynomial model has *physical* interpretation as

$$\Theta(t) = R_o + v_R t + a_R t^2 = [1 \quad t \quad t^2] \begin{bmatrix} R_o \\ v_R \\ a_R \end{bmatrix} = \mathbf{c}'(t)\theta$$

with $R_o$ the nominal range, $v_R$ the velocity and $a_R$ the acceleration of the vehicle. We show the results of applying the *WLS* estimator to measured trajectory data in Figure 3.2*b*. Here we see the effect is to remove the ballistic trajectory from the measurement enabling further processing to extract the pertinent signals.

Summarizing, we have:

| | |
|---|---|
| Criterion: | $J_{LS} = \frac{1}{2}(\mathbf{y} - \hat{\mathbf{y}})' W^{-1} (\mathbf{y} - \hat{\mathbf{y}})$ |
| Models: | |
| Measurement: | $\mathbf{y} = \mathbf{s} + C(t)\theta + \mathbf{n}$ |
| Signal: | $s(t)$ |
| Noise: | $n(t)$ and cov $\mathbf{n} = R_{nn}$ |
| Trend: | $\Theta(t) = \mathbf{c}'(t)\theta$ |
| Algorithm: | $\hat{\Theta}_{LS} = K_{LS}\mathbf{y}$ for $K_{LS} = \left(C'W^{-1}C\right)^{-1} C'W^{-1}$ |
| Quality: | $R_{\tilde{\Theta}\tilde{\Theta}} = K_{LS} R_{nn} K'_{LS}$ |

### 3.3.2 *LS*: A Geometric Perspective

The richness of the least-squares estimation problem is sometimes overlooked. It provides an interesting and full description of optimal signal processing techniques which have been exploited throughout the literature, especially in model-based approaches ([7], [11]) and many of the array signal processing methods ([12], [13], [14]) involving subspace processing. In this section we revisit the *batch least-squares* problem and investigate it from the geometric perspective using vector spacemethods. The results presented will be used throughout the text to develop some of the model-based processing schemes.

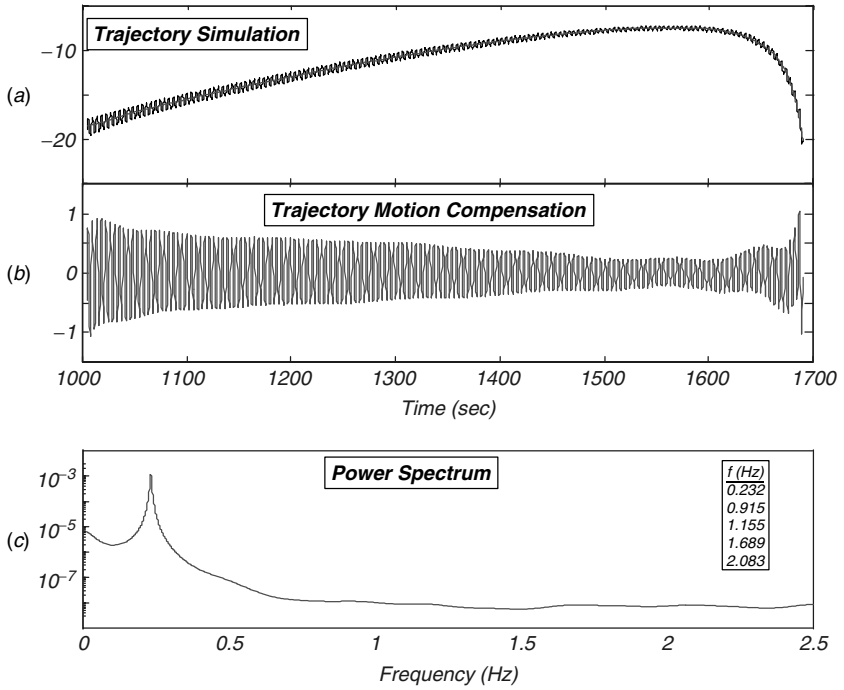



**Figure 3.2.** Trajectory motion compensation using the *WLS* estimator: (*a*) Raw data and *WLS* trajectory estimate. (*b*) Compensated (trend-removed) data. (*c*) Power spectral density estimate.

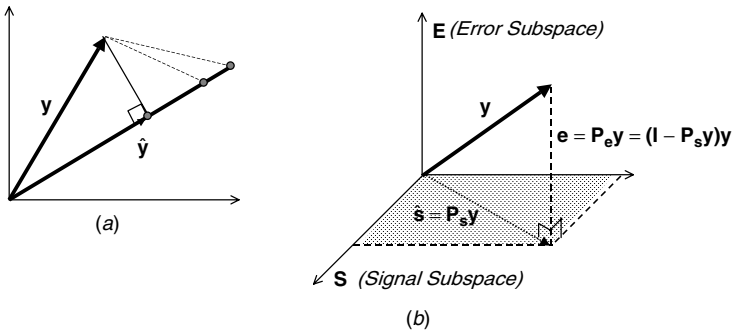



**Figure 3.3.** Geometric perspective of least squares: (*a*) Projection of point $\mathbf{y}$ onto line $\hat{\mathbf{y}}$. (*b*) *LS* estimation.

Perhaps the simplest way to motivate the geometric approach (as in [15]) is to consider the problem that we are given a vector $\mathbf{y}$ in an $N_y$-dimensional vector space whose tip defines a point, and we would like to find the distance from that point to a line that lies in a certain direction defined by the vector $\hat{\mathbf{y}}$ as shown in Figure 3.3*a*. We search along this line (vector) for the point closest to $\mathbf{y}$. We know

from geometry that the solution is a perpendicular line drawn from the tip of $\mathbf{y}$ or equivalently the point to the line defined by $\hat{\mathbf{y}}$. Knowledge of this fact allows us to find not only the point on the line defining $\hat{\mathbf{y}}$ but also its distance from the tip of $\mathbf{y}$. If we replace the line with a plane or more generally a subspace $\mathcal{S}$, then the problem becomes that of finding the location on the subspace that is "closest to $\mathbf{y}$." The intersection point defines the vector $\hat{\mathbf{y}}$, which is called the *projection* of $\mathbf{y}$ onto $\mathcal{S}$.

Now let us assume that the $N_y$-dimensional data space, $\mathcal{Y}$, is decomposed into two orthogonal subspaces, $\mathcal{S}$ and $\mathcal{E}$ of respective dimensions $N_s$ and $N_e$ [15] such that

$$\mathcal{Y} = \mathcal{S} \oplus \mathcal{E}$$

Therefore any vector in $\mathcal{Y}$, $\mathbf{y} \in \mathcal{R}^{N_y \times 1}$ can be decomposed as

$$\mathbf{y} = P_s\mathbf{y} + P_e\mathbf{y} \quad \text{for} \quad \mathcal{S} \perp \mathcal{E} \tag{3.57}$$

where $P_s\mathbf{y}$ represents the *projection* with $P_s$ the corresponding *projection matrix* of $\mathbf{y}$ onto the subspace $\mathcal{S}$. This subspace is defined as the "signal" subspace, while $P_e\mathbf{y}$ is the projection with $P_e$ the corresponding projection matrix of $\mathbf{y}$ onto the orthogonal "error" or "noise" subspace $\mathcal{E}$ as shown in Figure 3.3*b*.

Since this is an *orthogonal decomposition*, the orthogonal projection matrices have the following properties:

1. $P_s = P_s'$, $P_e = P_e'$ (symmetric)
2. $P_s = P_s^2$, $P_e = P_e^2$ (idempotent)
3. $P_e = I - P_s$ (identity)

Next let us apply the orthogonal decomposition and projection concepts to the batch least-squares problem. Recall the least-squares estimation problem of Eq. (3.50) with $W = I$ for simplicity. Here we are tasked with finding the best estimate of the signal defined by

$$\mathbf{s} = C\Theta = \sum_{i=1}^{N_\theta} \theta_i \mathbf{c}_i \qquad \text{for } C \in \mathcal{R}^{N_y \times N_\theta}$$

$$\Theta \in \mathcal{R}^{N_\theta \times 1}, \quad \mathbf{c}_i \in \mathcal{R}^{N_y \times 1}, \qquad i = 1, \ldots, N_\theta \tag{3.58}$$

where $\mathbf{c}_i$ are the set of $N_y$-column vectors of $C$ spanning the signal subspace. Another way of stating the same problem is to choose that value of $\Theta$ to minimize the square of the *error* or *noise* vector defined by

$$\mathbf{e} = \mathbf{y} - \mathbf{s} = \mathbf{y} - C\Theta \tag{3.59}$$

in the least-squares sense of Eq. (3.50). The solution to this problem then provides the closest point than any other point spanned by the column vectors of $C$. Thus minimizing the squared error is equivalent to finding the orthogonal projection of $\mathbf{y}$

onto the signal subspace defined spanned by the $\mathbf{c}_i$ vectors. Therefore we solve for the unknown parameter vector, $\Theta$, that defines the signal vector using orthogonal projection theory to gain the geometric insight we seek. In the least-squares problem we have that the projection of $\mathbf{y}$ onto $\mathcal{S}$ gives the signal vector

$$\hat{\mathbf{s}} = P_c\mathbf{y} = C\hat{\Theta}_{LS} = C\left(C'C\right)^{-1}C'\mathbf{y} \tag{3.60}$$

where $P_c := C\left(C'C\right)^{-1}C'$ is the orthogonal projection matrix defined on the signal subspace. Similarly the projection of $\mathbf{y}$ onto $\mathcal{E}$ (orthogonal complement) gives the error vector

$$\mathbf{e} = P_e\mathbf{y} = \mathbf{y} - \hat{\mathbf{s}} = \mathbf{y} - C\hat{\Theta}_{LS} = \mathbf{y} - P_c\mathbf{y} = (I - P_c)\,\mathbf{y} \tag{3.61}$$

We see that the orthogonal projection of the data $\mathbf{y}$ onto the signal subspace results in a linear transformation of the least-squares parameter estimate $\hat{\Theta}_{LS}$, while the orthogonal projection of the data vector onto the error subspace (orthogonal complement) results in the error vector. The orthogonality of the error vector to the signal vector defines the *orthogonality condition*

$$\mathbf{e}'\hat{\mathbf{s}} = \mathbf{0} \tag{3.62}$$

which states that the error vector is orthogonal to the space spanned by the signal vectors. To see this, we use the projection matrices and the error vector

$$\mathbf{e}'\hat{\mathbf{s}} = (P_e\mathbf{y})'(P_c\mathbf{y}) = \mathbf{y}'P_eP_c\mathbf{y} = \mathbf{y}'(I - P_c)P_c\mathbf{y} = \mathbf{y}'(P_c - P_c^2)(P_c - P_c^2)\mathbf{y} = \mathbf{0} \tag{3.63}$$

since the projection matrix is idempotent. The corresponding cost function is also defined in terms of the projections by

$$\begin{aligned}\mathcal{J}_{LS} &= \tfrac{1}{2}\mathbf{e}'\mathbf{e} = \mathbf{y}'P_e'P_e\mathbf{y} = \mathbf{y}'P_e\mathbf{y} = \mathbf{y}'(I - P_c)\mathbf{y} \\ &= \mathbf{y}'\mathbf{y} - (P_c\mathbf{y})'(P_c\mathbf{y}) = \mathbf{y}'\mathbf{y} - \hat{\mathbf{s}}'\hat{\mathbf{s}}.\end{aligned} \tag{3.64}$$

In summary, we have the batch "geometric" description of the *LS* estimator as follows:

| | |
|---|---|
| Criterion: | $J_{LS} = \frac{1}{2}\mathbf{e}'\mathbf{e} = \frac{1}{2}\mathbf{y}'P_e\mathbf{y}$ |
| Models: | $\mathbf{s} = P_c\mathbf{y} = C\Theta$ |
| Algorithm: | $\hat{\Theta}_{LS} = (C'C)^{-1}C'\mathbf{y};\ \hat{\mathbf{s}} = P_c\mathbf{y} = C\hat{\Theta}_{LS}$ |
| Quality: | $\mathcal{J}_{LS} = \mathbf{y}'\mathbf{y} - \hat{\mathbf{s}}'\hat{\mathbf{s}}$ |

It is interesting to see that the least-squares estimation problem, when viewed from the geometric perspective, leads to the idea of orthogonal projections on subspaces defined by the signal and associated error vectors. Next let us investigate the same ideas cast into a computational problem that can be attacked using the singular value decomposition [16] of linear algebra.

The *singular value decomposition* (*SVD*) of a (rectangular) matrix, say $Y \in \mathcal{R}^{N_y \times M_y}$ is defined by

$$Y = U \Sigma V' \tag{3.65}$$

where $U \in \mathcal{R}^{N_y \times N_y}$, $V \in \mathcal{R}^{M_y \times M_y}$, $U$, $V$ are *unitary* matrices ($U' = U^{-1}$), and $\Sigma \in \mathcal{R}^{N_y \times M_y}$ is a matrix with elements $\sigma_{ii}$ arranged in descending order of magnitude such that $\sigma_{11} \geq \sigma_{22}, \ldots, \geq \sigma_{\min(N_y, M_y)}$. The matrices $U$ and $V$ are called the left and right *singular vectors* of $Y$ and are the respective eigenvectors of $YY'$ and $Y'Y$. The singular values $\{\sigma_{ii}\}$ are the square roots of the eigenvalues of $YY'$ or $Y'Y$.

In general, the *SVD* of $Y$ can be written as

$$Y = [U_1 \mid U_2] \left[\begin{array}{c|c|c} \Sigma_1 & | & 0 \\ -- & -- & -- \\ 0 & | & \Sigma_2 \end{array}\right] \left[\begin{array}{c} V_1' \\ -- \\ V_1' \end{array}\right] = U_1 \Sigma_1 V_1' + U_2 \Sigma_2 V_2' \tag{3.66}$$

with $Y \in \mathcal{R}^{N_y \times M_y}$, $U_1 \in \mathcal{R}^{N_y \times N_1}$, $U_2 \in \mathcal{R}^{N_y \times N_2}$, $V_1 \in \mathcal{R}^{N_1 \times M_y}$, $V_2 \in \mathcal{R}^{N_2 \times M_y}$, $\Sigma_1 \in \mathcal{R}^{N_1 \times N_1}$ and $\Sigma_2 \in \mathcal{R}^{N_2 \times N_2}$. Thus the *SVD* explicitly demonstrates the orthogonal decomposition of $Y$ into its constituent subspaces.

Next let us return to the idea of the orthogonal decomposition of the data space by defining two sets of independent vectors: one set spanning the *signal subspace*, $\mathcal{S}$, such that $\mathbf{c_i} \in \mathcal{R}^{N_y \times 1}; i =, \ldots, N_\theta$, and the other set spanning the orthogonal *error subspace*, $\mathcal{E}$, such that $\boldsymbol{\epsilon_i} \in \mathcal{R}^{N_y \times 1}; i =, \ldots, N_y - N_\theta$. Collecting the column vectors, we construct two full-rank matrices,

$$C = \left[\mathbf{c}_1 \;\; \mathbf{c}_2 \cdots \mathbf{c}_{N_\theta}\right] \in \mathcal{R}^{N_y \times N_\theta} \quad \text{and}$$
$$\mathcal{E} = \left[\epsilon_1 \;\; \epsilon_2 \cdots \epsilon_{N_y - N_\theta}\right] \in \mathcal{R}^{N_y \times (N_y - N_\theta)}$$

such that the orthogonality condition holds

$$\mathcal{E}' C = \mathbf{0}$$

establishing the orthogonal decomposition of the $N_y$-dimensional data space (see Figure 3.2*b*).

If $\rho(Y) = n \leq \min(N_y, M_y)$, then it can be shown ([16], [17], [18]) that

$$\begin{cases} \sigma_{kk} > 0, & k = 1, \ldots, n \\ \sigma_{kk} = 0, & k = n+1, \ldots, \min(N_y, M_y) \end{cases} \tag{3.67}$$

Therefore the rank $n$ matrix $Y$ can be written as

$$Y = [U_1 \mid U_2] \left[\begin{array}{c|c|c} \Sigma_1 & | & 0 \\ -- & -- & -- \\ 0 & | & 0 \end{array}\right] \left[\begin{array}{c} V_1' \\ -- \\ V_2' \end{array}\right] = U_1 \Sigma_1 V_1' \tag{3.68}$$

where $\Sigma_1 \in \mathcal{R}^{n\times n}$ is nonsingular, $U_1$, $V_1$ are an orthonormal basis for $\mathcal{S}$, and $U_2$ and $V_2$ are an orthonormal basis for the $\mathcal{E}$.

It is also important to realize that once the *SVD* of a matrix has been performed, its *inverse* (in the square matrix case) is trivial. Let $X = U\Sigma V'$ be the *SVD* of $X \in \mathcal{R}^{N_x\times N_x}$. Then

$$X^{-1} = (V^{-1})'\Sigma^{-1}U^{-1} = (V')'\Sigma^{-1}U' = V\Sigma^{-1}U' \tag{3.69}$$

Thus, since $\Sigma$ is diagonal, we merely take reciprocals and multiply to construct the inverse. Next we use the *SVD* to revisit the batch least-squares problem.

With this information in mind, let us now construct the data matrix as

$$Y = [C \;\; | \;\; E] \in \mathcal{R}^{N_y\times N_y}$$

with $C$ and $E$ defined (above) as the signal and error subspaces. Recall that the batch least-squares estimator is obtained by solving the *normal equations* given by

$$\left(C'C\right)\hat{\Theta}_{LS} = C'\mathbf{y} \tag{3.70}$$

Alternatively, using the *SVD* of $C = U_C\Sigma_C V_C'$ and substituting, we have

$$\left(V_C\Sigma_C U_C'\right)\left(U_C\Sigma_C V_C'\right)\hat{\Theta}_{LS} = \left(V_C\Sigma_C\Sigma_C V_C'\right)\hat{\Theta}_{LS} = \left(V_C\Sigma_C U_C'\right)\mathbf{y} \tag{3.71}$$

or inverting yields

$$\hat{\Theta}_{LS} = \left(V_C\Sigma_C^{-1}\Sigma_C^{-1}V_C'\right)\left(V_C\Sigma_C U_C'\right)\mathbf{y} = V_C\Sigma_C^{-1}U_C'\mathbf{y} = \sum_{i=1}^{N_\theta}\left(\frac{\mathbf{v}_i\mathbf{u}_i'}{\sigma_i}\right)\mathbf{y} \tag{3.72}$$

Thus the signal estimate is

$$\hat{\mathbf{s}} = C\hat{\Theta}_{LS} = \left(U_C\Sigma_C V_C'\right)\left(V_C\Sigma_C^{-1}U_C'\right)\mathbf{y} = U_C U_C'\mathbf{y} = \sum_{i=1}^{N_\theta}\left(\mathbf{u}_i\mathbf{u}_i'\right)\mathbf{y} \tag{3.73}$$

The corresponding projection matrices are given by

$$P_c = C\left(C'C\right)^{-1}C' = U_C U_C' \quad \text{and} \quad P_e = (I - P_c) = I - U_C U_C' \tag{3.74}$$

Thus, in summary, we have that the *SVD* description of the batch least-squares problem yields

Criterion: $J_{LS} = \frac{1}{2}\mathbf{e}'\mathbf{e} = \frac{1}{2}\mathbf{y}'P_e\mathbf{y} = \mathbf{y}'(I - U_C U_C')\mathbf{y}$

Models: $\mathbf{s} = U_C\Sigma_C V_C'\Theta$

Algorithm: $\hat{\Theta}_{LS} = V_C\Sigma_C^{-1}U_C'\mathbf{y}; \quad \hat{\mathbf{s}} = U_C U_C'\mathbf{y}$

Quality: $\mathcal{J}_{LS} = \mathbf{y}'\mathbf{y} - \hat{\mathbf{s}}'\hat{\mathbf{s}}$

This completes the geometric perspective of the least-squares problem. The geometric description offers some insight into the idea of orthogonal decompositions, projections, and computations using the *SVD*. In the final section we extend these ideas from parameter to random signal estimates using orthogonal projection theory.

### 3.3.3 Recursive Least Squares

In this section we investigate "recursive" or equivalently "sequential" solutions to the least-squares estimation problem. Recursive estimation techniques evolved quite naturally during the advent of the digital computer in the late 1950s, since both are sequential processes. It is important to realize the recursive least-squares solution is identical to the batch solution after it converges, so there is *no gain* in estimator performance properties (e.g., *CRB*); however, the number of computations is significantly less than the equivalent batch technique. Also it is important to realize that the recursive approach provides the underlying theoretical and pragmatic basis of all *adaptive estimation* techniques; thus they are important in their own right!

Many processors can be placed in a recursive form with various subtleties emerging in the calculation of the current estimate ($\hat{\Theta}_{\text{old}}$). The standard technique employed is based on correcting or updating the current estimate as a new measurement data sample becomes available. The estimates generally take the *recursive form*:

$$\hat{\Theta}_{\text{new}} = \hat{\Theta}_{\text{old}} + K E_{\text{new}} \tag{3.75}$$

where

$$E_{\text{new}} = Y - \hat{Y}_{\text{old}} = Y - C\hat{\Theta}_{\text{old}}$$

Here we see that the new estimate is obtained by correcting the old estimate with a $K$-weighted error. The error term $E_{\text{new}}$ is the new information or innovation; that is, it is the difference between the actual measurement and the predicted measurement ($\hat{Y}_{\text{old}}$) based on the old estimate ($\hat{\Theta}_{\text{old}}$). The computation of the weight matrix $K$ depends on the criterion used (mean-squared error, absolute error, etc.).

Consider the following example, which shows how to recursively estimate the sample mean.

***Example 3.4*** The sample mean estimator can easily be put in recursive form. The estimator is given by

$$\hat{X}(N) = \frac{1}{N}\sum_{t=1}^{N} y(t)$$

Extracting the $N$th term from the sum, we obtain

$$\hat{X}(N) = \frac{1}{N}y(N) + \frac{1}{N}\sum_{t=1}^{N-1} y(t)$$

Identify $\hat{X}(N-1)$ from the last term; that is,

$$\hat{X}(N) = \frac{1}{N}y(N) + \frac{N-1}{N}\hat{X}(N-1)$$

The recursive form is given by

$$\underbrace{\hat{X}(N)}_{\text{NEW}} = \underbrace{\hat{X}(N-1)}_{\text{OLD}} + \underbrace{\frac{1}{N}}_{\text{WT}} \underbrace{[y(N) - \hat{X}(N-1)]}_{\text{ERROR}}$$

This procedure to develop the "recursive form" is very important and can be applied to a multitude of processors. Note the steps in determining the form:

1. Remove the $N$th term from the summation.
2. Identify the previous estimate in terms of the $N-1$ remaining terms.
3. Perform the algebra to determine the gain factor, and place the estimator in the recursive form of Eq. (3.75) for a *scalar* measurement.[3]

Let us develop the general recursive form of the "batch" least-squares estimator of Eq. (3.53). The batch *LS* estimator can be calculated recursively for a scalar measurement $y(t)$ by defining

$$\mathbf{y}(t-1) := \begin{bmatrix} y(1) \\ \vdots \\ y(t-1) \end{bmatrix}, \quad \Theta(t-1) := \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_{N_\theta} \end{bmatrix}, \quad C(t-1) := \begin{bmatrix} \mathbf{c}'(1) \\ \vdots \\ \mathbf{c}'(t-1) \end{bmatrix}$$

$\Theta \in \mathcal{R}^{N_\Theta \times 1}, \mathbf{c}' \in \mathcal{R}^{1 \times N_\theta}, \mathbf{y} \in \mathcal{R}^{(t-1)\times 1}, C \in \mathcal{R}^{(t-1)\times N_\theta}$

The batch estimate of $\theta$ based on the set of past data up to $t-1$ is obtained ($W = 1$) from Eq. (3.53) as

$$\hat{\Theta}_{LS}(t-1) = P(t-1)C'(t-1)\mathbf{y}(t-1) \tag{3.76}$$

where $P(t-1) = [C'(t-1)C(t-1)]^{-1}$. Now suppose that we would like to add a new (scalar) measurement data sample, $y(t)$. Then we have

$$\Theta_{LS}(t) = P(t)C'(t)\mathbf{y}(t) \tag{3.77}$$

Now the matrix, $C'$ can be augmented in terms of the past and new data sample as

$$C'(t) = [\mathbf{c}(1)\cdots\mathbf{c}(t-1) \mid \mathbf{c}(t)\ ] = \left[C'(t-1) \mid \mathbf{c}(t)\ \right] \tag{3.78}$$

[3]We choose to derive the recursive *LS* estimator, assuming a scalar measurement because it is notationally simpler than the vector case. The vector case follows the "identical" steps with weighting matrix, $W \in \mathcal{R}^{N_y \times N_\theta}$, the identity matrix instead of unity in the scalar equations (see [2] for details).

with the corresponding augmented data vector

$$\mathbf{y}(t) = \left[\mathbf{y}(t-1) \mid y(t)\right] \tag{3.79}$$

Therefore substituting these relations into Eq. (3.77) and multiplying, we obtain

$$\begin{aligned}\Theta_{LS}(t) &= P(t)\left[\, C'(t-1) \mid \mathbf{c}(t)\,\right]\begin{bmatrix}\mathbf{y}(t-1)\\ --\\ y(t)\end{bmatrix}\\ &= P(t)\left[C'(t-1)\mathbf{y}(t-1) + \mathbf{c}(t)y(t)\right]\end{aligned} \tag{3.80}$$

Also $P(t)$ can be expressed in terms of $C'$ of Eq. (3.78) as

$$P^{-1}(t+1) = C'(t+1)C(t+1) = \left[C'(t-1) \mid \mathbf{c}(t)\,\right]\begin{bmatrix}C(t-1)\\ --\\ \mathbf{c}'(t)\end{bmatrix}$$

or

$$P(t) = \left[P^{-1}(t-1) + \mathbf{c}(t)\mathbf{c}'(t)\right]^{-1} \tag{3.81}$$

Applying the *matrix inversion lemma*[4] with $A = P^{-1}(t-1)$, $B = \mathbf{c}(t)$, $C = 1$, and $D = \mathbf{c}'(t)$, we obtain

$$P(t) = P(t-1) - P(t-1)\mathbf{c}(t)\left[\mathbf{c}'(t)P(t-1)\mathbf{c}(t) + 1\right]^{-1}\mathbf{c}'(t)P(t-1) \tag{3.82}$$

Define the *weight* or *gain* matrix as

$$K(t) := P(t-1)\mathbf{c}(t)\left[\mathbf{c}'(t)P(t-1)\mathbf{c}(t) + 1\right]^{-1}$$

Therefore

$$P(t) = \left[I - K(t)\mathbf{c}'(t)\right]P(t-1) \tag{3.83}$$

Substituting this result into Eq. (3.80) for $P(t)$, we obtain

$$\Theta_{LS}(t) = \left[I - K(t)\mathbf{c}'(t)\right]P(t-1)\left[C'(t-1)\mathbf{y}(t-1) + \mathbf{c}(t)y(t)\right] \tag{3.84}$$

Thus, using Eq. (3.76), the *LS* estimate of Eq. (3.77) can be written as

$$\Theta_{LS}(t) = \left[I - K(t)\mathbf{c}'(t)\right]\Theta_{LS}(t-1) + \left[I - K(t)\mathbf{c}'(t)\right]P(t-1)\mathbf{c}(t)y(t) \tag{3.85}$$

[4]The matrix inversion lemma is given by $[A + BCD]^{-1} = A^{-1} - A^{-1}B[DA^{-1}B + C^{-1}]^{-1}DA^{-1}$ [2].

Concentrating on the last term of this expression, we perform the indicated multiplication to obtain

$$\left[I - K(t)\mathbf{c}'(t)\right] P(t-1)\mathbf{c}(t)y(t) = \left[P(t-1)\mathbf{c}(t) - K(t)\mathbf{c}'(t)P(t-1)\mathbf{c}(t)\right] y(t)$$

Then, substituting for $K(t)$, we have

$$\left[P(t-1)\mathbf{c}(t) - P(t-1)\mathbf{c}(t)\left[\mathbf{c}'(t)P(t-1)\mathbf{c}(t)+1\right]^{-1}\mathbf{c}'(t)P(t-1)\mathbf{c}(t)\right] y(t)$$

Factoring $P(t)\mathbf{c}(t)$ on the left gives

$$P(t-1)\mathbf{c}(t)\left[I - [\mathbf{c}'(t)P(t-1)\mathbf{c}(t)+1]^{-1}\mathbf{c}'(t)P(t-1)\mathbf{c}(t)\right] y(t)$$

Factoring the inverse from the left of this expression now gives

$$P(t-1)\mathbf{c}(t)\left[\mathbf{c}'(t)P(t-1)\mathbf{c}(t)+1\right]^{-1}\left\{[\mathbf{c}'(t)P(t-1)\mathbf{c}(t)+1] - \mathbf{c}'(t)P(t-1)\mathbf{c}(t)\right\} y(t) \tag{3.86}$$

But this term is just $K(t)y(t)$; therefore, substituting this result into Eq. (3.85), we obtain

$$\Theta_{LS}(t) = \left[I - K(t)\mathbf{c}'(t)\right]\Theta_{LS}(t-1) + K(t)y(t) \tag{3.87}$$

which by multiplying and combining like terms gives the *recursive LS estimate*

$$\Theta_{LS}(t) = \Theta_{LS}(t-1) + K(t)\left[y(t) - \mathbf{c}'(t)\Theta_{LS}(t-1)\right] \tag{3.88}$$

We summarize the results in Table 3.1 for the scalar measurement case.

Consider the following example of formulating the trend removal problem using the recursive rather than batch approach.

***Example 3.5*** Starting with the trajectory representation of Example 3.3, we have that the underlying scalar measurement model is

$$y(t) = s(t) + \Theta(t) + n(t) \qquad \text{for } t = 1, \ldots, N$$

where $\Theta(t)$ is the unknown trajectory (trend) and $y$, $s$, and $n$ are the respective measurement, signal, and random noise. The problem is to estimate the trend, $\hat{\Theta}(t)$, and remove it from the data using a temporal polynomial as the model; that is,

$$\Theta(t) = \theta_0 + \theta_1 t + \cdots + \theta_{N_\theta - 1}t^{N_\theta - 1} = \mathbf{c}'(t)\theta$$

where $\mathbf{c}'(t) = [1\ t \cdots t^{N_\theta - 1}]$ and $\theta = [\theta_0\ \theta_1 \cdots \theta_{N_\theta - 1}]'$. The dynamic measurement model at time $t$ is therefore

$$y(t) = s(t) + \Theta(t) + n(t) = s(t) + \mathbf{c}'(t)\theta + n(t)$$

**Table 3.1. Recursive Least-Squares (Scalar) Algorithm**

Parameter correction

$$\underbrace{\hat{\Theta}(t)}_{\text{New}} = \underbrace{\hat{\Theta}(t-1)}_{\text{Old}} + \underbrace{K(t)}_{\text{Weight}} \underbrace{e(t)}_{\text{Error}}$$

Error

$$e(t) = y(t) - \mathbf{c}'(t)\hat{\Theta}(t-1)$$

Gain

$$K(t) = P(t-1)\mathbf{c}(t)\left[\mathbf{c}'(t)P(t-1)\mathbf{c}(t) + 1\right]^{-1}$$

Covariance correction

$$P(t) = \left[I - K(t)\mathbf{c}'(t)\right]P(t-1)$$

Initial conditions

$$\hat{\Theta}(0), \quad P(0)$$

$\Theta \in \mathcal{R}^{N_\theta \times 1}$, $y \in \mathcal{R}^{1\times 1}$, $\mathbf{c}' \in \mathcal{R}^{1\times N_\theta}$ and $K \in \mathcal{R}^{N_\theta \times 1}$, $P \in \mathcal{R}^{N_\theta \times N_\theta}$

Now in the recursive formulation, $\theta(t) \longrightarrow \theta$, and at each point in time the estimate of $\theta(t)$ is accomplished using the *RLS* algorithm of Table 3.1; that is, (simply)

$$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)e(t)$$

is used to estimate the polynomial coefficients that provide the desired trend estimate

$$\hat{\Theta}(t) = \mathbf{c}'(t)\hat{\theta}(t)$$

which is removed from the measurements to yield the "trend-free" data

$$z(t) = y(t) - \hat{\Theta}(t)$$

This completes the example.

## 3.4 OPTIMAL SIGNAL ESTIMATION

In this section we discuss the concept of optimal signal estimation; that is, we extend the orthogonality concepts of the previous section to that of estimating random signals that exist in a random (vector) space.

The criterion function of most interest in signal processing literature is that of the mean-squared error defined by

$$J(t) := E\{e^2(t)\} \tag{3.89}$$

where the error is derived directly from the measurement and estimate as

$$e(t) = y(t) - \hat{y}(t) \tag{3.90}$$

The optimal estimate is the estimate that produces the smallest error. Geometrically we can think of this error sequence, $\{e(t)\}$ as the result of projecting the current data onto a space of past data. Intuitively, the minimum error occurs when the projection is perpendicular or orthogonal to the space, that is,

$$e(t) \perp Y(t-1) \tag{3.91}$$

where $Y(t-1) = \{y(0), \ldots, y(t-1)\}$ is the past data space. We depict this relationship in Figure 3.4. We can think of the error or more commonly the *innovation* (new information [19]) as the orthogonal projection of the data vector $y(t)$ on the past data space $Y(t-1)$ spanned by

$$\hat{y}(t|t-1) = E\{y(t)|Y(t-1)\} \tag{3.92}$$

So we see that the optimal estimator can be thought of as the orthogonal projection of $y(t)$ onto $Y(t-1)$.

Another interpretation of this result is to think of $\hat{y}(t|t-1)$ as an estimate of that part of $y(t)$ correlated with past data $Y(t-1)$ and $e(t)$ as the uncorrelated or orthogonal part, that is,

$$\underbrace{y(t)}_{\text{Data}} = \underbrace{e(t)}_{\text{Uncorrelated}} + \underbrace{\hat{y}(t|t-1)}_{\text{Correlated}} \tag{3.93}$$

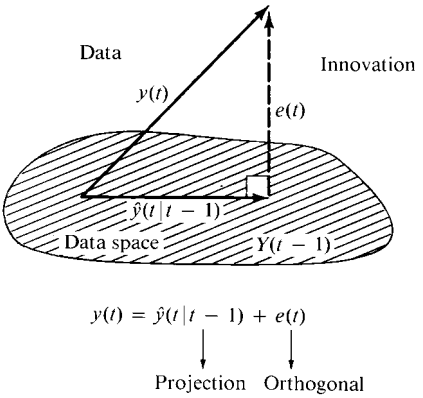



**Figure 3.4.** Orthogonal projection of the optimal estimate.

We know that the optimal estimator satisfies the *orthogonality condition* that is equivalent to Eq. (3.91):

$$E\{e'(t)y(T)\} = 0 \qquad \text{for } T < t \tag{3.94}$$

In fact we take the correlation canceling approach of [20]. To see this, consider the following example.

***Example 3.6*** Suppose that the data is composed of two parts: one correlated with the past and one not, that is,

$$y(t) = y_1(t) + y_2(t)$$

for

$$y_1(t) \in Y(t-1) \quad \text{and} \quad y_2(t) \perp Y(t-1)$$

We are asked to analyze the performance of an estimator. The optimal estimator is an estimate *only* of the correlated part

$$\hat{y}(t|t-1) = y_1(t)$$

Therefore the residual or innovation is

$$e(t) = y(t) - \hat{y}(t|t-1) = (y_1(t) + y_2(t)) - y_1(t) = y_2(t)$$

the uncorrelated part of the data. Thus we see that the results of canceling or removing the correlated part of the measurement with the optimal estimate is the uncorrelated part or innovation. This completes the example.

Example 3.6 emphasizes a very important point that is crucial to analyzing the performance of *any* optimal estimator. If the estimator is performing properly and we have estimated all of the information in the currently available data, then the *innovation sequence must be zero mean and white.* We will see when we design estimators, be they parametric, signal, spectral, or model-based, that checking the statistical properties of the corresponding residual or innovation sequence is the first step in analyzing proper performance.

The innovation sequence can be constructed from a set or batch of data, $Y(t)$ using the well-known Gram-Schmidt procedure (e.g., see [21]), in which we describe this data space as a random vector space with proper inner product given by

$$\langle a, b \rangle := E\{ab'\} \tag{3.95}$$

The construction problem becomes that of finding a nonsingular transformation $L$ such that

$$\mathbf{e} = L^{-1}\mathbf{Y} \qquad \text{for } \mathbf{e}, \mathbf{Y} \in \mathcal{R}^{N\times 1} \tag{3.96}$$

The *Gram-Schmidt* procedure recursively generates the innovation sequence as

$$\begin{aligned}
e(1) &= y(1) \\
e(2) &= y(2) - \langle y(2), e(1)\rangle \langle e(1), e(1)\rangle^{-1} e(1) \\
&= y(2) - R_{ye}(2,1) R_{ee}^{-1}(1) e(1) \\
&\vdots \quad \vdots \qquad\qquad\qquad \vdots \\
e(N) &= y(N) - \sum_{i=1}^{N-1} \langle y(N), e(i)\rangle \langle e(i), e(i)\rangle^{-1} e(i) \\
&= y(N) - \sum_{i=1}^{N-1} R_{ye}(N,i) R_{ee}^{-1}(i) e(i)
\end{aligned}$$

solving the sequence above for $y(t)$ implies that

$$y(t) = \sum_{i=1}^{t} L(t,i) e(i) \quad L(t,i) = R_{ye}(t,i) R_{ee}^{-1}(i) \qquad \text{with } L(t,t) = 1 \tag{3.97}$$

Therefore $L$ is lower triangular

$$L = \begin{bmatrix} 1 & & 0 \\ R_{ye}(2,1) R_{ee}^{-1}(1) & 1 & \\ \vdots & & \ddots \\ R_{ye}(N,1) R_{ee}^{-1}(1) & \cdots & R_{ye}(N,N-1) R_{ee}^{-1}(N,N-1) \quad 1 \end{bmatrix}$$

From Eq. (3.97) we see the orthogonal decomposition of the data in Eq. (3.93) becomes

$$\underbrace{y(t)}_{\text{Data}} = \underbrace{e(t)}_{\text{Orthogonal}} + \underbrace{\sum_{i=1}^{t-1} L(t,i) e(i)}_{\text{Projection}} \tag{3.98}$$

Therefore the optimal estimator is given by

$$\hat{y}(t|t-1) = \sum_{i=1}^{t-1} L(t,i) e(i) = \sum_{i=1}^{t-1} R_{ye}(t,i) R_{ee}^{-1}(i) e(i) \tag{3.99}$$

We note that the Gram-Schmidt procedure corresponds to transforming a correlated measurement vector to an uncorrelated innovation vector through a linear transformation [21] and therefore diagonalizing the corresponding covariance matrix; that is,

$$R_{yy} = E\{\mathbf{Y}\ \mathbf{Y}'\} = E\{L\mathbf{e}\ \mathbf{e}'L'\} = LR_{ee}L' \tag{3.100}$$

Equivalently[5]

$$R_{ee} = L^{-1}R_{yy}L^{-T} \tag{3.101}$$

where $L$ satisfies Eq. (3.97), and by construction,

$$R_{ee} = \text{diag}[R_{ee}(1)\cdots R_{ee}(N)] \tag{3.102}$$

Consider the following example emphasizing the geometric aspects of this transformation.

***Example 3.7*** For the case of $N = 3$, construct the transformation matrix $L$ using the Gram-Schmidt procedure. Using Eq. (3.97), we have immediately that

$$\begin{bmatrix} y(1) \\ y(2) \\ y(3) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ L(2,1) & 1 & 0 \\ L(3,1) & L(3,2) & 1 \end{bmatrix} \begin{bmatrix} e(1) \\ e(2) \\ e(3) \end{bmatrix}$$

and we depict the corresponding vectors in Figure 3.5. Note the projections in the figure given by Eq. (3.99). This completes the example.

We also note that this result corresponds directly to the optimal Wiener solution of the "batch" linear estimation problem discussed earlier in this chapter with the data vector replaced by

$$\mathbf{Y} = L\mathbf{e} \tag{3.103}$$

Using Eq. (3.101), we obtain the equivalent optimal estimate

$$\hat{\Theta}_{MV} = K_{MV}\mathbf{Y} = R_{\Theta y}R_{yy}^{-1}\mathbf{Y} = (R_{\Theta e}L')(L^{-T}R_{ee}L^{-1})(L\mathbf{e}) = R_{\Theta e}R_{ee}^{-1}\mathbf{e} \tag{3.104}$$

Therefore

$$K_e = R_{\Theta e}R_{ee}^{-1} \tag{3.105}$$

with the corresponding quality

$$R_{\tilde{\Theta}\tilde{\Theta}} = R_{\Theta\Theta} - R_{\Theta e}R_{ee}^{-1}R_{e\Theta} \tag{3.106}$$

[5]We use the notation $A^{-T} = [A^{-1}]'$ and diag[·] to mean diagonal matrix with entries [·].

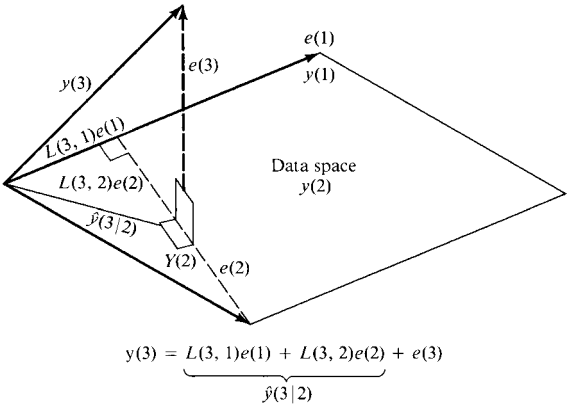



**Figure 3.5.** Innovations construction using Gram-Schmidt orthogonalization.

We can reconcile these results with the classical [22] by considering only the *scalar case*. Taking $Z$-transforms of Eq. (3.104), we obtain

$$K_{MV}(z) = \frac{S_{\Theta y}(z)}{S_{yy}(z)} \tag{3.107}$$

which is the frequency domain Wiener solution. If we further restrict the optimal estimator to be causal, then we must extract the causal parts of this solution. Since $S_{yy}(z)$ is a power spectrum, it can be factored (see [22] for details) as

$$S_{yy}(z) = S_{yy}(z^+)S_{yy}(z^-) \tag{3.108}$$

where $S_{yy}(z^+)$ contains only poles and zeros that lie inside the unit circle and $S_{yy}(z^-)$ only those lying outside. Thus the causal Wiener filter is given by

$$K(z) = [S_{\Theta y}(z)S_{yy}^{-1}(z^-)]_{\text{cp}}S_{yy}^{-1}(z^+) \tag{3.109}$$

where the *cp* notation signifies the "causal part" of $[\cdot]$. The same results can be obtained using the innovations or whitening filter approach, yielding

$$K_e(z) = [S_{\Theta e}(z)S_{ee}^{-1}(z^-)]_{\text{cp}}S_{ee}^{-1}(z^+) = [S_{\Theta e}(z)R_{ee}^{-1/2}]_{\text{cp}}R_{ee}^{-1/2} \tag{3.110}$$

since $S_{ee}(z) = R_{ee}$. Consider the following example of frequency domain Wiener filter design.

***Example 3.8*** Suppose that we are asked to design a Wiener filter to estimate a zero-mean, random parameter $\Theta$ with covariance $R_{\Theta\Theta}(k) = \alpha^{|k|}$ in random noise of unit variance, that is,

$$y(t) = \Theta(t) + v(t)$$

Then we have

$$S_{yy}(z) = S_{\Theta\Theta}(z) + S_{vv}(z) \quad \text{and} \quad S_{\Theta y}(z) = S_{\Theta\Theta}(z)$$

From the sum decomposition, we have

$$S_{yy}(z) = \frac{2 - \alpha(z + z^{-1})}{(1 - \alpha z^{-1})(1 - \alpha z)}$$

Thus the *noncausal* Wiener solution is simply

$$K(z) = \frac{S_{\Theta y}(z)}{S_{yy}(z)} = \frac{1 - \alpha^2}{2 - \alpha(z + z^{-1})}$$

while the causal solution is found by factoring $S_{yy}(z)$ as

$$S_{yy}(z) = S_{yy}(z^+)S_{yy}(z^-) = \frac{2 - \alpha(z + z^{-1})}{(1 - \alpha z^{-1})(1 - \alpha z)}$$
$$= \frac{(\beta_0 - \beta_1 z^{-1})(\beta_0 - \beta_1 z)}{(1 - \alpha z^{-1})(1 - \alpha z)}$$

which gives

$$\beta_0 = \frac{\alpha}{\beta_1} \quad \text{and} \quad \beta_1^2 = 1 - \sqrt{1 - \alpha^2}$$

Therefore

$$S_{yy}(z) = \left(\frac{\alpha}{\gamma}\right) \frac{(1 - \gamma z^{-1})(1 - \gamma z)}{(1 - \alpha z^{-1})(1 - \alpha z)}$$

where

$$\gamma = \frac{1 - \sqrt{1 - \alpha^2}}{\alpha}$$

The causal Wiener filter is then found by using the residue theorem [22] to find the causal part, that is,

$$K(z) = \left[\frac{\sqrt{\frac{\gamma}{\alpha}}(1 - \alpha^2)}{(1 - \gamma z^{-1})(1 - \gamma z)}\right]_{\text{cp}} \left[\frac{(1 - \alpha z^{-1})\sqrt{\frac{\gamma}{\alpha}})}{(1 - \gamma z^{-1})}\right]$$

Similar results can be obtained for the innovations approach by performing the spectral factorization first. This completes the discussion of optimal estimator and the related Wiener filter solutions.

## 3.5 SUMMARY

In this chapter we discussed the fundamental concepts underlying random signal analysis. Starting with the statistical notions of estimation, we developed desirable properties of an estimator and showed how to compare and evaluate its performance. Next we developed the minimum variance, maximum a posteriori, and maximum likelihood estimators and showed how they are related. Then we concentrated on least-squares estimation starting with the well-known batch solutions and progressing to the sequential or recursive approach. We also developed the basic geometric perspective of least squares using vector space and projection theory leading to the concepts of orthogonal decomposition and projections. These results led to the application of the singular value decomposition of a matrix. Finally these ideas were applied to random signals rather than the parameter estimation problem to develop the concept of optimal signal estimation.

## *MATLAB* NOTES

*MATLAB* has a wealth of linear algebraic commands to *solve* the various estimation problems discussed in this chapter. The **svd** command performs the singular value decomposition, providing the ordered set of singular values (diag $\Sigma$) and the unitary matrices of the decomposition ($U$, $V$). The **eig** command performs the eigendecomposition of a matrix as well, along with the efficient, numerically stable **qr** decomposition (Gram-Schmidt) and its variants (see [16], [18], [21] for details). Polynomial coefficient estimation is performed using the *SVD*-based, **polyfit** command, while the *RLS* algorithms require the *MATLAB System Identification* toolbox. The optimal Wiener solution is performed in the frequency domain using the transfer function estimation command **tfe**.

## REFERENCES

1. H. Van Trees, *Detection, Estimation and Modulation Theory*, New York: Wiley, 1968.
2. A. Sage and J. Melsa, *Estimation Theory with Applications to Communications and Control*, New York: McGraw-Hill, 1971.
3. M. Srinath, P. Rajasekaran, and R. Viswanathan. *Introduction to Statistical Signal Processing with Applications*, Englewood Cliffs, NJ: Prentice-Hall, 1996.
4. R. Hogg and A. Craig, *Introduction to Mathematical Statistics*, New York: Macmillan, 1970.
5. B. Porat, *Digital Processing of Random Signals: Theory and Methods*, Englewood Cliffs, NJ: Prentice-Hall, 1994.
6. J. Candy, "The Cramer-Rao bound: A measure of estimator quality," Lawrence Livermore National Laboratory Report, UCID-17660, 1977.
7. J. Candy, *Signal Processing: The Model-Based Approach*, New York: McGraw-Hill, 1986.

8. A. Jazwinski, *Stochastic Processes and Filtering Theory*, New York: Academic Press, 1970.
9. I. Rhodes, "A tutorial introduction to estimation and filtering," *IEEE Trans. Autom. Contr.*, **16**, 688–706 1971.
10. J. Candy, "Processing of RV signatures from radar tracking data," *RTO Set Symposium on High Resolution Radar Techniques*, RTO MP-40, pp. 38-1–38-12, 1999.
11. J. Mendel, *Lessons in Estimation Theory for Signal Processing, Communications, and Control*, Englewood Cliffs, NJ: Prentice-Hall, 1995.
12. S. Haykin, Ed., *Array Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1984.
13. D. Johnson and D. Dudgeon, *Array Signal Processing: Concepts and Techniques*, Englewood Cliffs, NJ: Prentice-Hall, 1993.
14. H. Van Trees, *Optimum Array Processing*, New York: Wiley-Interscience, 2002.
15. G. Strang, *Linear Algebra and Its Applications*, New York: Academic Press, 1976.
16. G. Golub and C. Van Loan, *Matrix Computations*, Baltimore: Johns Hopkins University Press, 1989.
17. L. Scharf, *Statistical Signal Processing: Detection, Estimation, and Time Series Analysis*, Reading, MA: Addison-Wesley, 1991.
18. C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*, Englewood Cliffs, NJ: Prentice-Hall, 1974.
19. T. Kailath, *Lectures on Kalman and Wiener Filtering Theory*, New York: Springer-Verlag, 1981.
20. S. Orfanidis, *Optimum Signal Processing*, New York: Macmillan, 1988.
21. G. Stewart, *Introduction to Matrix Computations*, New York: Academic Press, 1973.
22. S. Tretter, *Introduction to Discrete-Time Signal Processing*, New York: Wiley, 1976.

## PROBLEMS

**3.1** Suppose that we use the covariance estimator

$$\hat{R}_{xx}(k) = \frac{1}{N-k-1} \sum_{t=0}^{N-k-1} x(t)x(t+k) \qquad \text{for } k \geq 0$$

(**a**) Is this a biased estimate for $R_{xx}(k) = E\{x(t)x(t+k)\}$?

(**b**) What is the variance of this estimator?

**3.2** Derive the following properties of conditional expectations:

(**a**) $E_x\{X|Y\} = E\{X\}$ if $X$ and $Y$ are independent.

(**b**) $E\{X\} = E_y\{E\{X|Y\}\}$.

(**c**) $E_x\{g(Y)\ X\} = E_y\{g(Y)\ E\{X|Y\}\}$.

(**d**) $E_{xy}\{g(Y)\ X\} = E_y\{g(Y)\ E\{X|Y\}\}$.

(**e**) $E_x\{c|Y\} = c$.

(**f**) $E_x\{g(Y)|Y\} = g(Y)$.

(**g**) $E_{xy}\{cX + dY|Z\} = cE\{X|Z\} + dE\{Y|Z\}$.

**3.3** Verify the following properties:

**(a)** $\nabla_x(a'b) = (\nabla_x a')b + (\nabla_x b')a$ for $a, b \in R^n$ and functions of $x$.

**(b)** $\nabla_x(b'x) = b$.

**(c)** $\nabla_x(x'C) = C, C \in R^{n\times m}$.

**(d)** $\nabla_x(x') = I$.

**(e)** $\nabla_x(x'x) = 2x$.

**(f)** $\nabla_x(x'Ax) = Ax + A'x$ for $A$ not a function of $x$.

**3.4** Show that for any unbiased estimate of $\hat{\mathbf{x}}(y)$ of the parameter vector $\mathbf{x}$ the *Cramer-Rao bound* is

$$\text{cov}(\tilde{x}|x) \geq \mathcal{I}^{-1} \qquad \text{for } \tilde{x} = x = \hat{x}(y) \text{ and } \mathbf{x} \in \mathcal{R}^n, \mathcal{I} \in \mathcal{R}^{n\times n}$$

where the *information matrix* is defined by

$$\mathcal{I} := -E_y\{\nabla_x\,(\nabla_x\ \ln\ Pr(Y|X))'\}$$

**3.5** Let $x(t)$ be an unknown dynamic parameter obtained from the linear measurement system

$$y(t) = C(t)x(t) + v(t)$$

where $v \sim \mathcal{N}(0, R_{vv}(t))$ with $y, v \in \mathcal{R}^{p\times 1}$, and $x$ is governed by the state transition mechanism

$$x(t+1) = \Phi(t+1, t)x(t) \qquad \text{for } \Phi \in \mathcal{R}^{n\times n}$$

Calculate the Cramer-Rao bound for the case of estimating the initial state $x(0)$.

**3.6** Suppose that we have the following signal in additive gaussian noise:

$$y = x + n \qquad \text{with } n \sim \mathcal{N}(0, R_{nn})$$

**(a)** Find the Cramer-Rao bound if $x$ is assumed unknown.

**(b)** Find the Cramer-Rao bound if $p(x) = xe^{-x^2/R_{nn}},\ x \geq 0$.

**3.7** Suppose that we have two jointly distributed random vectors $x$ and $y$ with known means and variances. Find the linear minimum variance estimator. That is, find

$$\hat{x}_{MV} = \hat{A}y + \hat{b}$$

and the corresponding error covariance cov $\tilde{x}$

**3.8** We would like to estimate a vector of unknown parameters $p \epsilon \mathcal{R}^{n\times 1}$ from a sequence of $N$-noisy measurements given by

$$y = Cp + n$$

where $C \epsilon \mathcal{R}^{p\times n}$, $y, n \epsilon \mathcal{R}^{p\times 1}$, and $v$ is zero-mean and white covariance $R_{nn}$.

**(a)** Find the minimum variance estimate $\hat{p}_{MV}$.

**(b)** Find the corresponding quality $\text{cov}(p - \hat{p}_{MV})$.

**3.9** Suppose that we have $N$ samples $\{x(0) \cdots x(N-1)\}$ of a process $x(t)$ to be estimated by $N$ complex sinusoids of arbitrary frequencies $\{f_0, \ldots, f_{n-1}\}$. Then

$$x(k\triangle T) = \sum_{m=0}^{N-1} a_m \exp(j2\pi f_m k \triangle T) \qquad \text{for } k = 0, \ldots, N-1$$

Find the least-squares estimate $\hat{a}_{LS}$ of $\{a_m\}$.

**3.10** Suppose that we are given a measurement modeled by

$$y(t) = s + n(t)$$

where $s$ is random and zero-mean with variance $\sigma_s^2 = 4$ and $n$ is zero-mean and white with a unit variance. Find the two-weight minimum variance estimate of $s$, that is,

$$\hat{s}_{MV} = \sum_{i=1}^{2} w_i y(i)$$

that minimizes

$$J = E\{(s - \hat{s})^2\}$$

**3.11** Find the maximum likelihood and maximum a posteriori estimate of the parameter $x$,

$$p(x) = \alpha e^{-\alpha x}, \qquad x \geq 0, \alpha \geq 0$$

and

$$p(y|x) = xe^{-xy}, \qquad x \geq 0, y \geq 0$$

**3.12** Suppose that we are given a measurement system

$$y(t) = x(t) + v(t), \qquad t = 1, \ldots, N$$

where $v(t) \sim \mathcal{N}(0, 1)$.

**(a)** Find the maximum likelihood estimate of $x(t)$, that is, $\hat{x}_{\text{ML}}(t)$ for $t = 1, \ldots, N$.

**(b)** Find the maximum a posteriori estimate of $x(t)$, that is, $\hat{x}_{\text{MAP}}(t)$ if $p(x) = e^{-x}$.

**3.13** Suppose that we have a simple AM receiver with signal

$$y(t) = \Theta s(t) + v(t), \qquad t = 1, \ldots, N$$

where $\Theta$ is a random amplitude, $s$ is the known carrier, and $v \sim \mathcal{N}(0, R_{vv})$.

(**a**) Find the maximum likelihood estimate $\hat{\Theta}_{\text{ML}}$.

(**b**) Assume $\Theta \sim \mathcal{N}(\Theta_0, R_{\Theta\Theta})$. Find the maximum a posteriori estimate $\hat{\Theta}_{\text{MAP}}$.

(**c**) Assume that $\Theta$ is Rayleigh-distributed (a common assumption). Find $\hat{\Theta}_{\text{MAP}}$.

**3.14** We would like to estimate a signal from a noisy measurement

$$y = s + v$$

where $v \sim \mathcal{N}(0, 3)$ and $s$ is Rayleigh-distributed

$$p(s) = se^{-s^2/2}$$

with

$$p(y|s) = \frac{1}{\sqrt{6\pi}} e^{-(y-s)^2/6}$$

(**a**) Find the maximum likelihood estimate.

(**b**) Find the maximum a posteriori estimate.

(**c**) Calculate the Cramer-Rao bound (ignoring $p(s)$).

**3.15** Assume that a planet travels an elliptical orbit centered about a known point. Suppose that we make $M$ observations.

(**a**) Find the best estimate of the orbit and the corresponding quality, that is, for the elliptical orbit (gaussian problem)

$$\beta_1 x^2 + \beta_2 y^2 = 1$$

Find $\hat{\beta} = [\hat{\beta}_1 \quad \hat{\beta}_2]'$.

(**b**) Suppose that we are given the following measurements: $(x, y) = \{(2, 2), (0, 2), (-1, 1),$ and $(-1, 2)\}$. Find $\hat{\beta}$ and $\hat{J}$, the cost function.

**3.16** Find the parameters, $\beta_1$, $\beta_2$, and $\beta_3$ such that

$$f(t) = \beta_1 t + \beta_2 t^2 + \beta_3 \sin t$$

fits $(t, f(t)) := \{(0, 1), (\pi/4, 2), (\pi/2, 3),$ and $(\pi, 4)\}$ with corresponding quality estimate, $\hat{J}$.

**3.17** Calculate the batch and sequential least-squares estimate of the parameter vector $x$ based on two measurements $y(1)$ and $y(2)$, where

$$y(1) = C(1)x + v(1) = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$y(2) = c'x + v(2) = 4$$

$$C = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad c'(1) = [1\ 2], \quad W = I$$

**3.18** Suppose that we have a two-measurement system given by

$$y = \begin{bmatrix} 3 \\ 4 \end{bmatrix} + v$$

where $R_{vv} = \text{diag}[1,\ 0.1]$.

(**a**) What is the batch least-squares estimate ($W = I$) of the parameter $x$ if $y = [7\ 21]'$?

(**b**) What is the batch weighted least-squares estimate of the parameter $x$ with $W$ selected for minimum variance estimation?

**3.19** Given a sequence of random variables $\{y(t)\}$, define

$$e(t) = y(t) - \hat{y}(t|t-1), \quad \hat{y}(1|0) = 0$$

$$\hat{y}(t|t-1) = E\{y(t)|Y(t-1)\}$$

is the minimum variance estimate. Then

(**a**) Show that the $\{e(t)\}$ are orthogonal.

(**b**) Show that if $E\{e^2(t)\} > 0$, then

$$\underline{Y}(N) = \begin{bmatrix} y(1) \\ \vdots \\ y(N) \end{bmatrix} \quad \text{and} \quad \underline{e}(N) = \begin{bmatrix} e(1) \\ \vdots \\ e(N) \end{bmatrix}$$

are related by

$$\underline{Y}(N) = L(\underline{N})\underline{e}(N)$$

(**c**) If $W(\underline{N})$ is a linear operation that yields $\hat{\underline{Y}}(N)$ from $\underline{Y}(N)$, show that

$$W(\underline{N}) = I - L^{-1}(\underline{N}) \quad \text{and} \quad \hat{\underline{Y}}(N) = \underline{Y}(N) - \underline{e}(N)$$

(**d**) Show that $L(\underline{N}) = R_{ye}(\underline{N})R_{ee}^{-1}(\underline{N})$

(**e**) Show that $R_{yy}(\underline{N}) = L(\underline{N})R_{ee}(\underline{N})L'(\underline{N})$

**(f)** If $x$ is a related vector, show that

$$\hat{x} = R_{xy}(\underline{N})R_{yy}^{-1}(\underline{N})\underline{Y}(N) = R_{xe}(\underline{N})R_{ee}^{-1}(\underline{N})\underline{e}(N)$$

**(g)** If $R_{yy}(i, j) = \alpha^{|i-j|}$ for $0 < \alpha < 1$, find $L(\underline{N})$ and $R_{yy}^{-1}(\underline{N})$.

**3.20** Construct an innovation sequence $\{e(1), e(2)\}$ from the measurement sequence

$$y(1) = \begin{bmatrix} 1 \\ 4 \end{bmatrix}, y(2) = \begin{bmatrix} 2 \\ 3 \end{bmatrix} \text{ if } R_{ee} = \text{diag}\{1, 2\}, \text{ and } R_{ye}(2, 1) = \begin{bmatrix} 2 & 1 \\ 1 & 6 \end{bmatrix}$$

**3.21** Construct the Gram-Schmidt orthogonalization procedure starting with $e(N) = y(N)$ rather than $e(1) = y(1)$ as in the chapter.

**(a)** Determine the transformation matrix $L^*$, how does it differ from $L$ previously?

**(b)** Construct the covariance equivalence transformation as in Eq. (3.97). How does it differ?

# 4

# *AR, MA, ARMAX,* LATTICE, EXPONENTIAL, WAVE MODEL-BASED PROCESSORS

## 4.1 INTRODUCTION

Parametric methods of signal processing have evolved as the modern approach to processing both deterministic and random signals. Probably the catalyst that has initiated this now popular approach evolved from the application of signal processing techniques to the analysis and synthesis of speech ([1], [2]). In fact these techniques are utilized heavily in the modern parametric approach to spectral estimation [3]. It is only fair to mention that parametric methods have existed in other disciplines for quite some time. Initially, they evolved from time series analysis [4] and controls/estimation, notably after the development of the Kalman filter [5], [6]. More recently the work of Ljung [7], Goodwin [8], and others have provided an analytic and practical framework for designing and evaluating the performance of parametric processors.

Parametric methods are more appropriately called model-based (see Candy [9]) because each technique first assumes a prespecified model-set (all-pole, all-zero, etc.) and then estimates the model parameters. In fact the model-based approach consists of three essential ingredients: (1) data, (2) model-set, and (3) criterion. Once we select the model-set, we "fit" the model (parameter estimation) to the data according to some criterion. Parametric methods correspond to a model-based signal processing scheme in which the model parameters are unknown.

The model-based parametric approach to signal processing is best summarized in Figure 4.1. Here we see that once the model-set is selected, the estimator is used

*Model-Based Signal Processing*, by James V. Candy

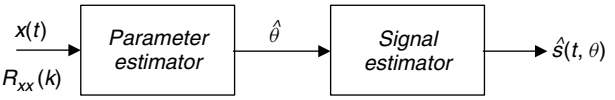



**Figure 4.1.** Model-based (parametric) signal processing method.

to obtain the unknown parameters that specify the model. The signal estimate is then constructed using these parameters.

In summary, the *model-based (parametric) signal processing* method consists of the following steps:

1. Select a representative model-set (e.g., *ARMAX*, lattice, state space)
2. Estimate the model parameters from the data. That is, given $\{y(t)\}$ and/or $\{u(t)\}$, find
$$\hat{\Theta} = \{\hat{A}(q^{-1}),\ \hat{B}(q^{-1}),\ \hat{C}(q^{-1}),\ R_{ee}\}$$
3. Construct the signal estimate using these parameters, that is,
$$\hat{s}(t, \Theta) = f(\hat{\Theta}, y(t))$$

## 4.2 *AR* (ALL-POLE) *MBP*

In this section we develop the model-based, all-pole processor using the autoregressive (*AR*) model of Chapter 2. In fact, the processor can be implemented using a recursive procedure called the Levinson-Durbin (*LD*) recursion to invert a Toeplitz matrix and obtain the set of parameters ($\{a_i\},\ R_{ee}$). These *AR* model-based processors have classically been called "linear predictors" or "prediction filters" [10]. In any case, we develop these processors as a means to produce a signal estimator and use the *LD* recursion to estimate the required model-based parameters.

Suppose that we would like to obtain an estimate of a signal, $s(t)$, based on past data, that is,
$$\hat{s}(t) = \hat{s}(t|Y_{past})$$

Taking the model-based parametric approach, we model the signal by the linear representation
$$s(t) = -\sum_{i=1}^{N_a} a_i y(t-i) \tag{4.1}$$

Therefore the measurement
$$y(t) = s(t) + e(t) = -\sum_{i=1}^{N_a} a_i y(t-i) + e(t) \tag{4.2}$$

is characterized by an *AR* model

$$A(q^{-1})y(t) = e(t) \tag{4.3}$$

for $e(t)$ zero-mean, white with variance $R_{ee}$ and $a_0 = 1$.

The basic model-based (parameter) estimation problem for the *AR* model, in the general case, is given as the minimum (error) variance solution to

$$\min_{\mathbf{a}} \; \mathcal{J}(t) = E\{e^2(t)\} \tag{4.4}$$

where **a** is the vector of unknown $a$-parameters with the corresponding *prediction error* (*innovation*) defined by

$$e(t) := y(t) - \hat{s}(t) \tag{4.5}$$

Here $\hat{s}(t)$ is the minimum variance estimate obtained from the *AR* model of Eq. (4.2) as

$$\hat{s}(t) = -\sum_{i=1}^{N_a} \hat{a}_i y(t-i) \tag{4.6}$$

Note that $\hat{s}$ is actually the one-step predicted estimate $\hat{s}(t|t-1)$ based on $[t-1]$ past data samples, hence the popular name "linear (combination of data) predictor." The estimator can be derived by minimizing the prediction error, $e(t)$ of Eq. (4.5), which can be expressed as (using Eq. 4.2)

$$e(t) = \sum_{i=0}^{N_a} a_i y(t-i) \text{ with } a_0 = 1 \tag{4.7}$$

Minimizing Eq. (4.7) with respect to the *AR* coefficients, $\{a_i\}$ for $i = 1, \ldots, N_a$, we obtain

$$\frac{\partial \mathcal{J}(t)}{\partial a_j} = \frac{\partial}{\partial a_j} E\{e^2(t)\} = 2E\left\{e(t)\frac{\partial e(t)}{\partial a_j}\right\} = 0 \tag{4.8}$$

The prediction error gradient is given by

$$\frac{\partial e(t)}{\partial a_j} = \frac{\partial}{\partial a_j}\left(\sum_{i=0}^{N_a} a_i y(t-i)\right) = y(t-j), \qquad j = 1, \ldots, N_a \tag{4.9}$$

Therefore, substituting into Eq. (4.8), we obtain the *orthogonality condition*

$$E\{e(t)y(t-j)\} = 0 \qquad \text{for } j = 1, \ldots, N_a \tag{4.10}$$

Now substituting for $e(t)$ of Eq. (4.7) gives

$$E\left\{\sum_{i=0}^{N_a} a_i\, y(t-i)y(t-j)\right\} = \sum_{i=0}^{N_a} a_i\, E\{y(t-i)y(t-j)\} \qquad \text{for } j = 1, \ldots, N_a$$

which yields to the so-called *normal or Yule-Walker* equations [4] as

$$\sum_{i=0}^{N_a} a_i R_{yy}(i-j) = 0 \qquad \text{for } j = 1, \ldots, N_a \tag{4.11}$$

Setting $a_0 = 1$ and using the fact that $R(-j) = R(j)$, we obtain the relation

$$\sum_{i=1}^{N_a} a_i R_{yy}(i-j) = -R_{yy}(j) \qquad \text{for} \quad j = 1, \ldots, N_a \tag{4.12}$$

Expanding these equations over $j$, we obtain the vector-matrix equations

$$\begin{bmatrix} R_{yy}(0) & R_{yy}(1) & \cdots & R_{yy}(N_a-1) \\ \vdots & \ddots & \ddots & \vdots \\ R_{yy}(N_a-2) & \cdots & \cdots & R_{yy}(1) \\ R_{yy}(N_a-1) & R_{yy}(N_a-2) & \cdots & R_{yy}(0) \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_{N_a-1} \\ a_{N_a} \end{bmatrix} = - \begin{bmatrix} R_{yy}(1) \\ \vdots \\ R_{yy}(N_a-1) \\ R_{yy}(N_a) \end{bmatrix} \tag{4.13}$$

which can be expressed compactly by

$$\mathbf{R}_{yy}\mathbf{a}(N_a) = -\mathbf{r}_{yy}(N_a) \tag{4.14}$$

for $\mathbf{a}(N_a), \mathbf{r}_{yy}(N_a) \in \mathcal{R}^{N_a \times 1}$, and $\mathbf{R}_{yy} \in \mathcal{R}^{N_a \times N_a}$, a Toeplitz matrix. Toeplitz matrices have some interesting properties, which we briefly discuss in a subsequent subsection.

Clearly, the solution to this estimation problem can now be obtained by inverting the Toeplitz matrix, that is,

$$\hat{\mathbf{a}}(N_a) = -\mathbf{R}_{yy}^{-1}\mathbf{r}_{yy}(N_a) \tag{4.15}$$

The quality is obtained by substituting Eq. (4.7) into Eq. (4.4):

$$\begin{aligned} R_{ee}(N_a) = E\{e^2(t)\} &= E\left\{\left(\sum_{i=0}^{N_a} a_i y(t-i)\right)\left(\sum_{j=0}^{N_a} a_j y(t-j)\right)\right\} \\ &= \sum_{i=0}^{N_a} a_i \sum_{j=0}^{N_a} a_j E\{y(t-i)y(t-j)\} \\ &= \sum_{i=0}^{N_a} a_i \sum_{j=0}^{N_a} a_j R_{yy}(i-j) \end{aligned}$$

Removing the term ($a_o = 1$) in the first sum and multiplying gives

$$R_{ee}(N_a) = \sum_{j=0}^{N_a} a_j R_{yy}(j) + \sum_{i=1}^{N_a} a_i \sum_{j=0}^{N_a} a_j R(i-j) \tag{4.16}$$

Now from the normal equations of Eq. (4.11) the second term is zero. Therefore we have

$$R_{ee}(N_a) = R_{yy}(0) + \sum_{j=1}^{N_a} a_j R_{yy}(j) \tag{4.17}$$

which gives the prediction error variance.

We summarize the batch all-pole approach as follows:

Criterion: $J = E\{e^2(t)\}$
Models:
  Measurement: $y(t) = s(t) + e(t)$
  Signal: $s(t) = -\sum_{i=1}^{N_a} a_i y(t-i)$
  Noise: $R_{ee}$
Algorithm: $\hat{\mathbf{a}} = \mathbf{R}_{yy}^{-1}\mathbf{r}_{yy}$
Quality: $R_{ee}(N_a) = R_{yy}(0) + \sum_{j=1}^{N_a} a_j R_{yy}(j)$

### 4.2.1 Levinson-Durbin Recursion

In this subsection we derive the Levinson-Durbin recursion that is the basis for some of the subsequent algorithms to follow. As we noted previously, this recursion also provides a method of transforming lattice parameters $\{k_i\}$ to *AR* parameters. From the computational viewpoint, the Levinson-Durbin recursion allow us to invert the underlying Toeplitz matrix of Eq. (4.14) in $N^2$ operations instead of the usual $N^3$.

The variance for the innovation or prediction error can be written

$$\sum_{i=0}^{N_a} a_i R_{yy}(i) = R_{ee}(N_a), \qquad a_0 = 1 \tag{4.18}$$

Combining Eqs. (4.11) and (4.18), we have

$$\sum_{i=0}^{N_a} a_i R_{yy}(i-j) = \begin{cases} R_{ee}(N_a), & j = 0 \\ 0, & j = 1, \ldots, N_a, \text{ for } \quad a_0 = 1 \end{cases}$$

If we expand these equations, we obtain an *augmented* Toeplitz equation

$$\begin{bmatrix} R_{yy}(0) & \cdots & R_{yy}(N_a) \\ \vdots & \ddots & \vdots \\ R_{yy}(N_a) & \cdots & R_{yy}(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_{N_a} \end{bmatrix} = \begin{bmatrix} R_{ee}(N_a) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{4.19}$$

or

$$\mathbf{R}_{yy}(N_a)\mathbf{a}(N_a) = \begin{bmatrix} R_{ee}(N_a) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{4.20}$$

Suppose that we would like the solution for the $(N_a + 1)$th-order estimator. Then we have

$$\mathbf{R}_{yy}(N_a+1)\mathbf{a}(N_a+1) = \begin{bmatrix} R_{ee}(N_a+1) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{4.21}$$

Now, if we assume that the optimal estimator is in fact $N_a$th-order, then

$$\mathbf{a}(N_a+1) = \begin{bmatrix} \mathbf{a}(N_a) \\ -- \\ 0 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} R_{ee}(N_a+1) \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} R_{ee}(N_a) \\ 0 \\ \vdots \\ \triangle_{N_a} \end{bmatrix} \tag{4.22}$$

Alternatively, we have from Eqs. (4.19) and (4.20) that

$$\begin{bmatrix} \mathbf{R}_{yy}(N_a) & | & R_{yy}(N_a+1) \\ ---- & | & \vdots \\ R_{yy}(N_a+1) & \cdots & R_{yy}(0) \end{bmatrix} \begin{bmatrix} \mathbf{a}(N_a) \\ -- \\ 0 \end{bmatrix} = \begin{bmatrix} R_{ee}(N_a) \\ \vdots \\ \triangle_{N_a} \end{bmatrix} \tag{4.23}$$

Therefore we must perform elementary operations on this equation to eliminate $\triangle_{N_a}$.

We perform these operations in two steps. First, starting with the right side of Eq. (4.23) to eliminate $\triangle_{N_a}$, we simply reorder the rows (elementary operation), multiply by the scalar, $\triangle_{N_a}/R_{ee}(N_a)$ and subtract, that is,

$$\begin{bmatrix} R_{ee}(N_a) \\ \vdots \\ \triangle_{N_a} \end{bmatrix} - \frac{\triangle_{N_a}}{R_{ee}(N_a)} \begin{bmatrix} \triangle_{N_a} \\ \vdots \\ R_{ee}(N_a) \end{bmatrix} = \begin{bmatrix} R_{ee}(N_a) - \dfrac{\triangle_{N_a}^2}{R_{ee}(N_a)} \\ \vdots \\ \triangle_{N_a} - \triangle_{N_a} \end{bmatrix}$$

$$= \begin{bmatrix} (1 - k_{N_a+1}^2) R_{ee}(N_a) \\ \vdots \\ 0 \end{bmatrix} \tag{4.24}$$

where we have defined the *reflection* or *partial correlation coefficient* as

$$k_{N_a+1} := \frac{\triangle_{N_a}}{R_{ee}(N_a)} \tag{4.25}$$

Now we perform the same operations on the left side of Eq. (4.23). That is, we interchange the rows, multiply by the reflection coefficient, and subtract:

$$\mathbf{R}_{yy}(N_a + 1) \left\{ \begin{bmatrix} \mathbf{a}(N_a) \\ -- \\ 0 \end{bmatrix} - k_{N_a+1} \begin{bmatrix} 0 \\ -- \\ \mathbf{a}_{\text{rev}}(N_a) \end{bmatrix} \right\} = \begin{bmatrix} (1 - k_{N_a+1}^2) R_{ee}(N_a) \\ \vdots \\ 0 \end{bmatrix} \tag{4.26}$$

where $\mathbf{a}_{\text{rev}}(N_a)$ is the $N_a$-coefficient vector in reverse order. Now expanding into the components of Eq. (4.26) and introducing the notation $a(i, j)$ as the $i$th coefficient of the $j$th-order predictor, we obtain

$$\mathbf{R}_{yy}(N_a + 1) \begin{bmatrix} 1 \\ a(1, N_a) - k_{N_a+1} a(N_a, N_a) \\ \vdots \\ a(i, N_a) - k_{N_a+1} a(N_a + 1 - i, N_a) \\ \vdots \\ -k_{N_a+1} \end{bmatrix} = \begin{bmatrix} (1 - k_{N_a+1}^2) R_{ee}(N_a) \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{4.27}$$

From Eq. (4.23), we see that $\triangle_{N_a}$ is given by

$$\triangle_{N_a} = \sum_{j=0}^{N_a} a(j, N_a) R_{yy}(N_a + 1 - j) \qquad \text{for } a(0, N_a) = 1 \tag{4.28}$$

**Table 4.1. Levinson-Durbin (*LD*) Recursion**

For $i = 1, \ldots, N_a$

$$\triangle_i = \sum_{j=0}^{i} a(j,i) R_{yy}(i+1-j)$$

$$k_{i+1} = \frac{\triangle_i}{R_{ee}(i)}$$

$$a(i+1, i+1) = -k_{i+1}$$

For $j = 1, \ldots, i$

$$a(j, i+1) = a(j,i) - k_{i+1} a(i+1-j, i)$$

$$R_{ee}(i+1) = (1 - k_{i+1}^2) R_{ee}(i)$$

Initialization

$$R_{ee}(0) = R_{yy}(0), \ \triangle_0 = R_{yy}(1), \ k_1 = \frac{R_{yy}(1)}{R_{yy}(0)}, \ a(0,i) = 1$$

Equating Eqs. (4.21) and (4.27), we obtain the *Levinson-Durbin* (*LD*) recursion, which we summarize in Table 4.1. Consider the following example demonstrating the recursion.

***Example 4.1*** Suppose that we have an exponentially correlated process, $R_{yy}(k) = \alpha^{|k|}$. We would like to develop a filtered estimate, $\hat{s}(t|t-1)$, using the (i) batch method and (ii) Levinson recursion.

- Case i: Batch solution

$$\mathbf{a} = -R_{yy}^{-1} \mathbf{r}_{yy}$$

$$R_{yy} = \begin{bmatrix} R_{yy}(0) & R_{yy}(1) \\ R_{yy}(1) & R_{yy}(0) \end{bmatrix} = \begin{bmatrix} 1 & \alpha \\ \alpha & 1 \end{bmatrix}$$

$$\mathbf{r}_{yy} = \begin{bmatrix} R_{yy}(1) \\ R_{yy}(2) \end{bmatrix} = \begin{bmatrix} \alpha \\ \alpha^2 \end{bmatrix}$$

Therefore

$$R_{yy}^{-1} = \frac{1}{1-\alpha^2} \begin{bmatrix} 1 & -\alpha \\ -\alpha & 1 \end{bmatrix}$$

and

$$\mathbf{a} = -\frac{1}{1-\alpha^2}\begin{bmatrix} 1 & -\alpha \\ -\alpha & 1 \end{bmatrix}\begin{bmatrix} \alpha \\ \alpha^2 \end{bmatrix} = -\frac{1}{1-\alpha^2}\begin{bmatrix} \alpha(1-\alpha^2) & 0 \end{bmatrix} = \begin{bmatrix} -\alpha \\ 0 \end{bmatrix}$$

$$R_{ee}(2) = \sum_{i=0}^{2} a(i,2)R_{yy}(i) = R_{yy}(0) + a(1,2)R_{yy}(1) + a(2,2)R_{yy}(2)$$
$$= 1 - \alpha^2$$

which gives $\hat{s}(t) = \hat{s}(t|t-1) = \alpha y(t-1)$ with $R_{ee} = 1 - \alpha^2$.

- Case ii: Levinson-Durbin recursion

$$R_{ee}(0) = R_{yy}(0) = 1, \quad \triangle_0 = R_{yy}(1) = \alpha, \quad k_1 = \frac{R_{yy}(1)}{R_{yy}(0)} = \alpha, \qquad (i = 0)$$

$$a(1,1) = -k_1 = -\alpha, \qquad (i = 1)$$

$$R_{ee}(1) = (1 - k_1^2)R_{ee}(0) = 1 - \alpha^2$$

$$\triangle_1 = \sum_{j=0}^{1} a(j,1)R_{yy}(2-j) = R_{yy}(2) + a(1,1)R_{yy}(1) = \alpha^2 - \alpha^2 = 0$$

$$k_2 = \frac{\triangle_1}{R_{ee}(1)} = 0$$

$$a(2,2) = 0$$

$$a(1,2) = a(1,1) = -\alpha$$

$$R_{ee}(2) = (1 - k_2^2)R_{ee}(1) = R_{ee}(1) = 1 - \alpha^2$$

which is identical to the batch solution.

We close this section with a practical design example.

***Example 4.2*** Suppose that we have a process that can be modeled by an *ARMAX*(2, 1, 1, 1) model given by the difference equation [6]

$$(1 - 1.5q^{-1} + 0.7q^{-2})y(t) = (1 + 0.5q^{-1})u(t) + \frac{1 + 0.3q^{-1}}{1 + 0.5q^{-1}}e(t)$$

where $u(t)$ is a delayed unit impulse at $t = 1.2$ second and $e \sim \mathcal{N}(0, 0.01)$. We choose this type of *ARMAX* model because, of the algorithms we will investigate, none can exactly reproduce the structure. We simulated this process using *MATLAB* [11] for a data set of $N = 128$ at $\triangle T = 0.1$ second and applied the *LD* algorithm of Table 4.1 to design an all-pole filter. We utilize an order estimation method (see Section 4.6 to follow) based on the Akaike information criterion (*AIC*) to automatically select the appropriate order. A third-order *AR* model was selected

($AIC = -278$) with the following estimated parameters:

| Parameter | True Value | Estimate |
|---|---|---|
| $a_1$ | −1.5 | −1.67 |
| $a_2$ | 0.7 | 1.11 |
| $a_3$ | — | −0.33 |
| $b_0$ | 1.0 | 0.15 |
| $b_1$ | 0.5 | — |
| $c_1$ | 0.3 | — |
| $d_1$ | 0.5 | — |

The estimated impulse response and signal are shown in Figure 4.2. The impulse response appears to have captured the major resonance in the data. The signal estimate obtained using the estimated parameters, that is,

$$\hat{s}(t) = (1 - \hat{A}(q^{-1}))y(t) = -\sum_{i=1}^{N_a} \hat{a}_i y(t-i)$$

produces a reasonable "filtered" signal.

This completes the design and the section on Levinson all-pole filters.

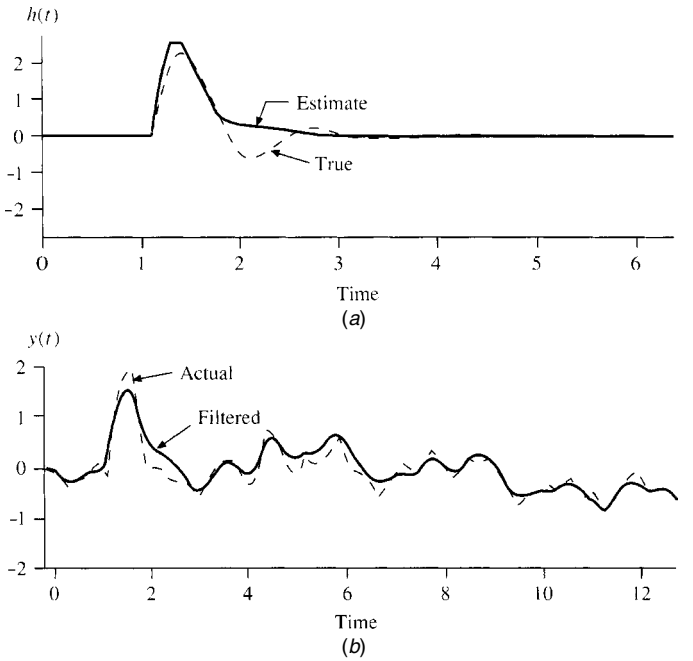



**Figure 4.2.** Levinson all-pole filter design for *ARMAX*(2, 1, 1, 1) simulation. (*a*) True and estimated impulse response. (*b*) Noisy and filtered measurement.

### 4.2.2 Toeplitz Matrices for *AR* Model-Based Processors

As we readily observe from the development of this algorithm, the *Toeplitz matrix* defined by its entries $R_{yy}(i, j) = R_{yy}(i - j)$ evolves in a crucial role of model-based (parametric) problems. In this subsection we investigate the basic properties of this matrix and show how it can be exploited to provide insight into the structure of the estimation problem.

First, we list (without proof) the following general *properties* of the Toeplitz (correlation) matrix, $R$ (see [10], [12] for details), of a *WSS* process:

1. $R \geq 0$ (positive semidefinite)
2. $R = R'$ (symmetric or Hermitian symmetric for $R \in C^{N \times N}$)
3. $\{\lambda_i\} \geq 0$ for $i = 1, \ldots, N$ (eigenvalues are real and positive)
4. $\mathbf{e}_i' \mathbf{e}_j = 0, \quad i \neq j$ (eigenvectors are orthogonal or orthonormal for Hermitian)
5. $R = E \Lambda E'$ (eigen-decomposition with $E$ unitary $(EE' = E'E = I)$)

Besides the eigen-decomposition property, the Toeplitz matrix can be decomposed using a *Cholesky decomposition* [10] of a positive definite matrix $(R > 0)$ defined by

$$R = LDU' = LDL' = \left(LD^{1/2}\right)\left(D^{T/2}L'\right) \tag{4.29}$$

where $L$ is a lower triangular matrix with unity on the diagonal, $D$ is a diagonal matrix with matrix *square roots* defined by $D := D^{1/2}D^{T/2}$ for $D^{T/2} := (D^{'1/2})$ and $U = L$ in the real symmetric case.

The structure of the Toeplitz matrix of Eq. (4.13) reveals much about the underlying model-based *AR* structure that is exploited to create efficient inversion algorithms. To put this in perspective, the derivation of the *Levinson-Durbin recursion* of the previous section followed a linear algebraic approach for ease of comprehension providing little insight into the "linear prediction" approach ([10], [12]) in which both forward and backward prediction error models evolve. The latter approach developed by Burg [10] will be used subsequently to develop lattice processors in which we explicitly define both forward and backward models. In the real case that we discussed, there is little difference in terms of the prediction error coefficients except that their order is reversed. But, in the complex case, $R$ is Hermitian, and the forward and backward coefficients are conjugates of one another as well as being reversed [12]. The backward predictor can be derived using the Gram-Schmidt process developed in Chapter 3 in which the $i$th-order backward prediction errors defined by the set $\{e_b(t, i)\}$ are obtained from the set of measurements $\{y(n)\}$, $n = t - N_a, \ldots, t - 1, t$. Therefore, using the Gram-Schmidt

transformation, we obtain

$$\begin{bmatrix} e_b(t,0) \\ e_b(t,1) \\ e_b(t,2) \\ \vdots \\ e_b(t,N_a) \end{bmatrix} = \begin{bmatrix} 1 & & & & \\ a(1,1) & 1 & & & \mathbf{O} \\ a(2,2) & a(2,1) & 1 & & \\ \vdots & \vdots & & \ddots & \\ a(N_a,N_a) & a(N_a,N_a-1) & \cdots & & 1 \end{bmatrix} \begin{bmatrix} y(t) \\ y(t-1) \\ y(t-2) \\ \vdots \\ y(t-N_a) \end{bmatrix}$$

or compactly

$$\mathbf{e}_b(N_a) = \mathbf{L}(N_a)\mathbf{y}(N_a) \tag{4.30}$$

for $\mathbf{e}_b$, $\mathbf{y} \in \mathcal{R}^{(N_a+1)\times 1}$ and $\mathbf{L} \in \mathcal{R}^{(N_a+1)\times(N_a+1)}$.

Now using the Cholesky decomposition of Eq. (4.29) and the Toeplitz matrix structure of Eq. (4.13) we have that

$$\begin{aligned} R_{e_be_b}(N_a) &= E\{\mathbf{e}_b(N_a)\mathbf{e}_b'(N_a)\} = E\{\mathbf{L}(N_a)\mathbf{y}(N_a)\mathbf{y}'(N_a)\mathbf{L}'(N_a)\} \\ &= \mathbf{L}(N_a)R_{yy}(N_a)\mathbf{L}'(N_a) \end{aligned} \tag{4.31}$$

Since the $\{e_b(t,i)\}$ are developed using the Gram-Schmidt construction procedure, they are *orthogonal*. Therefore $R_{e_be_b} > 0$ is *diagonal* and positive definite. The diagonals are defined by

$$R_{ee}(i) > 0, \qquad i = 0, \ldots, N_a \tag{4.32}$$

From the Levinson-Durbin recursion, we have that

$$R_{ee}(i) = \left(1 - k_i^2\right) R_{ee}(i-1) \tag{4.33}$$

From the $R_{ee} > 0$ property, the reflection coefficients must satisfy

$$|k_i| < 1, \qquad i = 1, \ldots, N_a \tag{4.34}$$

Ultimately we are interested in $R_{yy}^{-1}$ to solve the basic model-based estimation problem of Eq. (4.15). Therefore inverting Eq. (4.31), we have

$$R_{e_be_b}^{-1}(N_a) = \mathbf{L}^{-T}(N_a)R_{yy}^{-1}(N_a)\mathbf{L}^{-1}(N_a) \tag{4.35}$$

with $L^{-T} := \left(L'\right)^{-1}$ and

$$R_{yy}^{-1}(N_a) = \mathbf{L}'(N_a)R_{e_be_b}^{-1}(N_a)\mathbf{L}(N_a) \tag{4.36}$$

This result shows that if we can estimate the predictor coefficients $a(i, j)$ from the data (e.g. *LD* recursion) and construct $\mathbf{L}(N_a)$, then the inverse is easily obtained from the Cholesky factorization and the set of prediction error variances $\{R_{ee}(i)\}$ that form the diagonals of $R_{e_b e_b}$.

Finally, returning to the optimal batch solution of Eq. (4.15), we see that

$$\hat{\mathbf{a}}(N_a) = -\mathbf{R}_{yy}^{-1}(N_a)\mathbf{r}_{yy}(N_a) = -\left(\mathbf{L}'(N_a)R_{e_b e_b}^{-1}(N_a)\mathbf{L}(N_a)\right)\left(\mathbf{L}^{-1}(N_a)\mathbf{r}_{e_b e_b}(N_a)\right)$$

Alternatively, performing the indicated operations, we have

$$\hat{\mathbf{a}}(N_a) = -\mathbf{L}'(N_a)R_{e_b e_b}^{-1}(N_a)\mathbf{r}_{e_b e_b}(N_a) \tag{4.37}$$

which follows from

$$\mathbf{r}_{yy}(N_a) = E\{\mathbf{y}(N_a)y(t)\} = E\{\mathbf{e}_b(N_a)e_b(t)\} = \mathbf{r}_{e_b e_b}(N_a) \tag{4.38}$$

These expressions demonstrate the relationship of the backward prediction error solution by pre-whitening the data to the standard approach. A few facts should be extracted from this exposition. First, by performing the Cholesky decomposition of $R_{yy}$, the original inversion becomes trivial (reciprocal of the diagonals of $R_{ee}$). Second, we see that this solution is based on the forward-backward prediction error filters that evolve into the lattice forms in a subsequent section. Finally, the lattice approach using the Burg algorithm does not require the data correlation matrix directly, so these coefficients can be estimated recursively.

### 4.2.3 Model-Based *AR* Spectral Estimation

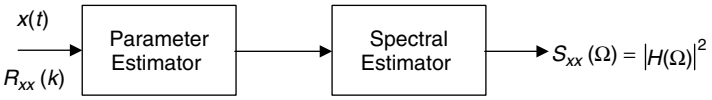
The model-based approach to spectral estimation is based on the underlying assumption that the measured data under investigation evolved from a process that can be represented or approximately represented by the selected model (all-pole) set. The MB spectral estimation approach is a three-step procedure:

- Select the model set.
- Estimate the model parameters from the data or correlation lags.
- Obtain the spectral estimate by using the (estimated) model parameters.

The major implied advantage of these model-based methods is that higher frequency resolution is achievable, since the data are not windowed [12]. It can be shown that these methods imply an infinite extension of the autocorrelation sequence [10]. We also note that this approach is indirect, since we must first find the appropriate model and then use it to estimate the *PSD*.

Let us recall the basic spectral relations developed previously for linear systems with random inputs. Recall that the measurement of the output spectrum of a process is related to the input spectrum by the factorization relations:

$$S_{yy}(z) = H(z)H^*(z)S_{xx}(z) = |H(z)|^2 S_{xx}(z) \tag{4.39}$$




$$\text{AR}: \quad H_{AR}(\Omega) = \frac{1}{A(\Omega)}$$

$$\text{MA}: \quad H_{MA}(\Omega) = C(\Omega)$$

$$\text{ARMA}: \quad H_{ARMA}(\Omega) = \frac{C(\Omega)}{A(\Omega)}$$

**Figure 4.3.** Model-based spectral estimation approach.

where $x$ is the input process, $H$ is the linear system transfer function, and $y$ is the measurement.

Since we are taking the signal processing viewpoint, then we will assume that *only* the measured data, $y(t)$, is available and not the excitation, $x(t)$. In fact, if both $x(t)$ and $y(t)$ are available, then the problem is called the system identification problem (see [7] for details). However, in this section we restrict ourselves to purely signal representations and select the *AR* model sets as representative for spectral estimation.

In summary, the model-based method of *spectral estimation* (using *ARMA* models) is given by (see Figure 4.3) the next three steps:

1. Select a representative model set (*AR*, *MA*, *ARMA*).
2. Estimate the model parameters from the data; that is, given $\{y(t)\}$, find

$$\hat{\Theta} = \left\{\hat{A}(q^{-1}), \hat{C}(q^{-1})\right\}$$

   and $\hat{R}_{ee}$.
3. Estimate the *PSD* using these parameters; that is,

$$S_{yy}(z) \approx \hat{S}_{yy}(z, \hat{\Theta}) = \hat{H}(z)\hat{H}^*(z)S_{ee}(z)$$

   where $H(z) = C(z)/A(z)$ and $S_{ee}(z) = R_{ee}$.

The autoregressive (*AR*) or all-pole model was given in Eq. (4.3). Taking $Z$-transforms, we obtain

$$H_{AR}(z) = \frac{Y(z)}{E(z)} = \frac{1}{A(z^{-1})} \tag{4.40}$$

Recall that

$$A(z^{-1}) = 1 + a_1 z^{-1} + \cdots + a_{N_a} z^{-N_a}$$

Substituting $H_{AR}$ for $H$ above, we have the model-based *PSD*:

$$S_{yy}(z) = H_{AR}(z)H_{AR}^{*}(z)S_{ee}(z) = |H_{AR}(z)|^2 S_{ee}(z) \tag{4.41}$$

or

$$S_{yy}(z) = R_{ee}\triangle T\left(\frac{1}{A(z^{-1})}\right)\left(\frac{1}{A(z)}\right) = \frac{R_{ee}\triangle T}{|A(z)|^2} \tag{4.42}$$

which is the desired representation of the all-pole model-based spectral density. Note that the spectral estimation procedure requires that we first estimate the AR model parameters, $(\{\hat{a}_i\},\ \hat{R}_{ee})$ from the data and then form the estimate[1]

$$S_{AR}(\Omega) := \hat{S}_{yy}(\Omega) = \hat{S}_{yy}(z)\Big|_{z=e^{j\Omega}} = \frac{\hat{R}_{ee}\triangle T}{|\hat{A}(e^{j\Omega})|^2} \tag{4.43}$$

We summarize the model-based spectral estimation method using the *AR* models as follows:

| | |
|---|---|
| Criterion: | $J = E\{e^2(t)\}$ |
| Models: | |
| Measurement: | $y(t) = s(t) + e(t)$ |
| Signal: | $s(t) = -\sum_{i=1}^{N_a} a_i y(t-i)$ |
| Noise: | $R_{ee}$ |
| Algorithm: | $S_{yy}(\Omega) = \dfrac{\hat{R}_{ee}\triangle T}{\lvert\hat{A}(e^{j\Omega})\rvert^2}$ |
| Quality: | $R_{ee}(N_a)$ |

Next consider the following example of model-based spectral estimation.

***Example 4.3*** Consider the problem of estimating sinusoids in bandpass noise at a 0 dB *SNR*. Here 101 realizations of this spectrum are generated with sinusoids at {35, 50, 56, 90 Hz} with the bandpass filter at cutoff frequencies of {30, 70 Hz}. We selected an *AR* model of order 35. The results are shown in Figure 4.4 where we see the ensemble estimates and the corresponding mean followed by the average spectral estimate with peaks at: {35.3, 49.0, 56.9, 90.2 Hz}. Clearly, the estimates are quite reasonable, demonstrating the effectiveness of the model-based approach.

[1]A popular method of "simulating" the spectrum at equally spaced frequencies is to find the DFT of $A(e^{j\Omega})$, multiply by its conjugate, and divide into $\hat{R}_{ee}$.

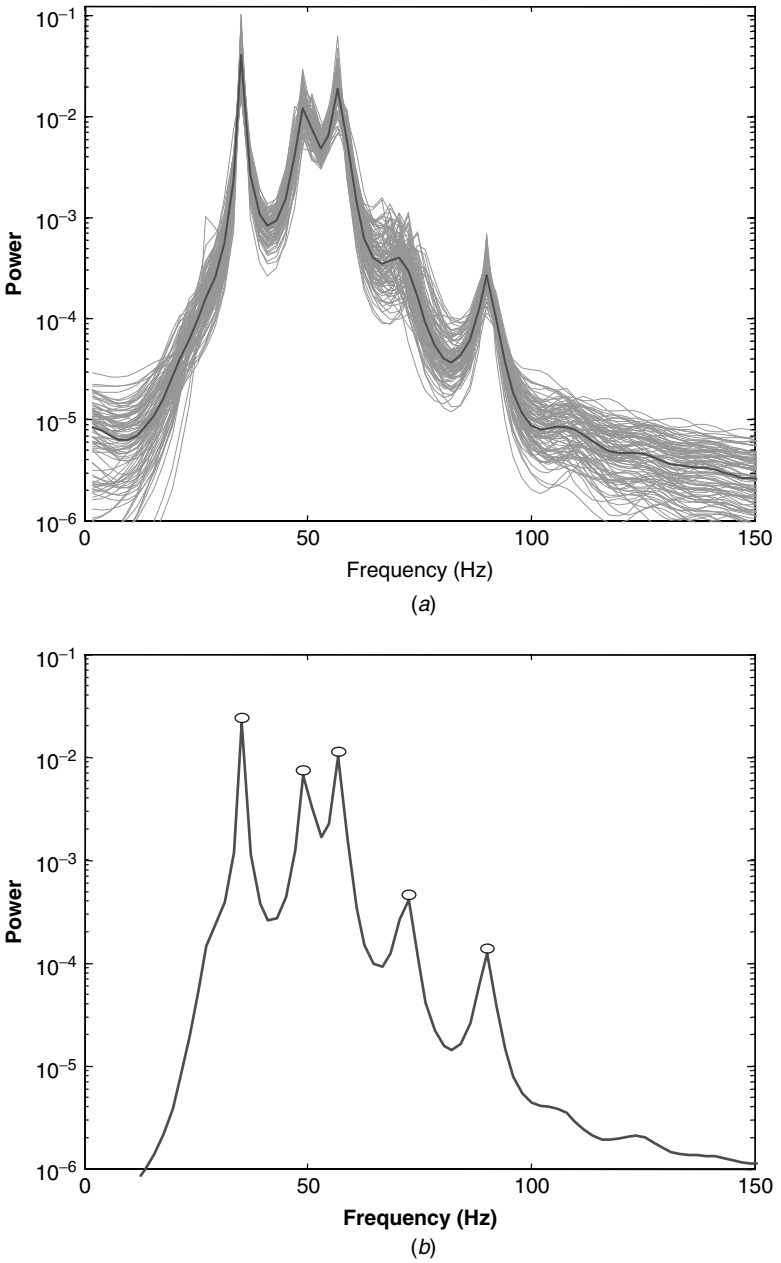



**Figure 4.4.** Model-based spectral estimation for bandpass sinusoids in noise simulation. (*a*) Ensemble of 35th order *AR* estimator with mean estimate shown. (*b*) Average spectral estimate with peaks at {35.3, 49.0, 56.9, 90.2 Hz}.

## 4.3 *MA* (ALL-ZERO) *MBP*

In this section we develop the *MA* or all-zero filters leading to an finite impulse-response (*FIR*) model.[2] The technique discussed is based on the Levinson-Durbin (*LD*) recursion of the previous section and is sometimes called the "generalized Levinson" or Levinson-Wiggins-Robinson (*LWR*) algorithm [13]. Again, the technique is based on inverting a Toeplitz matrix, but now for the case where the right side vector is a cross-rather than autocovariance. In fact, this method provides a recursive solution to the Wiener filtering problem for the scalar as well as multichannel cases [14]. This model set utilized provides the basic building block of many techniques used in adaptive signal processing [15].

Suppose that we excite a system with an input, say $x(t)$, and we measure the output, $y(t)$. We would like to obtain a linear estimate of the embedded signal $s(t)$ based on past excitation data, that is,

$$\hat{s}(t) = \hat{s}(t|X_{past})$$

Taking the model-based (parametric) approach, we model the signal by the *MA* (all-zero) representation

$$\hat{s}(t) = \sum_{i=0}^{N_h} h(i)x(t-i) \tag{4.44}$$

If we represent the measurement $y(t)$ as

$$y(t) = s(t) + n(t) \tag{4.45}$$

where $n$ is zero-mean, white noise of variance $R_{nn}$,[3] then the optimal estimator is obtained by

$$\min_{\mathbf{h}} \mathcal{J}(t) = E\{e^2(t)\} \tag{4.46}$$

where $e(t) := y(t) - \hat{s}(t)$.

Following the standard minimization approach, we differentiate Eq. (4.46) with respect to $h(j)$, set the result to zero, and obtain the *orthogonality condition*

$$\frac{\partial \mathcal{J}}{\partial h(j)} = 2E\left\{e(t)\frac{\partial e(t)}{\partial h(j)}\right\} = 0 \tag{4.47}$$

[2]We use the term *MA* "loosely" here, since our excitation is *not* random as required by the *MA* model. It is actually an *FIR* model with a noisy input excitation. We will use the terminology interchangeably, since neither is true in the "pure" sense. However, the final result will be a finite, *all-zero* response.

[3]Note that this problem is different from the previous in that here we have *both* input and output sequences, $\{x(t)\}$ and $\{y(t)\}$ respectively, while for the all-pole filter we just had $\{y(t)\}$. This problem is called the "joint process" estimation problem.

The error gradient is

$$\frac{\partial e(t)}{\partial h(j)} = \frac{\partial}{\partial h(j)}\left(y(t) - \hat{s}(t)\right) = \frac{\partial}{\partial h(j)}\left(y(t) - \sum_{i=0}^{N_h} h(i)x(t-i)\right) = x(t-j) \tag{4.48}$$

Therefore substituting Eq. (4.48) into Eq. (4.47), we obtain

$$E\left\{e(t)x(t-j)\right\} = E\left\{\left(y(t) - \sum_{i=0}^{N_h} h(i)x(t-i)\right)x(t-j)\right\} = 0, \;\; j = 0, \ldots, N_h \tag{4.49}$$

which yields the *normal equations* for the joint estimation problem as

$$\sum_{i=0}^{N_h} h(i)R_{xx}(i-j) = r_{yx}(j) \quad j = 0, \ldots, N_h \tag{4.50}$$

Alternatively, expanding the equations over the indexes gives

$$\begin{bmatrix} R_{xx}(0) & \cdots & R_{xx}(N_h) \\ \vdots & \ddots & \vdots \\ R_{xx}(N_h) & \cdots & R_{xx}(0) \end{bmatrix} \begin{bmatrix} h(0) \\ \vdots \\ h(N_h) \end{bmatrix} = \begin{bmatrix} r_{yx}(0) \\ \vdots \\ r_{yx}(N_h) \end{bmatrix} \tag{4.51}$$

This expression can be written in compact form as

$$\mathbf{R}_{xx}(N_h)\mathbf{h}(N_h) = \mathbf{r}_{yx}(N_h) \tag{4.52}$$

where $\mathbf{R}_{xx} \in \mathcal{R}^{(N_h+1)\times(N_h+1)}$ is a Toeplitz matrix. The optimal estimate, $\hat{h}(N_h)$ is obtained by inverting $\mathbf{R}_{xx}$ using the "generalized Levinson" or equivalently the *LWR* algorithm [13]. Note that this is the Wiener solution, since

$$\hat{\mathbf{h}}(N_h) = \mathbf{R}_{xx}^{-1}(N_h)\mathbf{r}_{yx}(N_h) \tag{4.53}$$

Note that the properties of the Toeplitz matrix decomposition can be applied to solve this problem in "batch" form as before. The backward prediction errors are developed by performing a Gram-Schmidt orthogonalization on the set of input data, $\{x(n)\}, n = t - N_h, \ldots, t-1, t$ as

$$\mathbf{e}_b(N_h) = \mathbf{L}(N_h)\mathbf{x}(N_h) \tag{4.54}$$

Following the same arguments as before

$$\begin{aligned} R_{xx}(N_h) &= E\{\mathbf{x}(N_h)\mathbf{x}'(N_h)\} \\ &= E\{(\mathbf{L}^{-1}(N_h)\mathbf{e}_b(N_h))(\mathbf{L}^{-1}(N_h)\mathbf{e}_b(N_h))'\} \end{aligned}$$

or

$$R_{xx}(N_h) = \mathbf{L}^{-1}(N_h)E\{\mathbf{e}_b(N_h)\mathbf{e}_b'(N_h)\}\mathbf{L}^{-T}(N_h)$$
$$= \mathbf{L}^{-1}(N_h)R_{e_be_b}(N_h)\mathbf{L}^{-T}(N_h) \tag{4.55}$$

Its corresponding inverse is

$$R_{xx}^{-1}(N_h) = \mathbf{L}'(N_h)R_{e_be_b}^{-1}(N_h)\mathbf{L}(N_h) \tag{4.56}$$

Similarly the cross-correlation vector becomes

$$\mathbf{r}_{yx}(N_h) = E\{y(t)\mathbf{x}(N_h)\} = E\{y(t)(\mathbf{L}^{-1}(N_h)\mathbf{e}_b(N_h))\}$$
$$= \mathbf{L}^{-1}(N_h)\mathbf{r}_{ye_b}(N_h) \tag{4.57}$$

Thus the Wiener solution of Eq. (4.53) becomes

$$\hat{\mathbf{h}}(N_h) = \left(\mathbf{L}'(N_h)R_{e_be_b}^{-1}(N_h)\mathbf{L}(N_h)\right)\left(\mathbf{L}^{-1}(N_h)\mathbf{r}_{ye_b}(N_h)\right)$$
$$= \mathbf{L}'(N_h)R_{e_be_b}^{-1}(N_h)\mathbf{r}_{ye_b}(N_h) \tag{4.58}$$

which is equivalent to pre-whitening the input data.

We summarize the batch all-zero approach as follows:

Criterion: $J = E\{e^2(t)\}$
Models:
  Measurement: $y(t) = s(t) + n(t)$
  Signal: $s(t) = -\sum_{i=1}^{N_h} h_i x(t-i)$
  Noise: $R_{nn}$
Algorithm: $\hat{\mathbf{h}} = \mathbf{R}_{xx}^{-1}\mathbf{r}_{yx}$
Quality: $R_{ee}(N_h)$

This completes the discussion of the batch approach, next we develop a recursive solution.

### 4.3.1 Levinson-Wiggins-Robinson (*LWR*) Recursion

The Toeplitz matrix inversion can be performed using the *LWR* recursion which will be developed in two steps. The first establishes the basic recursion, and the second is the standard Levinson-Durbin recursion for inverting Toeplitz matrices. We will use the notation $\{h(i,k)\}$ to denote the $i$th coefficient of the $k$th-order filter. Let us assume that we have the $k$th-order filter that satisfies the set of *normal*

*equations*:

$$\begin{bmatrix} R_{xx}(0) & \cdots & R_{xx}(N_h) \\ \vdots & \ddots & \vdots \\ R_{xx}(N_h) & \cdots & R_{xx}(0) \end{bmatrix} \begin{bmatrix} h(0, N_h) \\ \vdots \\ h(N_h, N_h) \end{bmatrix} = \begin{bmatrix} r_{yx}(0) \\ \vdots \\ r_{yx}(N_h) \end{bmatrix} \tag{4.59}$$

We want the $(N_h + 1)^{\text{th}}$-order solution given by

$$\begin{bmatrix} R_{xx}(0) & \cdots & R_{xx}(N_h + 1) \\ \vdots & \ddots & \vdots \\ R_{xx}(N_h + 1) & \cdots & R_{xx}(0) \end{bmatrix} \begin{bmatrix} h(0, N_h + 1) \\ \vdots \\ h(N_h + 1, N_h + 1) \end{bmatrix} = \begin{bmatrix} r_{yx}(0) \\ \vdots \\ r_{yx}(N_h + 1) \end{bmatrix} \tag{4.60}$$

Suppose the optimum solution for the $(N_h + 1)$th-order filter is given by the $N_h$th order, then $\mathbf{h}'(N_h + 1) = [\mathbf{h}'(N_h) \mid 0]$ and $\mathbf{r}'_{yx}(N_h + 1) = [\mathbf{r}'_{yx}(N_h) \mid \nabla_{N_h}]$ with $\nabla_{N_h} = r_{yx}(N_h + 1)$. We can rewrite Eq. (4.60) as

$$\begin{bmatrix} \mathbf{R}_{xx}(N_h) & \mid & R_{xx}(N_h + 1) \\ - - - - - & \mid & \vdots \\ R_{xx}(N_h + 1) & \cdots & R_{xx}(0) \end{bmatrix} \begin{bmatrix} \mathbf{h}(N_h) \\ \cdots \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{yx}(N_h) \\ \cdots \\ \nabla_{N_h} \end{bmatrix} \tag{4.61}$$

where, in general, we have

$$\nabla_i = \sum_{j=0}^{N_h} h(j, i) R_{xx}(i - j + 1) \qquad \text{for} \quad i = 0, \ldots, N_h \tag{4.62}$$

We must perform operations on Eq. (4.61) to ensure that $\nabla_{N_h} = r_{yx}(N_h + 1)$ to obtain the correct solution. Let us assume there exists a solution $\{\alpha(i, N_h + 1)\}$ such that[4]

$$\begin{bmatrix} R_{xx}(0) & \cdots & R_{xx}(N_h + 1) \\ \vdots & \ddots & \vdots \\ R_{xx}(N_h + 1) & \cdots & R_{xx}(0) \end{bmatrix} \begin{bmatrix} \alpha(0, N_h + 1) \\ \vdots \\ \alpha(N_h + 1, N_h + 1) \end{bmatrix} = - \begin{bmatrix} J(N_h + 1) \\ \vdots \\ 0 \end{bmatrix} \tag{4.63}$$

Now, by elementary manipulations, we can reverse the order of the components, multiply by a constant, $k_{N_h+1}$, and subtract the result from Eq. (4.61), that is,

[4]Note that the solution to this set of equations results in the standard Levinson-Durbin recursion of linear prediction theory discussed in the previous section.

$$\mathbf{R}_{xx}(N_h+1)\left\{\begin{bmatrix} h(0,N_h) \\ \vdots \\ h(N_h,N_h) \\ 0 \end{bmatrix} - k_{N_h+1}\begin{bmatrix} \alpha(N_h+1,N_h+1) \\ \vdots \\ \alpha(1,N_h+1) \\ \alpha(0,N_h+1) \end{bmatrix}\right\}$$

$$= \left\{\begin{bmatrix} r_{yx}(0) \\ \vdots \\ r_{yx}(N_h) \\ \nabla_{N_h} \end{bmatrix} - k_{N_h+1}\begin{bmatrix} 0 \\ \vdots \\ 0 \\ J(N_h+1) \end{bmatrix}\right\}$$

Alternatively

$$\mathbf{R}_{xx}(N_h+1)\left\{\begin{bmatrix} h(0,N_h) - k_{N_h+1}\alpha(N_h+1,N_h+1) \\ \vdots \\ h(N_h,N_h) - k_{N_h+1}\alpha(1,N_h+1) \\ -k_{N_h+1}\alpha(0,N_h+1) \end{bmatrix}\right\}$$

$$= \begin{bmatrix} r_{yx}(0) \\ \vdots \\ r_{yx}(N_h) \\ \nabla_{N_h} - k_{N_h+1}J(N_h+1) \end{bmatrix} \tag{4.64}$$

The multiplier $k_{N_h+1}$ is selected so that

$$\nabla_{N_h} - k_{N_h+1}J(N_h+1) = r_{yx}(N_h+1)$$

By identifying the coefficients $\mathbf{h}(N_h+1)$, $\mathbf{r}_{yx}(N_h+1)$ from Eq. (4.64) with $\alpha(0,i) = 1$, we obtain the first part of the recursion shown in Table 4.2.

In order to satisfy this recursion, we must also obtain the predictor, $\{\alpha(j,i+1)\}$, and $J(i+1)$ from the solution of Eq. (4.63). This can be accomplished using the *Levinson-Durbin* recursion of Table 4.1 for the $\{\alpha(j,i)\}$ and $J(i)$ as shown in Table 4.2. This completes the solution to the Toeplitz inversion using the generalized Levinson or *LWR* algorithm. Next consider an example.

***Example 4.4*** Suppose that we have a signal that is exponentially correlated with $R_{xx}(k) = \alpha^{|k|}$ and it is transmitted through a medium with a velocity of 1 m/s and attenuation $A$. The sensor is placed $d = 4$ m from the source, find the optimal (*FIR*) estimate of the signal, that is, $\hat{s}(t|t-1)$. From the characteristics of the medium we have

$$\tau = \frac{d}{c} = \frac{4 \text{ m}}{1 \text{ m/s}} = 4 \text{ s}$$

Therefore the received signal is

$$r(t) = Ax(t-4) + n(t)$$

**Table 4.2. Generalized Levinson or *LWR* Recursion**

Initialize

$$J(0) = R_{xx}(0),\ \ h(0,0) = \frac{R_{yx}(0)}{J(0)}, \alpha(0,i) = 1 \qquad \forall i$$

For $i = 0, \ldots, N_h$

$$\nabla_i = \sum_{j=0}^{i} h(j,i) R_{xx}(i-j+1)$$

$$k_{i+1} = \frac{\nabla_i - R_{yx}(i+1)}{J(i+1)}$$

$$h(i+1,i+1) = -k_{i+1}$$

$$h(j,i+1) = h(j,i) - k_{i+1}\alpha(i-j+1,i+1) \qquad \text{for } j = 0, \ldots, i$$

$\alpha(i,j)$ and $J(i)$ are obtained from the Levinson-Durbin recursion as

For $i = 1, \ldots, N_h$

$$\triangle_i = \sum_{j=0}^{i} \alpha(j,i) R_{xx}(i-j+1)$$

$$k_{i+1}^* = \frac{\triangle_i}{J(i)}$$

$$\alpha(i+1,i+1) = -k_{i+1}^*$$

$$\alpha(j,i+1) = \alpha(j,i) - k_{i+1}^* \alpha(i-j+1,i) \qquad \text{for } j = 1, \ldots, i$$

$$J(i+1) = \left(1 - \left(k_{i+1}^*\right)^2\right) J(i)$$

Signal estimation

$$\hat{s}(t|t-1) = \sum_{i=0}^{N_h} \hat{h}(i) x(t-i)$$

and

$$R_{rx}(k) = E\{r(t)x(t+k)\} = AE\{x(t-4)x(t+k)\} = AR_{xx}(k-4) = A\alpha^{|k-4|}$$

*Batch approach*: Using the optimal Wiener solution, we have

$$R_{xx}\mathbf{h} = \mathbf{r}_{rx}$$

$$\begin{bmatrix} 1 & \alpha \\ \alpha & 1 \end{bmatrix} \begin{bmatrix} h(0) \\ h(1) \end{bmatrix} = A \begin{bmatrix} \alpha^4 \\ \alpha^3 \end{bmatrix}$$

$$\mathbf{h} = R_{xx}^{-1}\mathbf{r}_{rx}$$

$$\begin{bmatrix} h(0) \\ h(1) \end{bmatrix} = \left(\frac{A}{1-\alpha^2}\right)\begin{bmatrix} 1 & -\alpha \\ -\alpha & 1 \end{bmatrix}\begin{bmatrix} \alpha^4 \\ \alpha^3 \end{bmatrix}$$

$$= \frac{A}{(1-\alpha^2)}\begin{bmatrix} 0 \\ \alpha^3(1-\alpha^2) \end{bmatrix} = \begin{bmatrix} 0 \\ A\alpha^3 \end{bmatrix}$$

which gives the signal estimate

$$\hat{s}(t) = A\alpha^3 x(t-1)$$

*LWR approach*: Initialize

$$J(0) = R_{xx}(0) = 1, \quad h(0,0) = \frac{R_{yx}(0)}{J(0)} = A\alpha^4, \quad \alpha(0,i) = 1 \qquad \forall i$$

$$\triangle_0 = \alpha(0,0)R_{xx}(1) = \alpha \qquad (i=0)$$

$$k_1^* = \frac{\triangle_0}{J(0)} = \alpha$$

$$\alpha(1,1) = -k_1^* = -\alpha$$

$$\nabla_0 = h(0,0)R_{xx}(1) = A\alpha^4(\alpha) = A\alpha^5$$

$$J(1) = \left(1-(k_1^*)^2\right)J(0) = 1-\alpha^2$$

$$k_1 = \frac{\nabla_0 - R_{yx}(1)}{J(1)} = \frac{A(\alpha^5-\alpha^3)}{1-\alpha^2} = -A\alpha^3$$

$$h(0,1) = h(0,0) - k_1\alpha(1,1) = A\alpha^4 + A\alpha^3(-\alpha) = 0$$

$$h(1,1) = -k_1 = A\alpha^3$$

which is identical to the batch approach.

Next we consider processing the simulation problem of the previous section.

***Example 4.5*** We would like to design a Levinson all-zero filter for the *ARMAX*(2, 1, 1, 1) simulated data of the previous example. As before, we utilize *MATLAB* [11] to perform both simulation and estimation. A 32-weight *FIR* Levinson filter was designed using the *LWR* algorithm of Table 4.2. The estimated parameters are given by

| Parameter | True | Estimated |
|---|---|---|
| $h(0)$ | 1 | 1 |
| $h(1)$ | 2 | 1.944 |
| $h(2)$ | 2.20 | 2.24 |
| $h(3)$ | 2.05 | 1.95 |
| $h(4)$ | 1.47 | 1.39 |
| $h(5)$ | 0.76 | 0.68 |
| $h(6)$ | 0.12 | 0.11 |

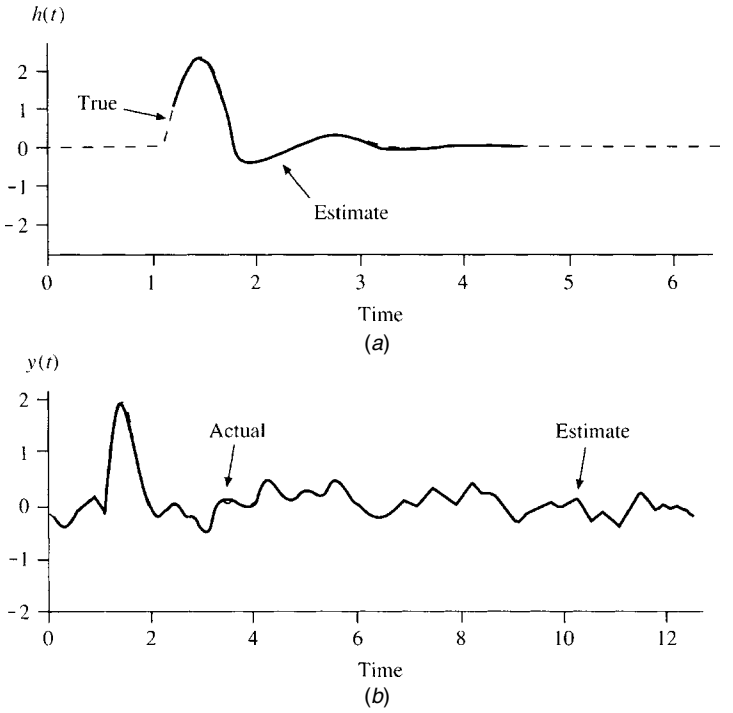



**Figure 4.5.** Levinson all-zero filter design for *ARMAX*(2, 1, 1, 1) simulation. (*a*) True and estimated impulse response. (*b*) True and estimated measurement (signal).

The results are shown in Figure 4.5 where we see the impulse response and reconstructed measurement. Here we see that the *FIR* approach more closely estimates the impulse response and measurement at the cost of a higher order filter.

Before closing the section on *FIR* Wiener solutions, it is interesting to note that if we replace $\{h(i)\} \rightarrow \{b_i\}$ of the $ARMAX(1, N_b, 0)$ model, then we have these Wiener solutions as special cases of the *ARMAX* solution. Also it is important to note that the *LWR* recursion leads to an *order recursive* solution of the Toeplitz matrix. The *FIR* Wiener solution is the basis of most of the popular adaptive solutions (see Widrow [15]). In fact, for the stationary case, the *FIR* adaptive solution converges precisely to $\mathbf{h}(N)$ of Eq. (4.53). Next we show how this technique leads to the solution of an important problem.

### 4.3.2 Optimal Deconvolution

The Wiener solution also provides an optimal method of solving the deconvolution problem. It is easy to derive the model-based processor by using the commutative property of the convolution relation, that is,

$$y(t) = h(t) * x(t) = x(t) * h(t) \tag{4.65}$$

The deconvolution problem is concerned with obtaining the "optimal" estimate of $x(t)$ given $(\{y(t)\}, \{h(t)\})$ in contrast to the Wiener filter design or identification problem where we want to estimate the impulse response, $h(t)$ given $(\{x(t)\}, \{y(t)\})$. Proceeding in similar fashion as before, we give the optimal estimator by

$$\min_{\mathbf{x}} \mathcal{J}(t) = \tfrac{1}{2} E\{e^2(t)\} \tag{4.66}$$

where $e(t) := y(t) - \hat{y}(t)$ and

$$\hat{y}(t) = x(t) * h(t) = \sum_{i=0}^{N_x} x(i)h(t-i) \tag{4.67}$$

Here $x(t)$ plays the role of the filter and the $h(t)$ that of the "known" input of the filter design problem. Following the standard approach, we give the *orthogonality condition* by (with $h \to x$)

$$\frac{\partial}{\partial x(j)} \mathcal{J}(t) = E\left\{ e(t) \frac{\partial e(t)}{\partial x(j)} \right\} = 0 \qquad \text{for } j = 0, \ldots, N_x \tag{4.68}$$

The error gradient is therefore

$$\frac{\partial e(t)}{\partial x(j)} = \frac{\partial}{\partial x(j)} (y(t) - \hat{y}(t)) = \frac{\partial}{\partial x(j)} \left( y(t) - \sum_{i=0}^{N_x} x(i)h(t-i) \right) = h(t-j) \tag{4.69}$$

Substituting, we obtain the *normal* equations,

$$E\{y(t)h(t-j)\} - \sum_{i=0}^{N_x} x(i) E\{h(t-i)h(t-j)\} = 0 \qquad \text{for } j = 0, \ldots, N_x$$

Solving, we have

$$\sum_{i=0}^{N_x} x(i) R_{hh}(i-j) = r_{yh}(j) \qquad \text{for } j = 0, \ldots, N_x \tag{4.70}$$

Expanding this equation obtains

$$\begin{bmatrix} R_{hh}(0) & \cdots & R_{hh}(N_x) \\ \vdots & \ddots & \vdots \\ R_{hh}(N_x) & \cdots & R_{hh}(0) \end{bmatrix} \begin{bmatrix} x(0, N_x) \\ \vdots \\ x(N_x, N_x) \end{bmatrix} = \begin{bmatrix} r_{yh}(0) \\ \vdots \\ r_{yh}(N_x) \end{bmatrix} \tag{4.71}$$

We can write it in compact form as

$$\mathbf{R}_{hh}(N_x)\mathbf{x}(N_x) = \mathbf{r}_{yh}(N_x) \tag{4.72}$$

where $\mathbf{R}_{hh} \in \mathcal{R}^{(N_x+1)\times(N_x+1)}$ is Toeplitz and can be solved in "batch" form as

$$\hat{\mathbf{x}}(N_x) = \mathbf{R}_{hh}^{-1}(N_x)\mathbf{r}_{yh}(N_x) \tag{4.73}$$

or with the *LWR* recursion with $x \longrightarrow h$ of Table 4.2.

It is straightforward to show that if we perform the Gram-Schmidt orthogonalization (as before), then

$$\mathbf{h}_b(N_x) = L(N_x)\mathbf{h}(N_x) \tag{4.74}$$

This leads to

$$\mathbf{R}_{hh}(N_x) = L^{-1}(N_x)\mathbf{R}_{h_bh_b}(N_x)L^{-T}(N_x) \tag{4.75}$$

and therefore

$$\mathbf{R}_{hh}^{-1}(N_x) = L'(N_x)\mathbf{R}_{h_bh_b}^{-1}(N_x)L(N_x) \tag{4.76}$$

Similarly (see Eq. 4.51)

$$\mathbf{r}_{yh}(N_x) = L^{-1}(N_x)\mathbf{r}_{yh_b}(N_x) \tag{4.77}$$

which gives the optimal deconvolution in terms of the backward predictor coefficients as

$$\hat{\mathbf{x}}(N_x) = L'(N_x)\mathbf{R}_{h_bh_b}^{-1}(N_x)\mathbf{r}_{yh_b}(N_x) \tag{4.78}$$

We summarize the batch all-zero deconvolution approach as follows:

| | |
|---|---|
| Criterion: | $J = E\{e^2(t)\}$ |
| Models: | |
| Measurement: | $y(t) = h(t) * x(t)$ |
| Signal: | $x(t) = -\sum_{i=0}^{N_x} x(i)h(t-i)$ |
| Noise: | $R_{ee}$ |
| Algorithm: | $\hat{\mathbf{x}} = \mathbf{R}_{hh}^{-1}\mathbf{r}_{yh}$ |
| Quality: | $R_{ee}(N_x)$ |

***Example 4.6*** In this example we utilize the simulated *ARMAX*(2, 1, 1, 1) of measurement data from Example 4.41 and the Levinson all-zero algorithm of Table 4.2 to deconvolve or estimate the input according to Eq. (4.73). The impulse response is that obtained in the previous example (see Figure 4.5). Using *MATLAB* [11], we show the results of the solution in Figure 4.6. Ideally we should see a unit impulse at 1.2 second. We see from the figure that the algorithm performs quite well and is able to produce a good estimate of the desired excitation.

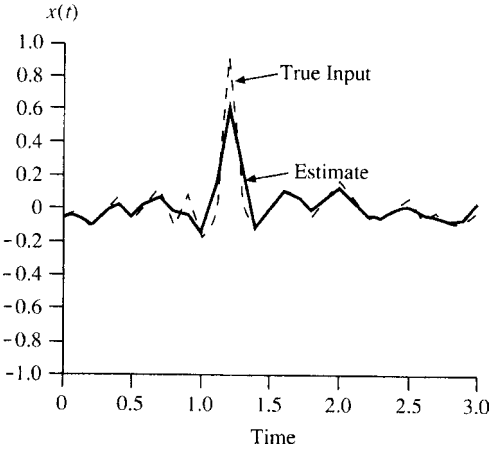



**Figure 4.6.** Levinson all-zero deconvolution filter design for *ARMAX*(2, 1, 1, 1) simulation.

### 4.3.3 Optimal Time Delay Estimation

In this subsection we develop an optimal time delay estimation scheme using a model-based approach; that is, we motivate the development of an all-zero or *FIR* model of the processor and demonstrate the solution using the *LWR* algorithm. The basic idea is to employ the *FIR* model as a delay line whose impulse response is given by

$$h(t) = \sum_{k=1}^{N_h} \alpha_k \delta(t - \tau_k) \tag{4.79}$$

where $\alpha$ represents the weights and $\tau$ the tap delays. The corresponding Fourier transform (continuous time) of this model is

$$H(f) = \sum_{k=1}^{N_h} \alpha_k e^{-j2\pi f(t-\tau_k)} \tag{4.80}$$

In discrete time, using a uniform sampling interval $\triangle T$, we obtain the discrete counterpart given by

$$h(t_n) = \sum_{k=1}^{N_h} \alpha_k \delta(t_n - (k-1)\triangle T) \tag{4.81}$$

which would consist of all zeros except the amplitude $\alpha_k$ at the time sample corresponding to the actual physical delay, that is, $\tau_k := (k-1)\triangle T$ (e.g., see Figure 4.7).

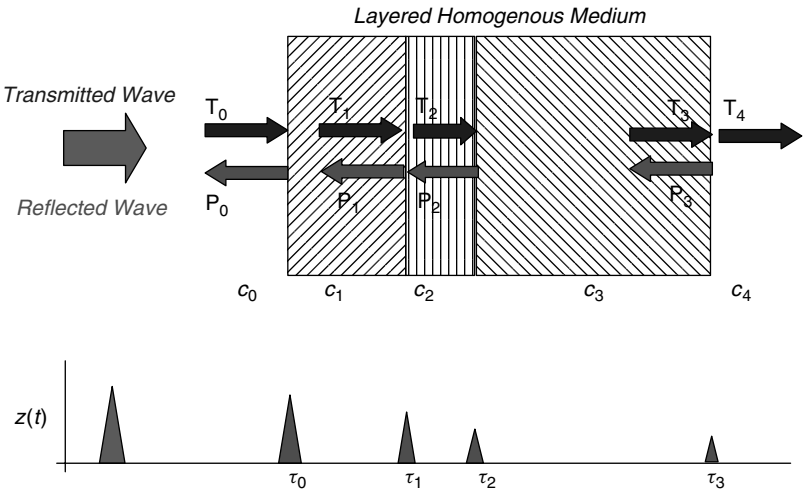



**Figure 4.7.** Layered homogeneous medium model showing layers, the layer velocities and the propagation pulse corresponding to each layer.

This model has applicability in many physical cases. For instance, suppose that we would like to characterize flaws in material for nondestructive evaluation (NDE) [16] or targets in the atmosphere for radar localization and tracking [17], or to detect tumors in breast tissue using ultrasonic transducers (active sonar) [18], or to determine the internal structure of the earth medium for seismic exploration [14]. The one-dimensional, homogeneous, wave-type model is characterized by

$$z(t) = g(r;t) * p(t) \tag{4.82}$$

where $z$ is the measured response, $p$ is the excitation pulse and $g(r;t)$ is the space-time impulse response or Green's function of the medium with $r$ the spatial coordinate [14]. We can assume that the homogeneous medium is layered with different velocities $\{c_k\}$ at the $k$th-layer, $k = 1, \ldots, K$. Therefore at each layer interface the wave energy will be transmitted and reflected based on the corresponding transmission and reflection coefficients. Thus a reasonable model of the *layered, homogeneous medium* is

$$g(r;t) = \sum_k \alpha_k \delta(t - \tau_k(r)) \quad \text{for} \quad \tau_k(r) := \frac{|r|}{c_k} \tag{4.83}$$

where $\alpha_k \leq 1$ is the corresponding attenuation coefficient at the $k$th-layer and $\tau_k(r)$ its associated time delay determined by the distance (thickness of layer) and propagation velocity. When the model is substituted in Eq. (4.82), we obtain

$$z(t) = g(r;t) * p(t) = \sum_k \alpha_k \delta(t - \tau_k(r)) * p(t) = \sum_k \alpha_k p(t - \tau_k(r)) \tag{4.84}$$

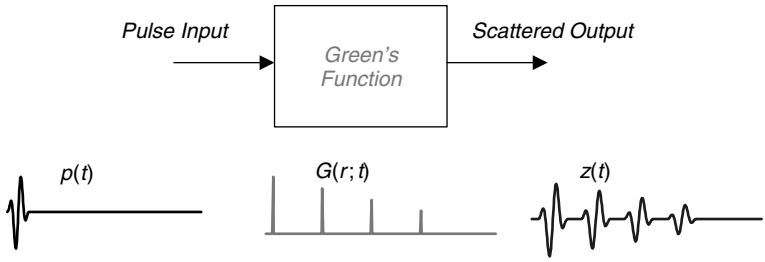



**Figure 4.8.** Linear system representation of layered, homogeneous medium model.

a delayed and attenuated replica of the excitation pulse. This medium is depicted in Figures 4.7 and 4.8.

If we view the time delay problem in the estimation framework in terms of the *FIR* model (see the Figure 4.8), we can define the problem as

> **GIVEN** a set of noisy measurements, $\{z(t)\}$, and excitation, $\{p(t)\}$, **FIND** the best estimate in a mean-squared error sense of the impulse response, $\hat{h}(t) \approx g(r;t)$.

With this model in mind, we can solve the basic problem as

$$\min_{\mathbf{h}} \mathcal{J} = E\{\epsilon^2(t)\} \quad \text{for} \quad \epsilon(t) := z(t) - \hat{\mathbf{h}}'\mathbf{p}(t) \tag{4.85}$$

leading to the batch Wiener solution of Eq. (4.53)

$$\hat{\mathbf{h}}(N_h) = \mathbf{R}_{pp}^{-1}(N_h)\mathbf{r}_{zp}(N_h) \tag{4.86}$$

Then we can apply the *LWR* of Table 4.2 as before.

***Example 4.7*** Consider the following simulations of a layered homogeneous medium for a nondestructive evaluation (NDE) application with the layer thickness varied to alter the corresponding time delays characterized by $\tau_k = |r_k|/c_k$ with $\{r_k, c_k\}$ the thickness and velocity of the $k$th-layer. For this example we have that $dt = 1$. Using the initial excitation pulse shown in Figure 4.9*a*, we developed the following cases:

Case 1: Separated layers *no* pulse overlap (delays:10,76,142,208 *seconds*)
Case 2: Separated layers *small* pulse overlap (delays:10,61,102,133 *seconds*)
Case 3: Separated layers *medium* pulse overlap (delays:10,41,72,103 *seconds*)
Case 4: Separated layers *large* pulse overlap (delays:10,31,52,73 *seconds*)

Each of the respective cases is depicted in Figure 4.9*b* through 4.9*e* with the corresponding response and superimposed delayed/attenuated Green's function medium shown. Applying the *LWR* algorithm to this data (noise free) results in a perfect

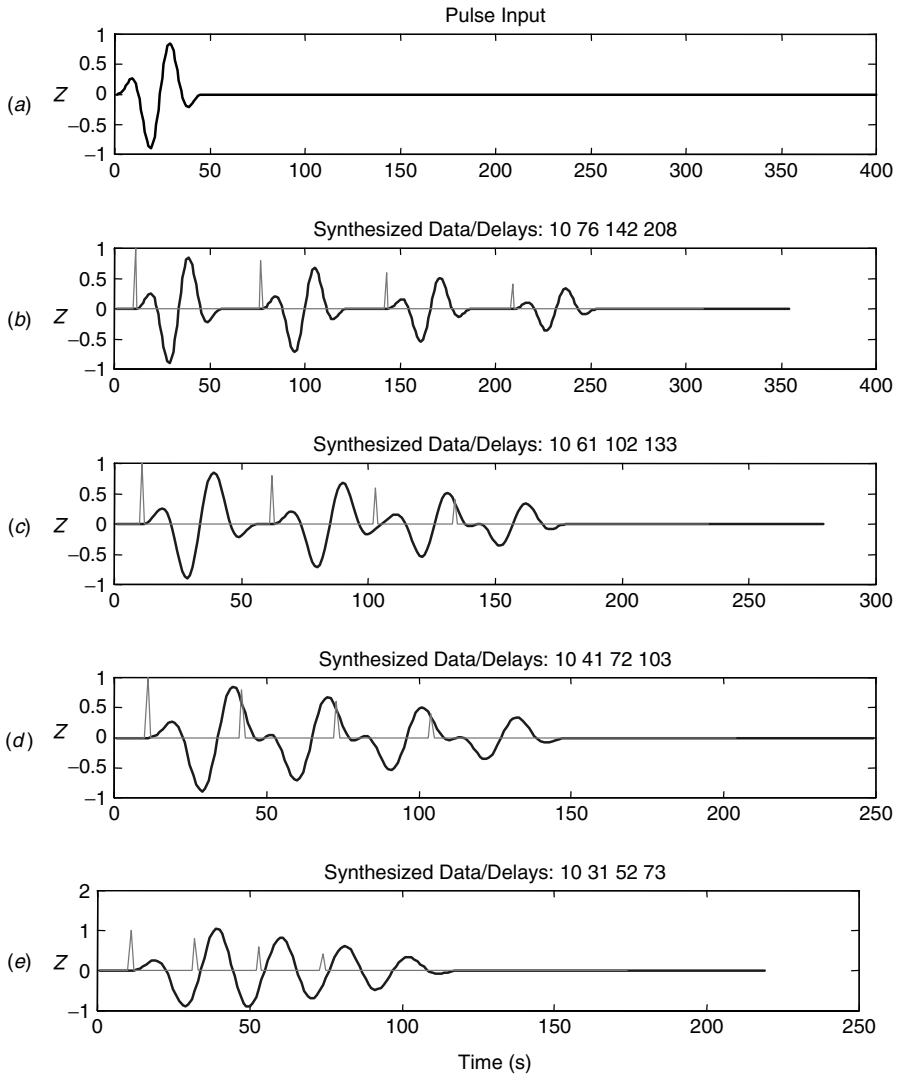



**Figure 4.9.** Optimal time delay simulation: (*a*) Excitation pulse. (*b*) No overlap for Case 1. (*c*) Small overlap for Case 2. (*d*) Medium overlap for Case 3. (*e*) Large overlap for Case 4.

estimation in each case as shown in Figure 4.10*a* through 4.10*d*, respectively. Here the *order* of the optimal processor is selected to be the length of the data record. So we see that the overlapping pulses have little impact on the optimal processor, since it is able to extract the delays perfectly in every case.

Next we apply this model-based approach to synthesized 5 MHz ultrasonic data sampled at 20 ns. The data, which was a prelude to a high-energy laser experiment

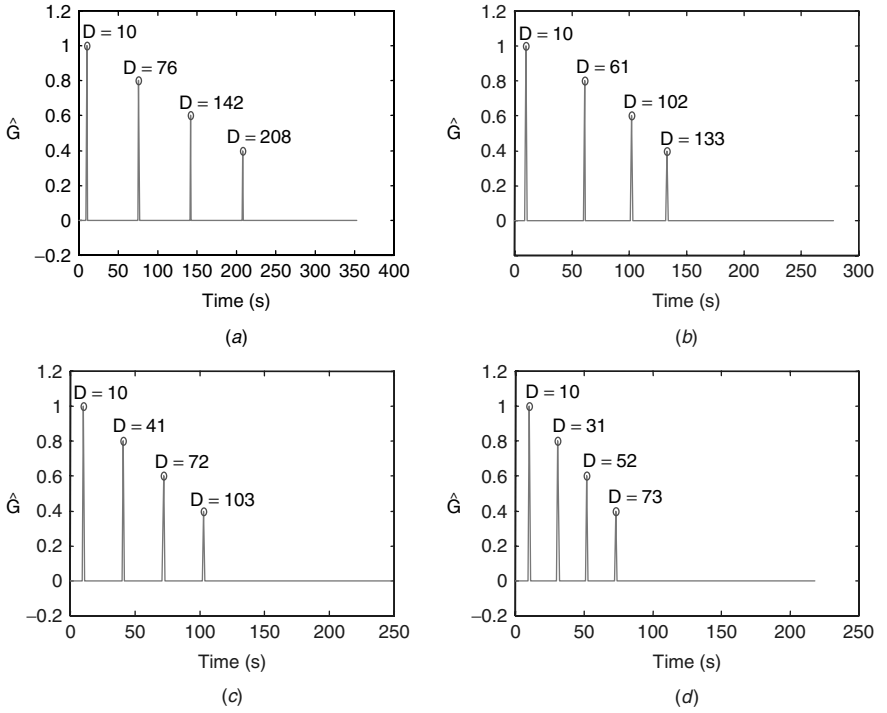



**Figure 4.10.** Optimal time delay estimation results for simulation: (*a*) Estimated impulse response for Case 1. (*b*) Estimated impulse response for Case 2. (*c*) Estimated impulse response for Case 3. (*d*) Estimated impulse response for Case 4.

concerned about inspection of its optics for pit damage, were generated using a sophisticated three-dimensional, finite-difference time-domain propagator [16]. The objective is NDE of the optics for flaw damage (pits) caused by firing the laser a large number of times. The synthesized data are shown in Figure 4.11*a* along with the resulting estimated response or "fit" using the optimal estimator of Eq. (4.86). Here the order of the *FIR* processor was selected to be 2500 weights for the 2750 simulated sample data set. Clearly, the *signal error test* (see Section 4.6) demonstrates that the estimated response overlays the data quite well. In Figure 4.11*b*, the estimated impulse response is shown along with its corresponding envelope [18]. Here we note in a physical application that the ideal delayed/attenuated impulse train corresponding to the reflections from the flaws is distorted by the more realistic, highly scattering medium of the synthesized optic. However, the peaks corresponding to the delays indicating the flaw locations are clearly discernable. A practical approach to delay (flaw detection) estimation is to use the squared impulse response, select a reasonable threshold and perform a peak detection. The results of this operation along with the corresponding envelope are shown in Figure 4.11*c*, where we observe the effectiveness of this approach. The estimated delays corresponded reasonably well to the known flaw locations estimated from

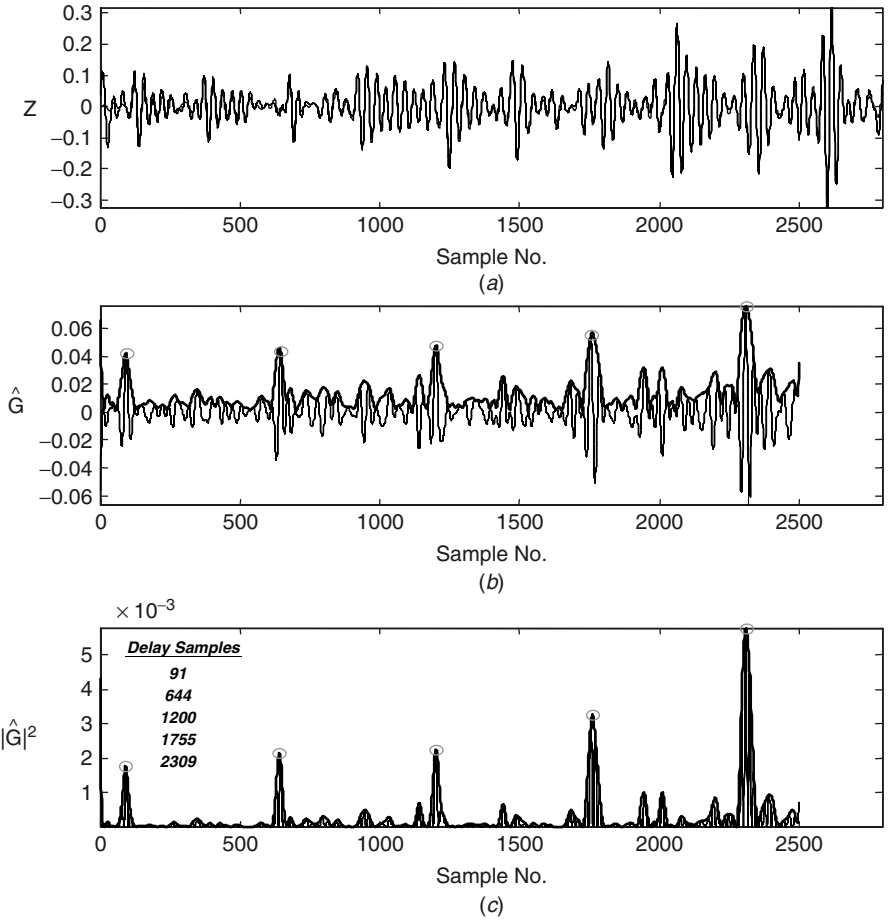



**Figure 4.11.** Optimal time delay estimation results for NDE application: (*a*) Signal error test of synthesized and estimated NDE data. (*b*) Estimated impulse response (time delays) and envelope. (*c*) Squared, thresholded, peak detected impulse response for time-delay estimates.

the delay and known velocity of the optic. Upon application to the real-world application, the processor performed quite well and was part of a sophisticated, real-time flaw detection system (see [16] for details).

We summarize the all-zero time-delay estimation approach as follows:

Criterion: $J = E\{\epsilon^2(t)\}$

Models:

Measurement: $z(t) = s(t) + n(t)$

Signal: $s(t) = g(r; t) * p(t)$

Noise: $R_{nn}$

Algorithm:

$$\hat{\mathbf{g}} = \mathbf{R}_{pp}^{-1}\mathbf{r}_{zp}$$

$$\hat{P}(k) = \hat{\mathbf{g}}^2(t)\Big|_{\tau_k = t_k} \quad \text{for peak}$$

Quality: $R_{\epsilon\epsilon}(N_g)$

This completes the section on Levinson all-zero filters. Next we consider a related design—the lattice all-pole processor.

## 4.4 LATTICE *MBP*

Since its roots were established in [1], the lattice filter and lattice techniques have evolved to a high degree of current popularity for many applications. The lattice method combines the orthogonal projection ideas in *both* time and parameter spaces ([19], [20]). Lattice filters can be developed directly from the model or indirectly from the Levinson recursion [10], we choose the former, because it follows the development of our previous model-based processors.

As before, we are interested in developing an estimate of the signal $s(t)$ based on past measurements and/or past excitations, that is,

$$\hat{s}(t) = \hat{s}(t|Y_{past} \text{ or } X_{past})$$

Again, taking the model-based approach using the parametric lattice model, we know from the results of Chapter 2 that given the lattice parameters we can construct all-pole or all-zero signal estimators. With this in mind let us now construct an estimator for the lattice model.

The *lattice recursion* for the $i$th section at time $t$ is given by

$$\begin{aligned} e_f(t,i) &= e_f(t,i-1) - k_i e_b(t-1,i-1) \\ e_b(t,i) &= e_b(t-1,i-1) - k_i e_f(t,i-1) \end{aligned} \tag{4.87}$$

where

$e_f(t,i) =$ *forward* prediction error of the $i$th section at time $t$

$e_b(t,i) =$ *backward* prediction error of the $i$th section at time $t$

$k_i$ is the reflection coefficient of the $i$th section

The model-based (parametric) problem is characterized by minimizing the joint error function in terms of the lattice parameters as

$$\min_{\mathbf{k}} \mathcal{J}(t) = E\{e_f^2(t) + e_b^2(t)\} \tag{4.88}$$

Performing the indicated minimization of Eq. (4.88) as before, we have the orthogonality condition

$$\frac{\partial J}{\partial k_i} = \frac{\partial}{\partial k_i} E\{e_f^2(t,i) + e_b^2(t,i)\} = 2E\left\{e_f(t,i)\frac{\partial e_f(t,i)}{\partial k_i} + e_b(t,i)\frac{\partial e_b(t,i)}{\partial k_i}\right\} = 0 \tag{4.89}$$

The error gradients can be calculated directly from the model of Eq. (4.87) as

$$\frac{\partial e_f}{\partial k_i} = -e_b(t-1, i-1)$$

Similarly we have

$$\frac{\partial e_b}{\partial k_i} = -e_f(t, i-1)$$

Substituting these gradients into Eq. (4.88) gives

$$\frac{\partial J}{\partial k_i} = -2E\{e_f(t,i)e_b(t-1,i-1) + e_b(t,i)e_f(t,i-1)\} = 0$$

Using the model recursion of Eq. (4.87), we obtain

$$\frac{\partial J}{\partial k_i} = -2E\{e_f(t,i-1)e_b(t-1,i-1) - k_i(e_f^2(t,i-1) + e_b^2(t-1,i-1))\} = 0 \tag{4.90}$$

Solving this equation for $k_i$ gives[5]

$$k_i = \frac{2E\{e_f(t,i-1)e_b(t-1,i-1)\}}{E\{e_f^2(t,i-1) + e_b^2(t-1,i-1)\}} \tag{4.91}$$

The variance of the lattice recursion can be calculated directly from the recursion, since

$$\begin{aligned} R_{ee}(i) &= E\{e_f^2(t,i)\} = E\{(e_f(t,i-1) - k_i e_b(t-1,i-1))^2\} \\ &= E\{e_f^2(t,i-1)\} - 2k_i E\{e_f(t,i-1)e_b(t-1,i-1)\} \\ &\quad + k_i^2 E\{e_b^2(t-1,i-1)\} \end{aligned} \tag{4.92}$$

Solving Eq. (4.91) for the numerator term and substituting, we obtain

$$\begin{aligned} R_{ee}(i) = E\{e_f^2(t,i-1)\} - k_i^2 \left(E\{e_f^2(t,i-1)\} + E\{e_b^2(t-1,i-1)\}\right) \\ + k_i^2 E\{e_b^2(t-1,i-1)\} \end{aligned}$$

[5]The reflection coefficient evolves from the Schur-Cohn stability technique and is used to test stability. If the reflection coefficient lies between $\pm 1$ the system is *stable*; that is, $|k_i| \leq 1$ for stability.

or

$$R_{ee}(i) = (1 - k_i^2) R_{ee}(i-1) \tag{4.93}$$

where $R_{ee}(0) = R_{yy}(0) = \text{var}(y(t))$. Expanding over an $M$-stage lattice, we see that

$$R_{ee} = R_{ee}(M) = \prod_{i=1}^{M} (1 - k_i^2) R_{ee}(0) \tag{4.94}$$

If we assume the data are ergodic, then Burg ([12], [21]) proposed replacing the ensemble averages of Eq. (4.91) with time averages using only the unwindowed data record, that is,

$$k_i = \frac{2 \sum_{t=N_a}^{N-1} e_f(t, i-1) e_b(t-1, i-1)}{\sum_{t=N_a}^{N-1} e_f^2(t, i-1) + e_b^2(t-1, i-1)} \qquad \text{for } i = 1, \ldots, N_a \tag{4.95}$$

Once the reflection coefficients are estimated from the data using Eqs. (4.87) and (4.95) for each stage, the prediction error variance is obtained from Eq. (4.94).

Note that the Burg method is a block (time) or batch model-based (parameter) technique which is *recursive in order*, just as the Levinson all-pole filters of the previous sections. We summarize the technique in Table 4.3. It should also be noted that when implementing this block estimator, the reflection coefficient is estimated and *all* of the data is processed through the corresponding lattice section. This data is then used to estimate the next reflection coefficient and so on (see [10] for more details).

We summarize the lattice *MBP* as follows:

Criterion: $\mathcal{J} = E\{e_f^2(t) + e_b^2(t)\}$

Models:

Signal:

$$s(t) = \underline{a}'(t) \underline{Y}(t)$$

$$e_f(t, i) = e_f(t, i-1) - k(t, i) e_b(t-1, i-1)$$

$$e_b(t, i) = e_b(t-1, i-1) - k(t, i) e_f(t, i-1)$$

$$\underline{k}(t) \longrightarrow \underline{a}(t)$$

Measurement: $y(t) = s(t) + n(t)$

Noise: $n(t)$ random

Initial conditions: $\underline{a}(0), \triangle(0, i)$

**Table 4.3. Lattice *MBP* Algorithm**

Prediction error estimation

$$e_f(t,i) = e_f(t,i-1) - k_i e_b(t-1,i-1) \qquad \text{for } i = 1,\ldots,N_a$$
$$e_b(t,i) = e_b(t-1,i-1) - k_i e_f(t,i-1)$$

Reflection coefficient estimation

$$k_i = \frac{2\sum_{t=N_a}^{N-1} e_f(t,i-1)e_b(t-1,i-1)}{\sum_{t=N_a}^{N-1} e_f^2(t,i-1) + e_b^2(t-1,i-1)}$$

Parameter calculation (Table 4.1)

$$\beta(i,i) = k_i$$
$$\beta(j,i) = \beta(j,i-1) - k_i\beta(i-j,i-1), \qquad j = 1,\ldots,i-1$$

Noise estimation

$$R_{ee} = \prod_{i=1}^{N_a}(1-k_i^2)R_{yy}(0)$$

Signal estimation

$$\hat{s}(t|t-1) = -\sum_{i=1}^{N_a}\beta_i y(t-1) \qquad \text{(All-pole)}$$

or

$$\hat{s}(t|t-1) = \sum_{i=1}^{N_a}\beta_i x(t-i) \qquad \text{(All-zero)}$$

Algorithm:

$$k(t+1,i) = \frac{2\sum_{t=N_1}^{N-1} e_f(t,i-1)e_b(t-1,i-1)}{\sum_{t=N_1}^{N-1} e_f^2(t,i-1)+e_b^2(t-1,i-1)} \qquad \text{for} \quad i = 1,\ldots,N_a$$

Quality: $R_{ee}$

It should also be noted that the lattice model actually evolves physically as a wave propagating through a layered medium. That is, if we interpret $e_f$ and $e_b$ as forward and backward waves (see Figure 4.12), then it is possible to derive the lattice recursions directly from the wave equation ([1], [14]). This model has been used quite successfully to estimate acoustic, seismic, and electromagnetic signals (waves) in layered media.

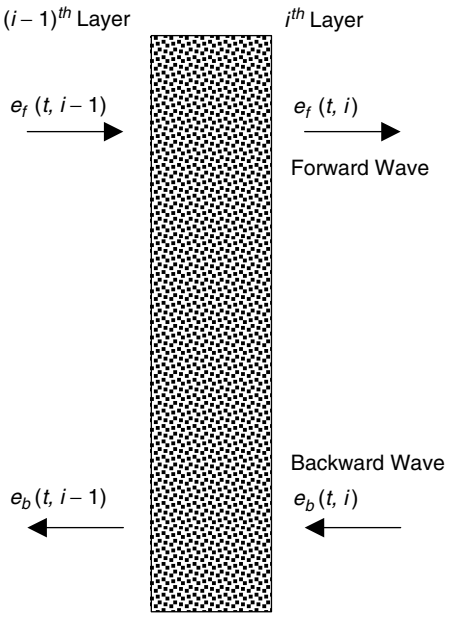



**Figure 4.12.** Wave propagation in layered medium—the lattice model.

Consider the following example to see how the lattice technique can be used to estimate a signal.

***Example 4.8*** Consider the deterministic version of the previous example. We would like to estimate the signal from a system with data set given by $Y(t) = \{1, \alpha\}$. The recursion begins with

$$k_1 = \frac{\sum_{t=1}^{2} e_f(t,0)e_b(t-1,0)}{\sum_{t=1}^{2} e_f^2(t,0) + e_b^2(t-1,0)}$$

$$= \frac{e_f(1,0)e_b(0,0) + e_f(2,0)e_b(1,0)}{e_f^2(1,0) + e_b^2(0,0) + e_f^2(2,0) + e_b^2(1,0)}$$

$$= \frac{\alpha(1)}{1+\alpha^2+1} = \frac{\alpha}{\alpha^2+2}, \qquad i = 0$$

$$e_f(t,1) = e_f(t,0) - \left(\frac{\alpha}{\alpha^2+2}\right) e_b(t-1,0)$$

$$e_b(t,1) = e_b(t-1,0) - \left(\frac{\alpha}{\alpha^2+2}\right) e_f(t,0)$$

or

$$\{e_f(1,1), e_f(2,1)\} = \left\{1, \frac{\alpha^2-\alpha+2}{\alpha^2+2}\right\}$$

$$\{e_b(1,1), e_b(2,1)\} = \left\{-\frac{\alpha}{\alpha^2+2}, \frac{\alpha^2-\alpha+2}{\alpha^2+2}\right\}$$

$$k_2 = \frac{e_f(1,1)e_b(0,1) + e_f(2,1)e_b(1,1)}{e_f^2(1,1) + d_b^2(0,1) + e_f^2(2,1) + e_b^2(1,1)}$$

$$= \frac{-\alpha(\alpha^2-\alpha+2)}{(\alpha^2+2)^2 + (\alpha^2-\alpha+2)^2 + \alpha^2}$$

For the all-pole model, we have

$$\hat{s}(t|t-1) = -\left(\frac{\alpha}{\alpha^2+2}\right) y(t-1)$$

and a second-order predictor can be developed using $\{k_1, k_2\}$ and the Levinson recursion. This completes the example.

Next we consider applying the algorithm to a simulated set of data to see how well it can perform.

***Example 4.9*** Consider the simulated data set of Example 4.2, where the data is simulated from an *ARMAX*(2, 1, 1, 1) model with $e \sim N(0, 0.01)$. Using *MATLAB* [11] to simulate this process and the algorithm of Table 4.3 we see the results in Figure 4.13. After selecting a third-order model to represent the data, we obtain the following estimated parameters:

| Parameter | True | Estimated | Reflection |
|---|---|---|---|
| $a_1$ | −1.5 | −1.47 | −0.89 |
| $a_2$ | 0.7 | 0.73 | 0.59 |
| $a_3$ | — | −0.10 | −0.10 |
| $b_0$ | 1.0 | 0.16 | |
| $b_1$ | 0.5 | — | |
| $c_1$ | 0.3 | — | |
| $d_1$ | 0.5 | — | |

The filtered output is shown in Figure 4.13*b*. Here we see that the lattice method appears to perform better than the Levinson-Durbin approach as indicated by the estimated impulse response and signal.

This completes the section on lattice filters. Next we consider the *ARMAX* processor.

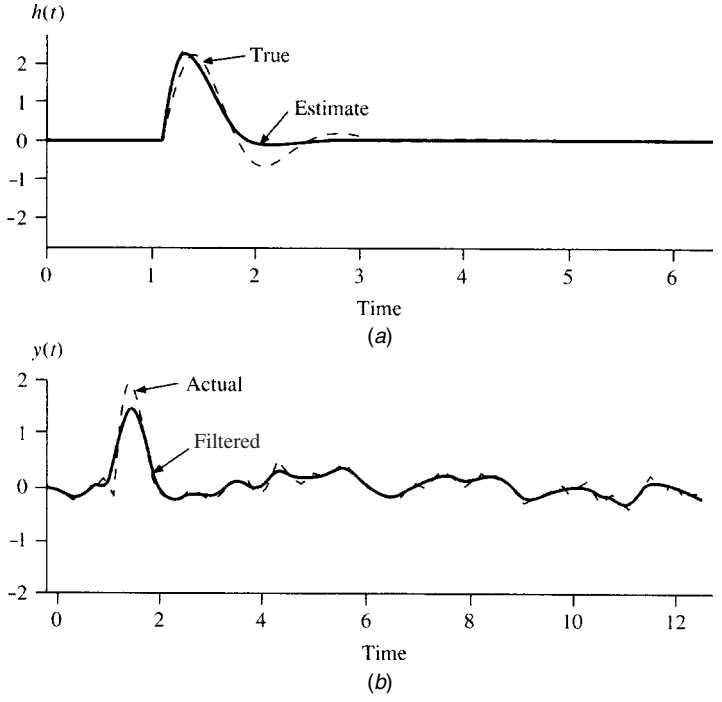



**Figure 4.13.** Lattice all-pole filter design for *ARMAX*(2, 1, 1, 1) simulation. (*a*) True and estimated impulse response. (*b*) Actual and filtered measurement (signal).

## 4.5 *ARMAX* (POLE-ZERO) *MBP*

In this section we develop the prediction error filters that evolved from controls/estimation ([6], [7]). In contrast to the previous model-based (parametric) estimators, the prediction error filter is *recursive in time* rather than *recursive in order*. Therefore it is not a block (of data) processor. Prediction error filters are used for adaptive control and have also been applied successfully in adaptive signal processing problems as well ([7], [8]).

As the parameter estimator for this case, was developed, many different parameter estimation schemes evolved. We choose the recursive prediction error method (*RPEM*) developed by Ljung [7], since it can be shown that most recursive parameter estimators are special cases of this approach.

By the prediction error approach, we can obtain an estimate of the signal based on past measurement data, $\{y(t)\}$, and include a known "exogenous" excitation (input) defined by the $\{u(t)\}$. The estimate is therefore

$$\hat{s}(t) = \hat{s}(t|Y_{Past})$$

Taking the model-based (parametric) approach, we model the signal by the relation

$$s(t) = -\sum_{i=1}^{N_a} a_i y(t-i) + \sum_{i=0}^{N_b} b_i u(t-i) \tag{4.96}$$

Therefore the measurement is given by

$$y(t) = s(t) + n(t) \tag{4.97}$$

If we choose to represent the noise as correlated rather than white, that is,

$$n(t) = \sum_{i=1}^{N_c} c_i e(t-i) + e(t) \tag{4.98}$$

then substituting Eqs. (4.96) and (4.98) into Eq. (4.97), we obtain

$$y(t) = -\sum_{i=1}^{N_a} a_i y(t-i) + \sum_{i=0}^{N_b} b_i u(t-i) + \sum_{i=0}^{N_c} c_i e(t-i), \qquad c_0 = 1 \tag{4.99}$$

More succinctly in backward shift operator notation ($q^{-1}$) we have

$$A(q^{-1})y(t) = B(q^{-1})u(t) + C(q^{-1})e(t) \tag{4.100}$$

Thus the $ARMAX(N_a, N_b, N_c)$ evolves as the underlying model in this model-based processor.

In developing the *RPEM* based on the *ARMAX* model, we first note that Eq. (4.99) can be rewritten in vector form as

$$y(t) = \phi'(t)\boldsymbol{\Theta} + e(t) \tag{4.101}$$

where

$$\phi' := [-y(t-1)\cdots - y(t-N_a) \mid u(t)\cdots u(t-N_b) \mid e(t-1)\cdots e(t-N_c)]$$

and

$$\boldsymbol{\Theta}' := [a_1 \cdots a_{N_a} \mid b_0 \cdots b_{N_b} \mid c_1 \cdots c_{N_c}]$$

We define the *prediction error criterion*[6] as

$$\min_{\Theta} \mathcal{J}_t(\Theta) = \frac{1}{2}\sum_{k=1}^{t} e^2(k, \Theta) \tag{4.102}$$

[6]We explicitly show the dependence of the prediction error on the unknown parameter vector $\Theta$.

where the *prediction error* is defined by

$$e(t, \Theta) := y(t) - \hat{s}(t|t-1) \tag{4.103}$$

and

$$\hat{s}(t|t-1) = (1 - \hat{A}(q^{-1}))y(t) + \hat{B}(q^{-1})u(t) \tag{4.104}$$

The model-based *ARMAX* parameter estimator evolves by performing the indicated minimization in Eq. (4.102) after performing a Taylor series expansion about $\Theta = \hat{\Theta}(t-1)$, that is,

$$\mathcal{J}_t(\Theta) = \mathcal{J}_t(\hat{\Theta}) + \nabla'_\Theta \mathcal{J}_t(\hat{\Theta})(\Theta - \hat{\Theta}) + \tfrac{1}{2}(\Theta - \hat{\Theta})' \nabla^2_\Theta \mathcal{J}_t(\hat{\Theta})(\Theta - \hat{\Theta}) + \text{H.O.T.} \tag{4.105}$$

where

$$\Theta = N_\Theta \times 1 \text{ parameter vector}$$
$$\nabla^2_\Theta = N_\Theta \times N_\Theta \text{ Hessian matrix}$$
$$\nabla_\Theta = N_\Theta \times 1 \text{ gradient vector}$$

If we perform the minimization, then differentiating, we obtain

$$\begin{aligned}\nabla_\Theta \mathcal{J}_t(\Theta) = \nabla_\Theta \mathcal{J}_t(\hat{\Theta}) &+ \nabla_\Theta \left[\nabla'_\Theta \mathcal{J}_t(\hat{\Theta})(\Theta - \hat{\Theta})\right] \\ &+ \nabla_\Theta \left[\tfrac{1}{2}(\Theta - \hat{\Theta})' \nabla^2_\Theta \mathcal{J}_t(\hat{\Theta})(\Theta - \hat{\Theta})\right] + \text{H.O.T.}\end{aligned}$$

Using the chain rule of vector calculus (see Eq. 3.12), performing the indicated operations, and neglecting the H.O.T., we have

$$\nabla_\Theta \mathcal{J}_t(\Theta) \approx \nabla_\Theta \mathcal{J}_t(\hat{\Theta}) + \nabla^2_\Theta \mathcal{J}_t(\hat{\Theta})(\Theta - \hat{\Theta}(t-1))\Big|_{\Theta = \hat{\Theta}(t)} = 0 \tag{4.106}$$

Solving for $\hat{\Theta}(t)$, we obtain the *Gauss-Newton* parameter estimator

$$\hat{\Theta}(t) = \hat{\Theta}(t-1) - \left[\nabla^2_\Theta \mathcal{J}_t(\hat{\Theta})\right]^{-1} \nabla_\Theta \mathcal{J}_t(\hat{\Theta}) \tag{4.107}$$

Thus we must develop the gradient and Hessian to construct the *RPEM* estimator. The gradient is given by

$$\nabla_\Theta \mathcal{J}_t(\Theta) = \frac{1}{2} \nabla_\Theta \sum_{k=1}^{t} e^2(k, \Theta) = \sum_{k=1}^{t} e(k, \Theta) \nabla_\Theta e(k, \Theta) \tag{4.108}$$

We define the negative gradient innovation vector as

$$\psi(t, \Theta) := -\nabla_\Theta e(t, \Theta) \tag{4.109}$$

Therefore substituting into Eq. (4.108), we obtain

$$\nabla_\Theta \mathcal{J}_t(\Theta) = -\sum_{k=1}^{t} \psi(k,\Theta)e(k,\Theta) \tag{4.110}$$

Since we are interested in a *recursive-in-time* technique, we can factor out the $t$th term

$$\nabla_\Theta \mathcal{J}_t(\Theta) = -\sum_{k=1}^{t-1} \psi(k,\Theta)e(k,\Theta) - \psi(t,\Theta)e(t,\Theta)$$

Then we can recognize the first term to be the gradient of the previous time step. Therefore we have the *gradient recursion*

$$\nabla_\Theta \mathcal{J}_t(\Theta) = \nabla_\Theta \mathcal{J}_{t-1}(\Theta) - \psi(t,\Theta)e(t,\Theta) \tag{4.111}$$

The Hessian can also be determined in a similar manner using this recursion, we have

$$\nabla^2_\Theta \mathcal{J}_t(\Theta) = \nabla^2_\Theta \mathcal{J}_{t-1}(\Theta) - \psi(t,\Theta)\nabla'_\Theta e(t,\Theta) - e(t,\Theta)\nabla_\Theta \psi'(t,\Theta) \tag{4.112}$$

The expressions for the gradient and Hessian can be simplified even further, if we assume that when the algorithm converges,

- $\hat{\Theta} \approx \Theta_{\text{TRUE}}$,
- $\nabla^2_\Theta \mathcal{J}_t(\hat{\Theta}(t)) \approx \nabla^2_\Theta \mathcal{J}_t(\hat{\Theta}(t-1))$,
- $\nabla_\Theta \mathcal{J}_{t-1}(\hat{\Theta}(t-1)) \approx 0$, and
- $e(t,\hat{\Theta})\nabla^2_\Theta e(t,\hat{\Theta}(t-1)) = 0$.

Under these convergence assumptions, when $\hat{\Theta} \to \Theta_{\text{TRUE}}$ or equivalently the estimate is "close" to the true value, then the gradient recursion simplifies to

$$\nabla_\Theta \mathcal{J}_t(\hat{\Theta}(t-1)) = -\psi(t,\hat{\Theta}(t-1))e(t,\hat{\Theta}(t-1)) \tag{4.113}$$

Therefore the Hessian recursion also simplifies to

$$\nabla^2_\Theta \mathcal{J}_t(\hat{\Theta}(t-1)) = \nabla^2_\Theta \mathcal{J}_{t-1}(\hat{\Theta}(t-1)) + \psi(t,\hat{\Theta}(t-1))\psi'(t,\hat{\Theta}(t-1)) \tag{4.114}$$

If we define the Hessian matrix as

$$R(t) = \nabla^2_\Theta \mathcal{J}_t(\Theta)$$

then Eq. (4.113) becomes simply

$$R(t) = R(t-1) + \psi(t,\hat{\Theta}(t-1))\psi'(t,\hat{\Theta}(t-1)) \tag{4.115}$$

and the Gauss-Newton parameter estimator becomes

$$\hat{\Theta}(t) = \hat{\Theta}(t-1) + R^{-1}(t)\psi(t,\hat{\Theta})e(t,\hat{\Theta}) \tag{4.116}$$

These equations make up the *RPEM* algorithm, all that is required is to determine $e(t,\Theta)$ and $\psi(t,\Theta)$. Using the *ARMAX* model and differentiating, we have[7]

$$C(q^{-1})\nabla_\Theta e(t) = \nabla_\Theta[A(q^{-1})y(t) - B(q^{-1})u(t)] \tag{4.117}$$

$$C(q^{-1})\begin{bmatrix} \dfrac{\partial e(t)}{\partial a_i} \\ --- \\ \dfrac{\partial e(t)}{\partial b_i} \\ --- \\ \dfrac{\partial e(t)}{\partial c_i} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial}{\partial a_i} \\ --- \\ \dfrac{\partial}{\partial b_i} \\ --- \\ \dfrac{\partial}{\partial c_i} \end{bmatrix} [A(q^{-1})y(t) - B(q^{-1})u(t)] = \begin{bmatrix} y(t-i) \\ --- \\ -u(t-i) \\ --- \\ -e(t-i) \end{bmatrix}$$

or succinctly

$$C(q^{-1})\psi(t) = \phi(t) \tag{4.118}$$

Equation (4.118) is equivalent to inverse filtering to obtain $\psi(t)$, that is,

$$C(q^{-1})\psi(t) = \begin{bmatrix} C(q^{-1})y_f(t,i) \\ ----- \\ -C(q^{-1})u_f(t,i) \\ ----- \\ C(q^{-1})e_f(t,i) \end{bmatrix} = \begin{bmatrix} y(t-i) \\ --- \\ u(t-i) \\ --- \\ e(t-i) \end{bmatrix} \tag{4.119}$$

where $y_f$, $u_f$, and $e_f$ are the outputs of the specified filtering operation. For instance,

$$y_f(t,i) = y(t-i) - \left(C(q^{-1}) - 1\right) y_f(t,i) \quad \text{for } i = 1,\ldots,N_a \tag{4.120}$$

Similar relations are given for $u_f$ and $e_f$ above. Therefore the gradient is

$$\psi'(t) = [y_f(t,i) \mid u_f(t,i) \mid e_f(t,i)]'$$

after the "filtering" operation of Eq. (4.120) has been performed.

Finally the prediction error is calculated using the most current available parameter estimate, that is,

$$e(t) = y(t) - \phi'(t)\hat{\Theta}(t-1) \tag{4.121}$$

The *RPEM* algorithm can be implemented at this point. However, it requires the inversion of the $N_\Theta \times N_\Theta$ Hessian matrix at each time step. This calculation can

[7]We drop the $\Theta$-dependent notation, $e(t,\Theta) = e(t)$, $\psi(t,\Theta) = \psi(t)$.

**Table 4.4. Recursive Prediction Error Method (*RPEM*)**

| | |
|---|---|
| Prediction error | |
| $e(t) = y(t) - \hat{y}(t \mid t-1) = y(t) - \phi'(t)\hat{\Theta}(t-1)$ | (Prediction error) |
| $R_{ee}(t) = \psi'(t)P(t-1)\psi(t) + \lambda(t)$ | (Prediction error variance) |
| Update | |
| $\Theta(t) = \Theta(t-1) + \mu K(t)e(t)$ for $\mu$ the step-size | (Parameter update) |
| $r(t) = y(t) - \overline{\phi}'(t)\hat{\Theta}(t)$ | (Residual update) |
| $P(t) = (I - K(t)\psi'(t))P(t-1)\lambda(t)$ | (Covariance update) |
| $\lambda(t) = \lambda_0\lambda(t-1) + (1-\lambda_0)$ | (Forgetting factor update) |
| Filtering | |
| $y_f(t,i) = y(t-i) - [\hat{C}(q^{-1}) - 1]y_f(t,i)$ | (Measurement filter) |
| $u_f(t,i) = u(t-i) - [\hat{C}(q^{-1}) - 1]u_f(t,i)$ | (Input filter) |
| $r_f(t,i) = r(t-i) - [\hat{C}(q^{-1}) - 1]r_f(t,i)$ | (Residual filter) |
| Gain | |
| $K(t) = P(t-1)\psi(t)R_e^{-1}(t)$ | (Gain or weight) |
| $\hat{\phi}'(t) = [-y(t-1)\cdots - y(t-N_a) \mid u(t)\cdots u(t-N_b) \mid \times r(t)\cdots r(t-N_c)]$ | (Phi-update) |
| $\psi'(t) = [-y_f(t-1)\cdots - y_f(t-N_a) \mid u_f(t)\cdots u_f(t-N_b) \mid \times r_f(t)\cdots r_f(t-N_c)]$ | (Gradient update) |

be reduced by applying the matrix inversion lemma[8] to $R^{-1}(t)$; that is, taking the inverse of both sides of Eq. (4.115) and defining, $P(t) := R^{-1}(t)$, we obtain

$$P(t) = [P^{-1}(t-1) + \psi(t)\psi'(t)]^{-1}$$

Now applying the inversion lemma with $A = P^{-1},\ B = \psi,\ C = 1,\ D = \psi'$, we obtain

$$P(t) = P(t-1) - \frac{P(t-1)\psi(t)\psi'(t)P(t-1)}{1 + \psi'(t)P(t-1)\psi(t)} \tag{4.122}$$

which completes the algorithm. We summarize the *RPEM* in Table 4.4. From the table we note that the algorithm is implemented with the *residuals* (as well as $\phi$)

$$r(t) := y(t) - \phi'(t)\hat{\Theta}(t) \tag{4.123}$$

[8]The matrix inversion lemma is given by $[A + BCD]^{-1} = A^{-1} - A^{-1}B[DA^{-1}B + C]^{-1}DA^{-1}$ [9] see page 151.

as soon as the new parameter update is available. It is also important to note that $A(z^{-1})$ must have roots inside the unit circle for convergence and therefore stability test is usually incorporated in the algorithm. We refer the interested reader to Ljung [7] for more details.

We summarize the *RPEM* within the model-based framework as follows:

Criterion: $\mathcal{J}_t(\Theta) = \frac{1}{2}\sum_{k=1}^{t} e^2(k, \Theta)$
Models:
  Measurement: $y(t) = s(t) + n(t)$
  Signal: $s(t) = (1 - \hat{A}(q^{-1}))y(t) + \hat{B}(q^{-1})u(t)$
  Noise: $R_{ee}$
Algorithm: $\hat{\Theta}(t) = \hat{\Theta}(t-1) + \mu K(t)e(t)$
Quality: $\mathcal{J}_t(\hat{\Theta})$

It should also be noted that most other recursive (in-time) parameter estimation techniques are special cases of the *RPEM*. For instance, the extended least-squares (*ELS*) technique is implemented as a special case of *RPEM* with

$$\psi(t) = \phi(t)$$

and the filtering operation ($y_f$, $u_f$, $e_f$) eliminated (see Table 4.6). Consider the following example of applying the *RPEM* to a simulated problem.

***Example 4.10*** Again using the *ARMAX*(2, 1, 1, 1) simulated data (256 points) of Example 4.2, we applied the *ELS* algorithm in *MATLAB* [11]. The prediction error filter was selected as *ARMAX*(2, 1, 1, 1) using the *FPE* criterion (*FPE* = 0.0125). The estimated parameters are

| Parameter | True | Estimated |
|---|---|---|
| $a_1$ | −1.5 | −1.49 |
| $a_2$ | 0.7 | 0.7 |
| $b_0$ | 1.0 | 1.95 |
| $b_1$ | 0.5 | −0.22 |
| $c_1$ | 0.3 | −0.12 |
| $d_1$ | 0.5 | — |

The results of this run are shown in Figure 4.14. Here we see the true and estimated impulse response. The *RPEM* estimator gives better results than the previous recursive filters because of the zeros available in both $B$ and $C$ polynomials. The "filtered" measurement or estimated signal is also shown and it again confirms a reasonable estimate. Both *ELS* and *RPEM* results are shown in the figure.

This completes the section on *RPEM* methods, next we consider the problem of order estimation.

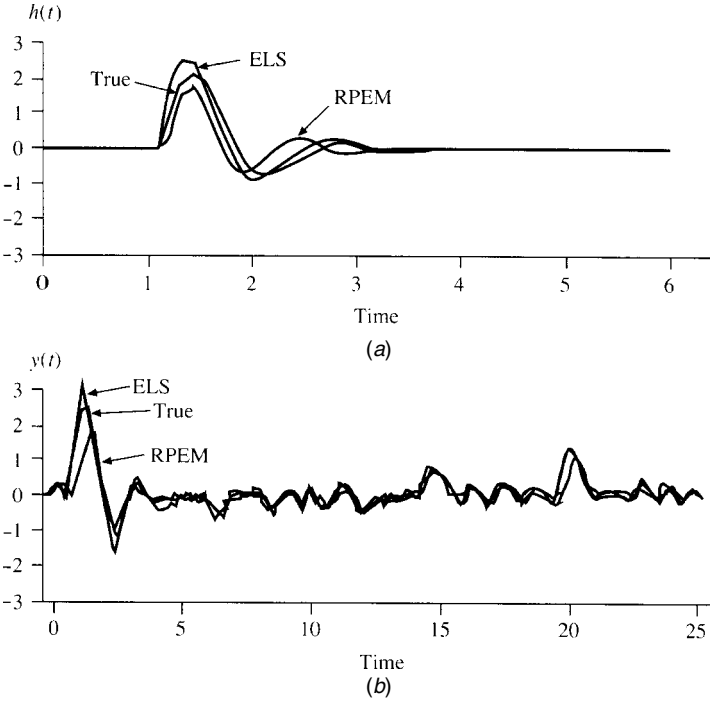



**Figure 4.14.** *RPEM* pole-zero filter design for *ARMAX*(2, 1, 1, 1) simulation. (*a*) True and estimated impulse response. (*b*) Actual and filtered measurement (signal).

## 4.6 ORDER ESTIMATION FOR *MBP*

An important and related problem in any of these model-based (parametric) estimation schemes is that of estimating the order of the underlying system model. In fact most schemes require the system order to be "known" prior to estimating the model parameters. Therefore we must first determine the correct number of parameters to estimate from the data directly. This is a real requirement as we have seen in the spectral estimation as well as parametric signal-processing problems. Order estimation has evolved specifically from the identification problem [6] where the first step is to estimate or guess the system order and then attempt to validate prediction error or innovations through whiteness testing of the estimated model. Much effort has been devoted to this problem ([6], [7]) with the most significant results evolving from the information theoretical point of view ([22]–[29]). Here we start with the basic ideas and quickly converge to the *order estimation* problem for model-based processors following the Akaike information theoretic approach ([22], [23]). We discuss the idea of order estimation, that is, the estimation of the order of an all-pole ($AR(N_a)$), all-zero ($MA(N_b)$), pole-zero ($ARMA(N_a,N_b)$) models as well as state-space–multichannel model-based systems, $\Sigma$.

Perhaps the simplest form of order estimation results from the technique called *repeated least squares*. Here the least squares parameter estimator is used to fit various assumed model orders and the estimated mean-squared error calculated. The model order selected is the one with minimal error, that is,

$$\mathcal{J}(N_i) \geq \mathcal{J}_{\min}(N_{\text{true}}) \qquad \forall i$$

where $\mathcal{J}(N_i) := E\{e^2(t, N_i)\}$, $N_i$ is the assumed order, while $N_{\text{true}}$ is the true model order. The problem with this technique is that the mean-squared error monotonically decreases as model order increases and therefore it is difficult to determine the true order, since there is *no* penalty for choosing a large number of free parameters (order). Akaike [22] shows that this can be improved when criteria are developed from the information theoretical viewpoint. This is the approach we pursue.

Before we consider this case, let us define some concepts about probabilistic information. These concepts are applied extensively in communications problems and will prove useful in designing parametric signal processors. The *information* (self) contained in the occurrence of the event $\omega_i$ such that $X(\omega_i) = x_i$ is

$$\mathcal{I}(x_i) = -\log_b \Pr\left(X(\omega_i) = x_i\right) = -\log_b \Pr_X(x_i) = -\log_b \Pr(x_i) \tag{4.124}$$

where $b$ is the base of the logarithm which results in different units for information measures (e.g., base $2 \rightarrow$ bits, while base $e \rightarrow$ implies nats, see Appendix A for details). The *entropy* or *average information* is defined by

$$\mathcal{H}(x_i) := E_X\{I(x_i)\} = \sum_{i=1}^{N} \mathcal{I}(x_i)\Pr(x_i) = -\sum_{i=1}^{N} \Pr(x_i)\log_b \Pr(x_i) \tag{4.125}$$

*Mutual information* is defined in terms of the information available in the occurrence of the event $Y(\omega_j) = y_j$ about the event $X(\omega_i) = x_i$ or

$$\mathcal{I}(x_i; y_j) = \log_b \frac{\Pr(x_i|y_j)}{\Pr(x_i)} = \log_b \Pr(x_i|y_j) - \log_b \Pr(x_i) \tag{4.126}$$

Now, using these concepts, we take the information theoretic approach to model order estimation following ([22], [23]). Since many models are expressed in terms of their probability distributions (e.g., Gauss-Markov model), quality or "goodness" can be evaluated by their similarity to the true underlying probability distribution generating the measured data. This is the fundamental concept embedded in this approach.

Suppose that $\Pr(x_i)$ is the *true* discrete probability distribution and that $\Pr(m_i)$ is the probability distribution of the model, then the *Kullback-Leibler Information*

(*KLI*) quantity of the true distribution relative to the model is defined by using

$$\mathcal{I}_{KL}\left(\Pr(x_i);\Pr(m_i)\right) := E_X\left\{\ln\frac{\Pr(x_i)}{\Pr(m_i)}\right\} = \sum_{i=1}^{N}\Pr(x_i)\ln\frac{\Pr(x_i)}{\Pr(m_i)}$$

$$= \sum_{i=1}^{N}\Pr(x_i)\ln\Pr(x_i) - \sum_{i=1}^{N}\Pr(x_i)\ln\Pr(m_i) \qquad (4.127)$$

here we chose $\log_b = \ln$. The *KLI* possesses some very interesting properties that we state without proof (see [23] for details) such as

1. $\mathcal{I}_{KL}\left(\Pr(x_i);\Pr(m_i)\right) \geq 0$
2. $\mathcal{I}_{KL}\left(\Pr(x_i);\Pr(m_i)\right) = 0 \Leftrightarrow \Pr(x_i) = \Pr(m_i)$ for all $i$
3. The negative of the *KLI* is the *entropy*, $\mathcal{H}_{KL}\left(\Pr(x_i);\Pr(m_i)\right)$

The second property implies that as the *model* approaches the *true* distribution, then the value of the *KLI* approaches *zero* (minimum). It also implies that if we have two models, $\Pr(m_1)$ and $\Pr(m_2)$, of the true distribution, then $\Pr(m_1)$ is "closer" to $\Pr(x_i)$ if $\mathcal{I}_{KL}\left(\Pr(x_i);\Pr(m_1)\right) < \mathcal{I}_{KL}\left(\Pr(x_i) < \Pr(m_2)\right)$ therefore choose $m_1$. Thus, investigating Eq. (4.127), we see that the first term is a constant specified by the true distribution; therefore, we only need to estimate the average value of the model relative to the true distribution, that is,

$$\mathcal{L}(m_i) := E_X\{\ln\Pr(m_i)\} = \sum_{i=1}^{N}\Pr(x_i)\ln\Pr(m_i) \qquad (4.128)$$

where $\mathcal{L}(m_i)$ is defined as the *average loglikelihood* of the random variable of value $\ln\Pr(m_i)$. Clearly, the *larger* the average loglikelihood, the *smaller* the *KLI* implying a *better* model.

The third property, *entropy*, is approximately equal to $1/N$ times the probability that the relative frequency distribution of $N$ measurements obtained from the assumed model equals the true distribution.

From the *law of large numbers*, it follows that [23]

$$\mathcal{L}(m) := E_Y\{\ln\Pr(m)\} \longrightarrow \frac{1}{K}\sum_{k=1}^{N}\ln\Pr\left(m(y_k)\right) \qquad \text{for } K \to \infty \qquad (4.129)$$

That is, if the event $\omega_i$ occurs at the $k$th-measurement, $y_k$, then $\Pr\left(m(y_k)\right) = \Pr\left(m_i\right)$ converges to the average loglikelihood as $K \to \infty$. Further, if $k_i$ is the number of times $\ln\Pr(m)$ takes on the value $\ln\Pr(m_i)$, then Eq. (4.129) becomes the loglikelihood of the model, that is,

$$\mathcal{L}(m_i) = \frac{1}{K}\sum_{i=1}^{N}k_i\ln\Pr\left(m_i\right) \qquad (4.130)$$

and Eqs. (4.129) and (4.130) are identical in the limit. Thus the conclusion from this discussion is that the *larger* $\mathcal{L}(m_i)$, the *better* the model and the loglikelihood is a biased estimator. With this background information defined, let us develop the *AIC*.

The basic concept of the *AIC* is that the average loglikelihood of the model, $\mathcal{L}(\cdot)$, where the average is with respect to the measurement data, is used for a goodness of fit metric as developed above in the general case. A more natural estimate seems to be the maximum loglikelihood; however, it is a *biased* estimator for this problem which tends to overestimate the statistic [23]. This tendency has been observed especially when the number of "free parameters" in the model, $N_\Theta$, $\Theta = \{\theta_1, \ldots, \theta_{N_\Theta}\}$, is large. Akaike [22] observed that the bias is approximately equal to the number of free parameters and developed an asymptotically unbiased estimator of $\mathcal{L}(\cdot)$ defined verbally by correcting for the bias as

$$AIC = -2 \times (\text{maximum loglikelihood of model})$$
$$+ 2 \times (\text{number of free parameters})$$

where historical reasons account for the 2 ([22], [23]). A model that minimizes the *AIC* is defined as the minimum *AIC* estimate or *MAICE*. This definition implies that if there are several models that have approximately the same *AIC*, then the one with the "fewest" free parameters should be selected. This choice is called *parsimony* in the identification literature [29].

From the model-based perspective, we restrict the distributions of truth and model to the same inherent structure (*AR*, *MA*, *ARMA*, state-space, etc.). We assume that the *true* model parameters are represented by $\Theta_{\text{true}}$, while those of the estimated model are $\Theta$. In terms of these parameters the corresponding distributions are respectively, $\Pr(y(t)|\Theta_{\text{true}})$ and $\Pr(y(t)|\Theta)$ with $y(t)$ the discrete measurement data generated by the true model distribution. Therefore, if we define the *AIC* in terms of loglikelihood and the $N_\Theta$-parameter set, $\Theta$, then

$$AIC(N_\theta) = -2 \times \mathcal{L}(\Theta) + 2 \times N_\theta \tag{4.131}$$

where $\mathcal{L}(\Theta)$ is the *average loglikelihood*.

***Example 4.11*** Let us develop the *AIC* for the autoregressive ($AR$) model of Chapter 2. The AR model can be expressed in terms of the backward shift operator $q^{-i}$ simply as

$$A(q^{-1})y(t) = e(t)$$

where $y, e$ are the output and prediction error (innovations) with $e \sim \mathcal{N}(0, R_{ee})$. The unknown model parameters are the polynomial coefficients, $\{a_i\}, i = 1, \ldots, N_a$ and variance $R_{ee}$, that is, $\Theta := [\{a_i\}, R_{ee}] \in \mathcal{R}^{N_a+1}$. Assuming independent measurements, it is straightforward to show that the loglikelihood function is given by

$$\mathcal{L}(\Theta) = \ln \Pr(Y_N|\Theta) = -\frac{N}{2}(\ln 2\pi R_{ee}) + \frac{1}{2R_{ee}} \sum_{t=1}^{N} e^2(t; \Theta)$$

Where we explicitly show the dependence on the unknown parameter $\Theta$ with $Y_N := \{y(0) \cdots y(N)\}$. Performing maximum likelihood estimation or equivalently minimizing the loglikelihood leads to the prediction error variance estimate

$$\hat{R}_{ee} = \frac{1}{N} \sum_{t=1}^{N} e^2(t; \hat{\Theta})$$

and therefore substituting into $\mathcal{L}(\Theta)$ we have

$$\mathcal{L}(\hat{\Theta}) = \frac{-N}{2}(\ln 2\pi \hat{R}_{ee}) - \frac{1}{2\hat{R}_{ee}}(N\hat{R}_{ee}) = \frac{-N}{2}(\ln 2\pi + \ln \hat{R}_{ee}) - \frac{N}{2}$$

from the definition of the *AIC* we obtain

$$\begin{aligned} AIC(N_a) = -2\mathcal{L}(\hat{\Theta}) + 2(N_a + 1) &= (N(\ln 2\pi + \ln \hat{R}_{ee}) + N) + 2(N_a + 1) \\ &= N(\ln 2\pi + \ln \hat{R}_{ee} + 1) + 2(N_a + 1) \end{aligned}$$

Next let us investigate the set of prediction error or innovation *MBP*. If the innovations are given by $\{e(t)\}$, then the loglikelihood function is specified by the following relations, since the $\{e(t)\}$ are independent. From Eq. (4.129) we have the average loglikelihood

$$\mathcal{L}(N, \Theta) = E_e\{\ln \Pr(e(t)|\Theta)\} = \frac{1}{N} \sum_{t=1}^{N} \ln \Pr(e(t)|\Theta) \tag{4.132}$$

which in the limit approaches the ensemble *mean loglikelihood* function

$$\lim_{N \to \infty} \mathcal{L}(N, \Theta) \to E_e\{\ln \Pr(e(t)|\Theta)\} \tag{4.133}$$

For the prediction error (innovations) models, the *AIC* is given by

$$AIC(N_\theta) = -2 \times \mathcal{L}(N, \Theta) + 2 \times N_\theta \tag{4.134}$$

but from Eq. (4.129), we have that

$$E_e\{\ln \Pr(e(t)|\Theta)\} \approx \frac{1}{N} \sum_{t=1}^{N} \ln \Pr(e(t)|\Theta) \qquad \text{for large } N \tag{4.135}$$

Now switching to vector notation for ease of development, $\mathbf{e} = [e(1) \cdots e(N)]'$, and following [39], then for $AR(N_a)$ model, the criterion with $N_\Theta = N_a$ becomes

$$AIC(N_\Theta) = -2\mathcal{L}(N, \Theta) + 2N_\Theta = -N \ln R_{ee} + 2N_\Theta \tag{4.136}$$

**Table 4.5. Order Tests**

| Test | Criterion | Remarks |
|---|---|---|
| Prediction error fit | $\frac{\sum_{k=1}^{N_\Theta} e^2(k)}{\sum_{k=1}^{N_\Theta} y^2(k)}$ | Locate knee of curve |
| Whiteness test | $\left\lvert \frac{R_{ee}(k)}{R_{ee}(0)} \right\rvert \leq \frac{\pm 1.96}{\sqrt{N}}$ | Check 95% of covariances lie within bounds |
| Signal-error test | $\hat{y}(t) = \frac{\hat{B}(q^{-1})}{\hat{A}(q^{-1})} u(t)$ | Check for fit visually |
| Final prediction error (*AR*, *MA*) | $FPE(N_\Theta) = \left( \frac{N+1+N_\Theta}{N-1-N_\Theta} \right) R_{ee}$ | Penalties for overparameterization reduced for large $N$ |
| Criteria for *AR* | $CAT = \frac{1}{N} \sum_{i=0}^{N_\Theta} \frac{1}{R_{ee}(i)} - \frac{1}{R_{ee}(N_\Theta)}$ | All $N$th-order models Transfer function must be estimated |
| Akaike information (*AR*, *MA*) | $AIC(N_\Theta) = -N \ln R_{ee} + 2N_\Theta$ | Penalties for overparameterization biased |
| Akaike information (*ARMA*) | $AIC(N_a, N_b) = -N \ln R_{ee} + 2(N_a + N_b)$ | Penalties for overparameterization biased |
| Minimum data length (*AR*, *MA*) | $MDL(N_\Theta) = -\ln R_{ee} + \left( \frac{N_\Theta}{2} \right) \ln N$ | Penalties for overparameterization consistent |
| Minimum data length (*ARMA*) | $MDL(N_a, N_b) = -\ln R_{ee} + \left( \frac{N_a + N_b}{2} \right) \ln N$ | Penalties for overparameterization consistent |

where $N_\Theta$ is the number of free parameters, $N$ the number of data samples, and $R_{ee}$ the (estimated) prediction error variance. Model order is determined by selecting the order corresponding to the minimum value of the criteria (*MAICE*) as before.

In the parameter estimation problem several families of $\Pr(e(t)|\Theta)$ with different forms of $\Pr(e(t)|\Theta)$ are given and we are required to select the best "fit." The *AIC* still provides a useful estimate of the entropy function in this case. Here the *AIC* is calculated for various model orders and the value with the minimum *AIC* is selected, that is, the *MAICE*. The *AIC* for the *AR*, *MA*, *ARMA* models are shown in Table 4.5.

Similar to the *AIC* is the minimum data length (*MDL*) order estimator ([10], [40]). For the $AR(N_a)$ model, we have

$$MDL(N_\theta) = -\ln R_{ee} + \frac{N_\theta}{2} \ln N \tag{4.137}$$

The difference between the *AIC* and *MDL* is in the final free parameter penalty (last) term for over estimating the order. The *AIC* tends to overestimate the order,

while the *MDL* parameter penalty term increases with data length $N$ and order $N_\theta$. It has also been shown that the *MDL* is a more consistent order estimator, since it converges to the true order as the number of measurements increase [40].

There are quite a wide variety of order estimators. For instance, it can also be shown that the final prediction error (*FPE*) criterion given by [22]

$$FPE(N_\Theta) = \left(\frac{N+1+N_\Theta}{N-1-N_\Theta}\right) R_{ee} \tag{4.138}$$

is asymptotically equivalent to the $AIC(N_\Theta)$ for large $N$ (data samples).

Other tests are also utilized to aid in the selection process. The *signal error test* of Table 4.5 consists of identifying *ARMAX* models for a sequence of increasing orders, exciting each model with the exogenous input to produce an estimated signal response, $\hat{s}(t)$, and overlaying the estimated and the actual response ($y(t)$ versus $\hat{s}(t)$) to select a "reasonable" fit to the data. That is, these tests are performed by producing the estimated (filtered or smoothed) response

$$\hat{s}(t) = \frac{\hat{B}(q^{-1})}{\hat{A}(q^{-1})} u(t) \tag{4.139}$$

Before we leave this section, let us briefly consider the *multichannel case* that is typically performed in the state-space framework using *canonical forms*[9] where the system $\Sigma \rightarrow \{N_x, N_u, N_y\}$ can be captured by the number of states $N_x$, the number of inputs $N_u$, and the number of outputs $N_y$. These order indexes are typically captured by the decomposition of the *Hankel* matrix, $\mathcal{H}$ (see Sec. 2.113). For instance, using the Popov controllability (see [24] for details), $\{\mu_i\}$ and observability $\{\nu_i\}$ invariants, we can easily deduce the internal structure of the multichannel system. That is, knowledge of these invariants enable the poles and zeros (matrix fraction transfer function form) of each subsystem to be extracted based on the total of controllability $N_\mu$ and observability $N_\nu$ invariants leading to the following order tests

$$\begin{aligned} N_x &= \rho(\mathcal{H}) \quad \text{rank} \\ N_\mu &= \text{independent ordered column indexes} \\ N_\nu &= \text{independent ordered row indexes} \\ N_x &= \textstyle\sum_{i=1}^{N_u} \mu_i = \sum_{i=1}^{N_y} \nu_i \end{aligned} \tag{4.140}$$

These order estimators can be solved for deterministic and noisy data sets using multichannel impulse response or covariance Hankel matrices (see [24], [25], [26] for more details).

We list some of the more popular order tests in Table 4.5 and refer the interested reader to [28] for more details. We will illustrate the *MAICE* procedure in the next section when we develop an application.

[9]Canonical forms are unique representations of multichannel state-space structures in a specific coordinate system such as those depicted in Chapter 2 for single-channel systems.

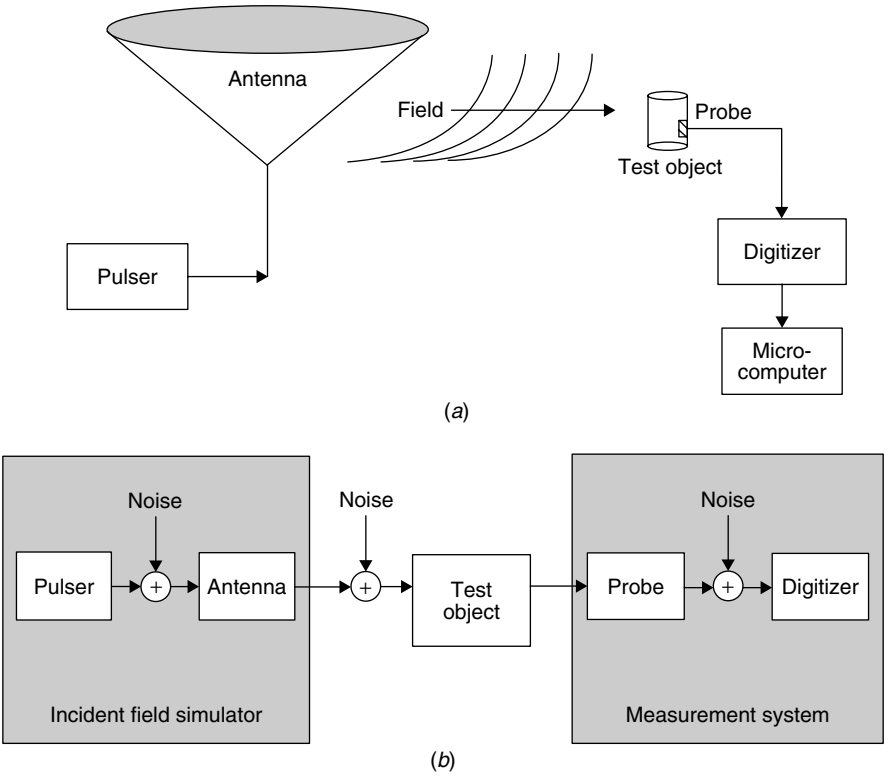



**Figure 4.15.** Electromagnetic test facility: (*a*) *EMTF* system diagram. (*b*) Signal-processing model.

## 4.7 CASE STUDY: ELECTROMAGNETIC SIGNAL PROCESSING

In this section we consider the development of model-based (parametric) signal processors to improve the signal processing capability of an electromagnetic test facility (*EMTF*). The *EMTF* was developed primarily to determine the transient response characteristics of scale models subjected to various *EM* excitations. The facility consists of three major subsystems: (1) incident field simulator, (2) test object, and (3) measurement system. The system diagram is depicted in Figure 4.15 (see [30], [31], [32] for details).

There are two classes of signal processing problems to consider in this diagram; subsystem interaction and the introduction of noise and distortions into the system. The major subsystems interact with each other and therefore must be taken into account when characterizing the test facility. There are several points where noise may be present in the facility. Internal noise and distortion may be introduced by the pulse generator, transmitting antennas, cables, sensors, and digitizers. External noise may enter the system at the test object from reflections or environmental

electromagnetic noise. The purpose of noise characterization is to remove the noise and compensate for the distortions.

The actual physical diagram of Figure 4.15*a* can be transformed into a signal processing model. We choose to use linear models, allowing us to utilize a wide range of signal processing and identification techniques. Figure 4.15*b* shows the model. The incident pulse simulator subsystem is modeled as a pulse generator transfer function, antenna transfer function, and propagation equation. The incident field generator is modeled with an impulse as the input and a noisy incident field pulse as output. The noise sources in the field generator include measurement noise on the output of the pulse generator and environmental electromagnetic noise and reflections on the output of the antenna. The output of the incident field generator is the input to the test object. One of the goals of this work is to provide a low noise input to the test object. The test object is modeled as a linear system according to the singularity expansion method [33]. The output of the object is sensed by a probe with a transfer function and digitized. The identification of the object transfer function has most chance of success when low noise input and undistorted object response can be achieved—this is the goal of signal processing.

Unlike typical electronic circuits where signals can be extracted with a simple probe, electromagnetic systems require special probes. Electromagnetic probes are small antennas excited by a field (the signal). When the field probe is placed a radial distance from the source antenna, and no object is present, the probe measures the *free field response*. When a test object is present, and the probe is located on the object, the probe measures the *object response*. Unfortunately, the free field (input) and object response (output) cannot be measured simultaneously because the field is disturbed by the presence of the object. Thus input and output data relative to a test object must be obtained by separate measurements. This practice brings up the problem of an output sequence generated by a input sequence different than the measured one.

The primary goal of the *EMTF* is to obtain the transient response of scale models. This response is classically characterized by the spectral content of the signals using fast Fourier transforms. We apply model-based (parametric) techniques to estimate the variety of signals and models required to achieve this goal.

Recall that the *identification problem* is simply given a set of noisy input and output data, find the best fit of a preselected parametric model to the data. Identification is most successful when both input and output are available simultaneously; however, due to the nature of the EM measurements, this is not possible. Therefore we choose to identify the free field response or input to the object a priori and then make object response measurements. The object identification is performed using the identified (and filtered) free field and measured object responses. In order to satisfy this objective, we must characterize the various component models depicted in Figure 4.15*b*. Once these models are identified, they can be used to estimate the free field response. Since the models are to be used *only* to predict the data, we use a "black-box" approach. Black-box modeling fits parametric models to measured data with no attempt to obtain physical meaning of the estimated parameters.

The solution to the identification problem consists of three basic ingredients:

1. Criterion
2. Model set
3. Algorithm.

Once the model set and criterion are selected various algorithms can be developed. For our problem, we choose the *prediction error criterion* given by

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

where $e(t) = y(t) - \hat{s}(t \mid t-1)$ is the *prediction error* and $\hat{y}(t \mid t-1)$ is the best estimate of $y(t)$ given data up to $t-1$.

For black-box modeling we choose the autoregressive moving average with exogenous input (*ARMAX*) model given by

$$A(q^{-1})y(t) = B(q^{-1})u(t) + C(q^{-1})e(t)$$

where $y,\ u,\ e$ are the process output, input, and noise sequences, and $A,\ B,\ C$ are the respective polynomials in backward shift operator $q^{-1}$ dimensioned $N_a$, $N_b,\ N_c$.

The *ARMAX* model was selected for a number of reasons. First, from Figure 4.15*b* we see that system or driving noise contaminates the object responses from various sources—the pulser, extraneous sources within the laboratory, and reflections. This noise is then filtered along with the actual plane wave excitation by the object. Therefore the coloring filter ($C(q^{-1})$) inherent in the *ARMAX* model is a key element to produce unbiased parameter estimates. Since the antenna excitation is independently estimated, prior to the object testing, an exogenous input ($u(t)$) must also be included in the model structure, again justifying the applicability of the *ARMAX* model.

Since much of the facility work is performed in real time, we use recursive prediction error algorithms of the previous section to identify the *ARMAX* models, namely, the recursive extended least-squares (*RELS*) and recursive maximum likelihood (*RML*) methods. (see Tables 4.4 and 4.6) These algorithms are employed for preprocessing and calibration runs.

A practical approach to perform the identification consists of the following:

1. Designing the test (input selection, etc.)
2. Preprocessing (averaging, filtering, etc.) the raw data
3. Estimating the *ARMAX* model order ($N_a$, $N_b$, $N_c$)
4. Estimating the *ARMAX* model parameters ($\{a_i\},\ \{b_i\},\ \{c_i\}$)
5. Performing prediction error tests for "fit" validation
6. Performing ensemble tests for model validation

**Table 4.6. Recursive Extended Least Squares (*RELS*)**

| | |
|---|---|
| Prediction error | |
| $e(t) = y(t) - \hat{y}(t \mid t-1) = y(t) - \phi\prime(t)\hat{\Theta}(t-1)$ | (Prediction error) |
| $R_{ee}(t) = \phi'(t)P(t-1)\phi(t) + \lambda(t)$ | (Prediction error variance) |
| Update | |
| $\Theta(t) = \Theta(t-1) + \mu K(t)e(t)$ for $\mu$ the step-size | (Parameter update) |
| $r(t) = y(t) - \overline{\phi}\prime(t)\hat{\Theta}(t)$ | (Residual update) |
| $P(t) = (I - K(t)\phi\prime(t))P(t-1)/\lambda(t)$ | (Covariance update) |
| $\lambda(t) = \lambda_0\lambda(t-1) + (1-\lambda_0)$ | (Forgetting factor update) |
| Gain | |
| $K(t) = P(t-1)\phi(t)R_{ee}^{-1}(t)$ | (Gain or weight) |
| $\hat{\phi}\prime(t) = [-y(t-1)\cdots - y(t-N_a) \mid u(t)\cdots u(t-N_b) \mid r(t)\cdots r(t-N_c)]$ | (phi-update) |

Using this approach each component of the *EMTF* can be characterized.

The crucial part of the identification process is estimating the order of the *ARMAX* model. Various order tests have been suggested to solve this problem (see Table 4.5) and a number of them are employed to initially screen models.

Once the order is determined and the parameters of the selected $ARMAX(N_a, N_b, N_c)$ are estimated, then the adequacy of the fit is determined by examining the statistical properties of the corresponding prediction error or innovations sequence, $\{e(t)\}$ [30]. The prediction errors must be zero-mean and statistically white, that is, 95% of the sample correlation estimates must lie within the bounds established by the interval, $I_e$, that is,

$$I_e = \left[0.0 \pm 1.96\frac{\hat{R}_{ee}(0)}{\sqrt{N}}\right]$$

The first component characterized is the pulse generator unit. Here the pulser is initiated and an ensemble of measurements generated. *MATLAB* [11] is used to identify the average pulser response. The identification process is initiated by performing a series of order tests using the *FPE* and *AIC* criteria to select the model order, $ARMAX(N_a, N_b, N_c)$, estimating the model parameters for the selected order, and finally performing signal and prediction error tests to assure a reasonable fit to the measured data. The results for the pulser identification are shown in Figure 4.16. Here we see the *signal error test* overlaying the estimated (solid) and measured (dashed) responses. Note that the fit is quite reasonable and smooth compared to

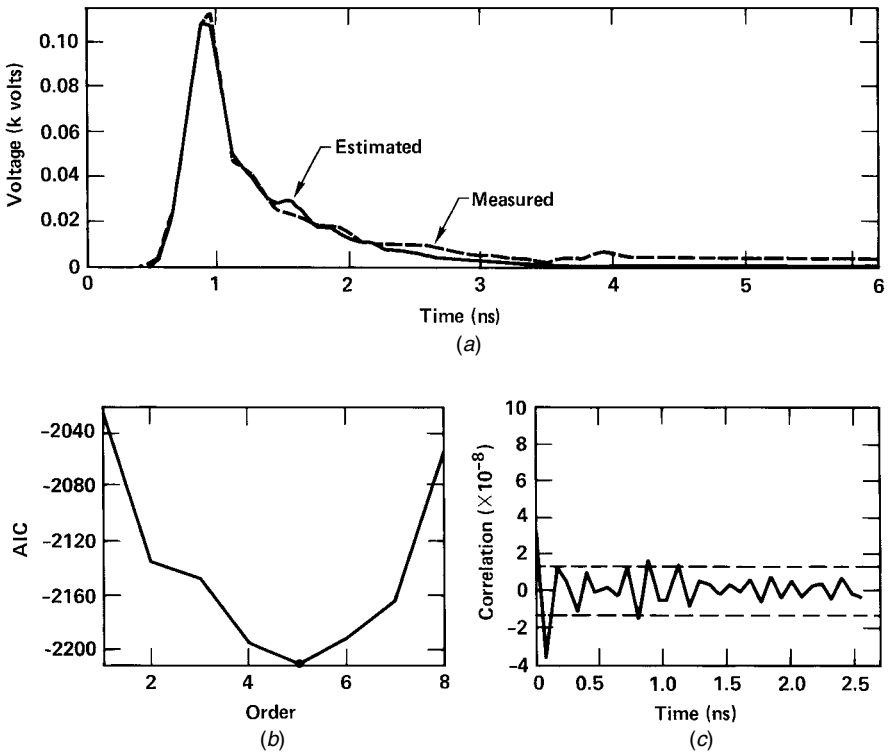



**Figure 4.16.** Pulser identification: (*a*) Signal error. (*b*) Order. (*c*) Whiteness tests.

the original measurement. We also note the results of the *order test* displaying the *AIC* versus order. Here we see that the minimum occurs for the *ARMAX*(5,4,1) model.[10] Note also that the *FPE* criteria yielded the same results (order) throughout all of the runs as well. These results are further validated from the prediction error whiteness test also shown in the figure. Aside from an early lag correlation, the test indicates that the errors are statistically white.

The model is also validated for prediction by calculating the residual error series over the ensemble of measurements, that is,

$$\epsilon_i(t) = y_i(t) - \hat{s}(t)$$

where $i$ corresponds to the $i$th record in the ensemble and $\hat{y}$ is obtained from (1) using the estimated model parameters. Once determined, the corresponding standard errors can be calculated to ascertain the adequacy of the model. That is,

[10]The order tests are performed over all of the parameters starting with the $a$- parameters to determine $N_a$ and then varying $N_b$ and $N_c$, respectively, in search of a minimum.

**Table 4.7. *EMTF* Models and Validation**

| Type | Pulse Generator Order 5 | Probe 2 |
|---|---|---|
| | Coefficient estimates | |
| $a_1$ | −1.262 | −0.904 |
| $a_2$ | 1.101 | 0.231 |
| $a_3$ | −0.926 | |
| $a_4$ | 0.445 | |
| $a_5$ | −0.169 | |
| $b_0$ | 0.110 | 0.529 |
| $b_1$ | 0.218 | −0.530 |
| $b_2$ | 0.324 | |
| $b_3$ | 0.399 | |
| $b_4$ | 0.243 | |
| $c_1$ | 0.559 | 0.519 |
| | Prediction error | |
| $\mu_e$ | $0.244 \times 10^{-5}$ | $1.83 \times 10^{-5}$ |
| $\sigma_e$ | $5.60 \times 10^{-4}$ | $5.38 \times 10^{-4}$ |
| | Validation | |
| $\beta_\varepsilon$ | $1.14 \times 10^{-2}$ | $-49 \times 10^{-5}$ |
| $\sigma_\varepsilon$ | $9.8 \times 10^{-3}$ | $1.008 \times 10^{-3}$ |
| $\text{RMS}_\varepsilon$ | $1.5 \times 10^{-2}$ | $1.11 \times 10^{-3}$ |

the standard error, bias error, and *RMS* error are given by

$$\sigma_\epsilon = \sqrt{E\{\epsilon^2(t)\} - E^2\{\epsilon(t)\}}$$
$$\beta_\epsilon = E\{\epsilon(t)\} - 0$$
$$RMS_\epsilon = \sqrt{\sigma_\epsilon^2 + \beta_\epsilon^2}$$

The results of this validation and the particular fit are depicted in Table 4.7.

The next component to consider is the bicone antenna shown in Figure 4.15*a*. From first principles, the antenna model can be derived [34] as

$$E_\Theta(r,t) = \frac{2V_p}{\ln\ \cot(\Psi/2)}\left[\frac{Z_{\text{bicone}}}{Z_{\text{bicone}} + 2Z_L}\right]\left(\frac{1}{r}\right)$$

where $E_\Theta$, $V_p$ are the electric field produced by the bicone and pulser voltage, $Z_{\text{bicone}}$, $Z_L$ are the bicone and pulser terminal impedances, and $\Psi$, $r$ are the bicone half angle and radial distance from the bicone ground plane center line. For the *EMTF*, it is known that $Z_{\text{bicone}} = 375.8$ ohms, $\Psi = 45^\circ$, $Z_L = 50$ ohms, and $r$ is determined by the particular facility configuration used in the given experiment.

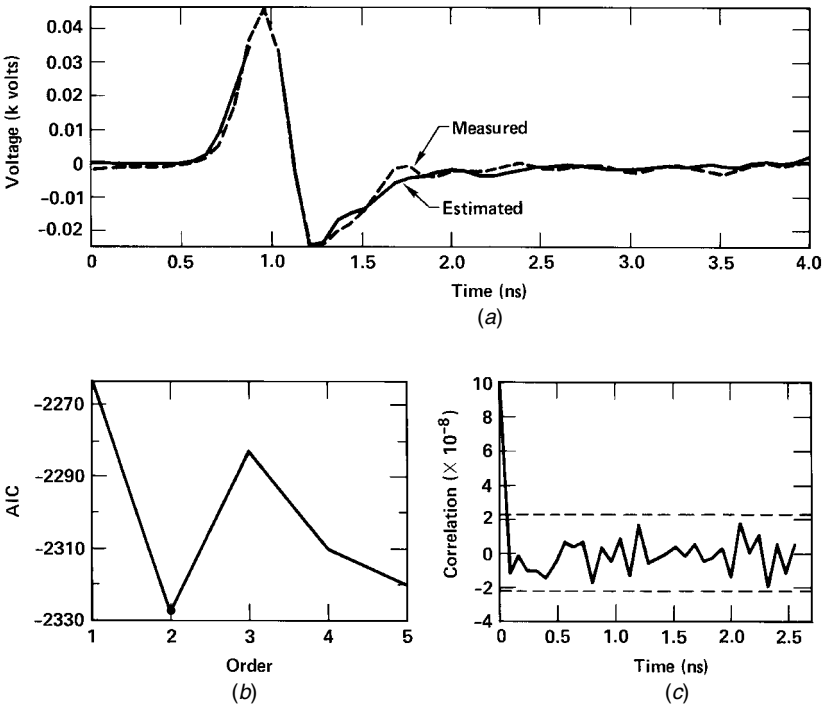



**Figure 4.17.** *D*-Dot probe identification: (*a*) Signal error. (*b*) Order. (*c*) Whiteness test.

Next the *D*-dot (sensor) probe is modeled. It transduces the derivative of the electric field intensity into an output voltage. The first principles model obtained from the manufacturer's specifications did not match the measurement data well; therefore a model was identified from the data. The results of the probe identification are shown in Figure 4.17. Here we again see the smooth, estimated response, $\hat{y}(t)$, obtained in the *signal error test* as well as the corresponding *order test* that indicates that the *ARMAX*(2, 1, 1) model satisfies the minimum value of the *AIC* and *FPE* tests. The prediction error whiteness test is clearly white and the validation statistics are shown in Table 4.7 as well.

Finally we investigate the sampling scope, a prime instrument used in measuring EM responses. This scope has essentially a flat frequency response in the region of most interest ($\sim$1 GHz). Thus no modeling was necessary.

Since the measurement probes are bandlimited, their effect on the input and output data must be removed or deconvolved prior to identification. Deconvolution of sensor probes from the data can be achieved in a variety of ways. A simple (suboptimal) procedure is to use the inverse of the identified model and excite it with the sensor measurement to obtain an estimate of the deconvolved signal, say $\hat{u}(t)$, that is,

$$\hat{u}(t) = \hat{H}^{-1}(q^{-1})y(t)$$

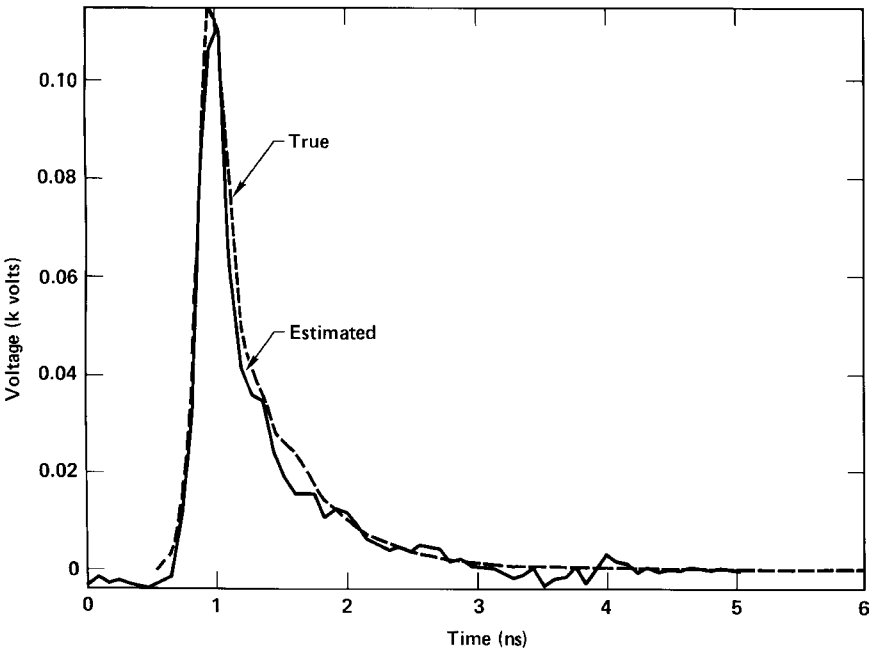



**Figure 4.18.** Deconvolved free field antenna response (probe deconvolution).

where $\hat{H}^{-1}(q^{-1}) = \hat{A}(q^{-1})/\hat{B}(q^{-1})$, the inverse model. In this approach we are merely filtering the data with the inverse of the identified model.

Another and more desirable approach is to actually solve the corresponding optimal (mean-squared error) deconvolution problem (see Section 4.3.2), which gives the solution of the following set of normal equations,

$$\hat{\mathbf{u}} = \mathbf{R}_{hh}^{-1}\mathbf{r}_{hy}$$

where $h(t)$ is the impulse response obtained from the identified model and $\mathbf{R}_{hh}$, $\mathbf{r}_{hy}$ are the corresponding auto- and cross-covariance matrices.

Since $\mathbf{R}_{hh}$ has a Toeplitz structure, it can be solved efficiently using the *LWR*(see Sec 4.3.1). Applying this recursion to the free field response measurement should yield the actual antenna response. The results are shown in Figure 4.18. Here we see that the deconvolver (128-weight finite impulse response filter) performs reasonably well as indicated by the similarity between the true modeled (dashed) response and the estimated response.The overall *EMTF* model was validated using an ensemble of free field measurements and the average error was quite reasonable. This completes the characterization of the *EMTF* for this particular configuration. The model developed is used for experimental design, reconfiguration studies, and, of course, primarily for use in identifying the EM response of an object.

Next we discuss the application of the *EMTF* to a thin-wire dipole—a fundamental electromagnetic object. The purpose of this experiment is to develop a

model of the dipole which can predict its response when subjected to various excitations. It has been shown that a dipole can be represented in the time domain by a series of damped exponentials using the singularity expansion method (SEM) [33]. In general, the EM response can be represented as

$$y(\mathbf{r}, t) = \sum_i \nu_i(\mathbf{r}) e^{s_i t} \eta_i(\mathbf{e}, s_i) u(t) \tag{4.141}$$

where

$s_i$ = natural complex frequency dependent on object parameters, ($s_i = d_i \pm j2\pi f_i$)
$\nu_i$ = natural mode, a nontrivial solution of field equations at $s_i$
$\mathbf{r}$ = spatial coordinates of position over the body
$\eta_i$ = coupling coefficient vector, the strength of the oscillation
$\mathbf{e}$ = exciting field characteristics

The most important feature of this model is the set of complex frequencies and natural modes ($\{s_i\}$, $\{\nu_i(\mathbf{r})\}$) which are excitation independent (not a function of $\mathbf{e}$). The effect of the exciting wave is in the coupling coefficients. To completely characterize the electromagnetic interaction, we need only $s_i$, $\nu_i$, and $\eta_i$. In most *EM* problems the quantities of interest are the poles and residues, which can be represented in the frequency domain by

$$Y(\mathbf{r}, s) = \sum_i \frac{R_i(\mathbf{r})}{s + s_i} \tag{4.142}$$

where $R_i$ is a complex residue such that $R_i(\mathbf{r}) = \nu_i(\mathbf{r})\ \eta_i(\mathbf{e},\ s_i.)$

Thus the response of an *EM* object can be predicted by its poles and residues ($\{s_i\}$, $\{R_i(\mathbf{r})\}$). Since the residues are dependent on characteristics of the excitation due to $\eta$, the electromagnetics researcher is primarily interested in the poles. The *EM* response prediction problem becomes that of finding the object poles. From the signal processing viewpoint this is a system identification problem, solved in discrete time and transformed to the $s$-plane.

To study dipole scattering and validate *EMTF* models, 0.4 meter, thin-wire dipole was mounted on the *EMTF* ground plane and excited with a transient electromagnetic field. Its response was measured by means of an electric-field probe mounted at its center and connected directly to a sampling scope. These data were low-pass filtered by an anti-aliasing filter, decimated by a factor of two, and deconvolved. Furthermore they were obtained from an ensemble average of 10 records. From the measured data shown in Figure 4.19, the first two resonances appear to be adequately excited, but the others are not and are probably contaminated with noise. Therefore we should expect to be able to identify the first two resonances easily but to have trouble with the others because of the low signal levels. Two algorithms were used to estimate the dipole parameters: the *RELS* algorithm

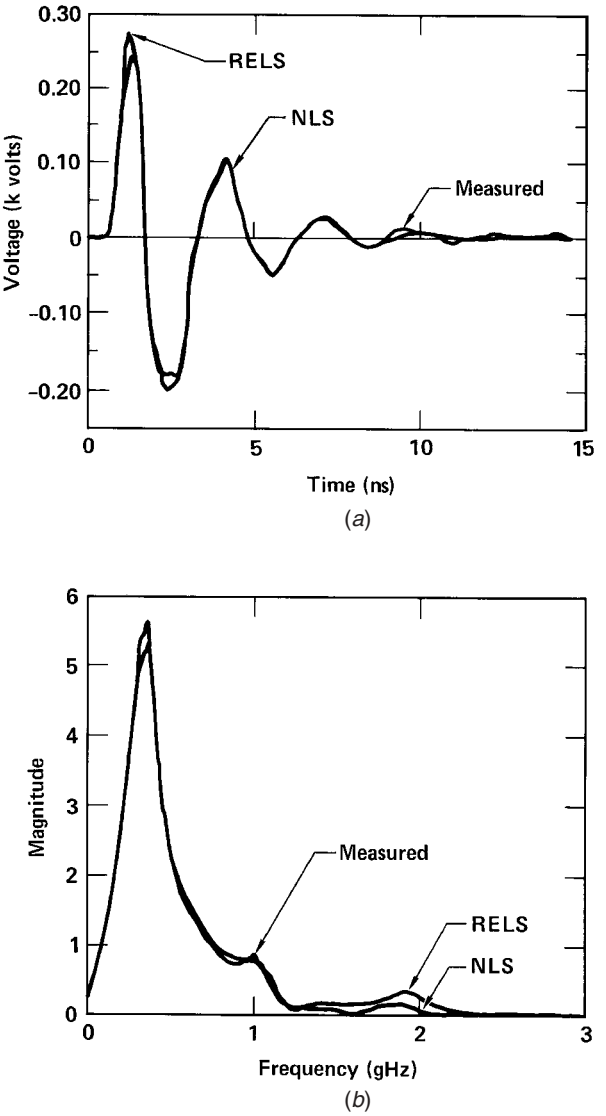



**Figure 4.19.** Identified dipole response: (*a*) Temporal. (*b*) Spectra.

for prediction error modeling, and a nonlinear least-squares *NLS* algorithm for output error modeling [35]. The RELS algorithm, which was mentioned as a special case of the *RPEM* algorithm in the previous section, was iterated a number of times enabling it to achieve performance equivalent to the off-line maximum likelihood [7].

The *NLS* algorithm is (simply) a nonlinear least-squares numerical optimization algorithm which determines the parameters of an *ARX* model, that is,

*ARMAX*$(N_a, N_b, 0)$. It produces parameter estimates $\Theta$ that minimize the mean-squared *output error*,

$$\min_{\Theta} J(\Theta) = E\{\epsilon^2(t)\} \approx \sum_i \epsilon^2(i) \tag{4.143}$$

where

$$\epsilon(t) = y(t) - \hat{y}_m(t)$$

and $\hat{y}_m(t)$ is obtained by exciting the identified model with the exogenous input, that is,

$$A_m(q^{-1})\hat{y}_m(t) = B_m(q^{-1})u(t) \tag{4.144}$$

The parameter vector $\theta$ is updated using the classical gradient iteration equation from numerical optimization,

$$\mathbf{\Theta}(i+1) = \mathbf{\Theta}(i) + \triangle(i)\mathbf{d}(\mathbf{i}) \tag{4.145}$$

where $\triangle(i)$ is the step-size and $\mathbf{d}(\mathbf{i})$ is a descent direction.

The algorithm utilizes Newton's method and has proved very robust for transient data. Note that this is a computationally intensive (off-line) output error algorithm, rather than a recursive (one-step) prediction error algorithm. For Newton's method the direction vector is given by

$$\mathbf{d}(\mathbf{i}) = -\left[\frac{\partial^2 J(\Theta_i)}{\partial \Theta^2}\right]^{-1}\left[\frac{\partial J(\Theta_i)}{\partial \Theta}\right]'$$

and the step size is varied according to the Armijo rule [35].

Models were tried for several different orders and for several different starting points. Many local minima were observed for orders ranging from 5 to 15. To obtain better starting points, the recursive least-squares (*RLS*) algorithm was used. Since the dipole theoretically contains an infinite number of poles, large order models were selected to approximate as many of the theoretical poles as possible. Order tests performed with the *RLS* algorithm revealed that either a twelfth or fifteenth-order models contained approximations to most of the theoretically predicted resonances and provided reasonable fits to the data as indicated by the response and Fourier spectrum in Figure 4.19. Furthermore the prediction error sequence satisfied a statistical test for whiteness. We decided that an *ARMAX*(15, 12, 0) model that we estimated using *RLS* was a reasonable starting point for the *RELS* and *NLS* algorithms.

The *RELS* and *NLS* algorithms were applied to the data records. The *RELS* algorithm was used to estimate an *ARMAX*(15, 12, 2) model, and the result, according to the *FPE* and *AIC*, was found to be superior to the model estimated with *RLS*. Similarly an output error model, with $N_a = 15$ and $N_b = 12$, was estimated with

*NLS*. The identified response and spectrum are also shown in Figure 4.19. The fits were reasonable.

The identified parameters are transformed to the continuous domain using the impulse-invariant transformation [36], which first requires the partial fraction expansion of the identified model to produce,

$$H(z) = \sum_i \frac{R_i}{1 + z^{-1} z_i} + \frac{R_i^*}{1 + z^{-1} z_i^*} \tag{4.146}$$

where $z_i = e^{s_i \Delta T}$, $s_i = -d_i \pm j2\pi f_i$ is a complex pole, and $R_i$ is a complex residue.

Once this is accomplished the impulse-invariant mapping (see Chapter 2) is given by

$$\left( \left\{ \frac{1}{\Delta T} \ln z_i \right\}, \{R_i\} \right) \longmapsto (\{s_i\}, \{R_i(\mathbf{r})\})$$

We note that using the impulse-invariant transformation, we have implicitly assumed that the continuous system or the EM object response in this application can be characterized by a bandlimited response or the transformation will produce aliased results. This is in fact a valid assumption, since bandlimiting occurs during both data acquisition and decimation processing.

Tesche [37] has calculated theoretical values for the poles of a dipole. These results are dependent on both its length and its diameter. Typically, the results are normalized to remove the effects of length. Table 4.8 shows the theoretical and the estimated poles. Since the response was measured at the center of the dipole, only the odd numbered poles are excited. It is clear that neither algorithm gave good estimates of the damping factors (real part of the poles). This result is not unexpected because of the limited amount of data in a transient signal. Both algorithms performed much better on the frequency estimates (imaginary parts) and were even able to extract some of the higher frequency resonances. We conclude that for a simple object such as a dipole the singularity expansion method coupled with an appropriate model-based parametric technique can be a useful tool for evaluating transient responses.

## 4.8 EXPONENTIAL (HARMONIC) *MBP*

In this section we develop model-based (parametric) processing techniques using the exponential model as the fundamental building block and then investigate the important special case of the harmonic (undamped exponential) model. This fundamental algorithm is based on the Prony technique [38] of extracting damped exponentials from noisy data.

**Table 4.8. Identified Poles versus Analytical Poles for 40 cm Dipole Antenna**

| | Theoretical | | *RELS* (on-line)—*ARMAX* (15, 12, 2) | | | | *NLS* (off-line)—*ARMAX* (15, 12, 0) | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Pole | $-d(\times 10^8)$ | $f$(GHz) | $-\hat{d}$ $(\times 10^8)$ | $\hat{d}_{\text{err}}$ | $\hat{f}$(GHz) | $\hat{f}_{\text{err}}$(GHz) | $-\hat{d}(\times 10^8)$ | $\hat{d}_{\text{err}}$ | $\hat{f}$(GHz) | $\hat{f}_{\text{err}}$(GHz) |
| (1) | 0.30 | 0.35 | 0.78 | 1.57 | 0.33 | 0.04 | 0.79 | 1.59 | 0.33 | 0.04 |
| (3) | 0.53 | 10.87 | 1.05 | 0.99 | 10.77 | 0.01 | 0.84 | 0.59 | 10.87 | 0 |
| (5) | 0.66 | 18.25 | 0.40 | 0.30 | 19.35 | 0.06 | 1.43 | 1.17 | 17.17 | 0.06 |
| (7) | 0.79 | 25.65 | 2.38 | 2.01 | 28.19 | 0.10* | 1.03 | 0.31 | 18.99 | 0.26 |
| (9) | 0.99 | 32.76 | 1.75 | 0.76 | 37.83 | 0.16* | 1.02 | 0.02 | 35.90 | 0.10 |
| (11) | 1.32 | 40.04 | 1.77 | 0.347 | 48.13 | 0.20 | 0.50 | 0.62 | 49.15 | 0.23 |

$a_{\text{err}} = \frac{a_{\text{TRUE}} - \hat{a}}{a_{\text{TRUE}}}$

* Residue order of magnitude low

### 4.8.1 Exponential *MBP*

Let us assume that the measured data are characterized by a set of $N_e$ complex exponential signals in additive random noise given by

$$y(t) = s(t) + n(t) \qquad \text{for } t = 0, \ldots, N-1 \tag{4.147}$$

with the damped exponential signal model

$$s(t) = \sum_{i=1}^{N_e} A_i e^{p_i t} \tag{4.148}$$

where $A_i$ is complex and $p_i$ is a complex pole defined by

$$p_i := d_i + j\Omega_i$$

for $d_i$ the damping and $\Omega_i$ the angular frequency.

An optimization approach to solve the model-based parameter estimation problem is based on the sum-squared (model) error (*SSQE*) between the data and the signal model, that is,

$$\mathcal{J}(\Theta) = \sum_{t=0}^{N-1} (y(t) - \hat{s}(t;\Theta))^2 \tag{4.149}$$

for $s(t;\hat{\Theta}) = \hat{s}(t)$ with the parameter set defined by: $\Theta := \{A_i, d_i, \Omega_i, N_e\}$, the respective unknown complex amplitude, damping, frequency and model order. Direct minimization of the *SSQE* is a nonlinear optimization problem. Therefore a popular alternative is to use a suboptimal technique based on the Prony method ([38], [39]). This involves solving two sets of linear equations with intermediate polynomial rooting that is the nonlinear component of the problem.

The crucial ingredient leading to the Prony technique is to realize that Eq. (4.148) is the solution of a homogeneous linear difference equation with appropriate initial conditions $\{\hat{s}(0), \ldots, \hat{s}(N_e - 1)\}$, that is,

$$\sum_{i=0}^{N_e} a_i \hat{s}(t-i) = 0 \qquad \text{for } \; a_0 = 1, N_e \le t \le N-1 \tag{4.150}$$

Taking $Z$-transforms, we obtain the characteristic equation of the system relating coefficients of the difference equation to the roots or equivalently, the complex exponentials,

$$A(z) = \sum_{i=0}^{N_e} a_i z^{-i} = \prod_{i=1}^{N_e} (1 - p_i z^{-1}), \qquad a_0 = 1$$

or $A(z)$ has $\{p_i\}$ as its roots and coefficients $\{a_i\}$ when expanded as above. Based on this representation the extended Prony technique can be developed.

We first define the *modeling error* as

$$e(t) := y(t) - \hat{s}(t) \qquad \text{for } 0 \le t \le N-1 \tag{4.151}$$

Alternatively, solving for $\hat{s}(t)$ and substituting, we obtain

$$e(t) = y(t) + \sum_{i=1}^{N_e} a_i \hat{s}(t-i) = y(t) + \sum_{i=1}^{N_e} a_i \left(y(t-i) - e(t-i)\right)$$

Now solving this expression for $y(t)$, we obtain

$$y(t) = -\sum_{i=1}^{N_e} a_i y(t-i) + \sum_{i=0}^{N_e} a_i e(t-i) \qquad \text{for } N_e \le t \le N-1 \tag{4.152}$$

or combining like variables gives

$$A(q^{-1})y(t) = A(q^{-1})e(t) \tag{4.153}$$

which is an equivalent *ARMA* model, $(ARMAX(N_a, 0, N_a))$. Note the indexes on $t$ that follow directly from the limits of Eq. (4.150). If we attempt to construct an estimator for this model, it will also be nonlinear [38]. Therefore we develop a suboptimal approach called the *extended Prony technique* where the prediction error is defined as

$$\varepsilon(t) := \sum_{i=0}^{N_e} a_i e(t-i) \qquad \text{for } t = N_e, \ldots, N-1 \tag{4.154}$$

Substituting into Eq. (4.152) gives an *AR* model

$$A(q^{-1})y(t) = \sum_{i=0}^{N_e} a_i y(t-i) + \varepsilon(t) \tag{4.155}$$

Now expanding the *AR* model of Eq. (4.155) over $N_e \le t \le N-1$ gives the partitioned matrix

$$\begin{bmatrix} \varepsilon(N_e) \\ \vdots \\ \varepsilon(N-1) \end{bmatrix} = \left[\begin{array}{c|ccc} y(N_e) & y(N_e-1) & \cdots & y(0) \\ \vdots & & \ddots & \vdots \\ y(N-1) & y(N-2) & \cdots & y(N-1-N_e) \end{array}\right] \left[\begin{array}{c} 1 \\ \hline a_1 \\ \vdots \\ a_{N_e} \end{array}\right] \tag{4.156}$$

The compact vector-matrix form is

$$\mathcal{E} = \begin{bmatrix} \mathbf{y} & | & \mathbf{Y} \end{bmatrix} \begin{bmatrix} 1 \\ \hline -\mathbf{a} \end{bmatrix} = \mathbf{y} - \mathbf{Ya} \tag{4.157}$$

where $\mathbf{y} \in \mathcal{R}^{N-1-N_e \times 1}$, $\mathbf{Y} \in \mathcal{R}^{N-1-N_e \times N_e}$, $\mathbf{a} \in \mathcal{R}^{N_e \times 1}$.

The optimal least-squares parameter estimate is found by minimizing the sum-squared prediction error criterion over $N_e \leq t \leq N-1$, that is,

$$J_{\min} := \min_{\mathbf{a}} J(t) = \mathcal{E}'\mathcal{E} = \sum_{t=N_e}^{N-1} \varepsilon^2(t) \tag{4.158}$$

Using the chain rule of Eq. (3.12) with $a' = \mathcal{E}'$ and $b = \mathcal{E}$, we obtain the *orthogonality condition*

$$\nabla_{\mathbf{a}} \mathcal{J} = 2\left(\nabla_{\mathbf{a}} \mathcal{E}'\right)\mathcal{E} = 2\sum_{t=N_e}^{N-1} \varepsilon(t) \frac{\partial \varepsilon(t)}{\partial a_k} = 0 \tag{4.159}$$

which can be rewritten as

$$\nabla_{\mathbf{a}} \mathcal{J} = 2\left(\nabla_{\mathbf{a}} (\mathbf{y} - \mathbf{Ya})'\right)\mathcal{E} = -2\mathbf{Y}'(\mathbf{y} - \mathbf{Ya}) = \mathbf{0} \tag{4.160}$$

Solving for $\mathbf{a}$, we obtain the familiar *LS* estimate

$$\hat{\mathbf{a}} = \left[\mathbf{Y}'\mathbf{Y}\right]^{-1}\mathbf{Y}'\mathbf{y} \tag{4.161}$$

with the corresponding filtered data estimate as

$$\hat{\mathbf{y}} = \mathbf{Y}\hat{\mathbf{a}} = \mathbf{Y}\left[\mathbf{Y}'\mathbf{Y}\right]^{-1}\mathbf{Y}'\mathbf{y} \tag{4.162}$$

The minimum *SSQE* is then given by

$$\mathcal{J}_{\min} = \mathcal{E}'\mathcal{E} = (\mathbf{Y}\hat{\mathbf{a}})'(\mathbf{Y}\hat{\mathbf{a}}) = \hat{\mathbf{y}}'\hat{\mathbf{y}} \tag{4.163}$$

Once the $\{\hat{a}_i\}$ are estimated using *AR* techniques, the complex exponentials $\{p_i\}$ are determined from the roots of the characteristic equation, $A(z)$. The set of complex gains $\{A_i\}$ can be estimated by expanding Eq. (4.148) with $i = 1, \ldots, N_e$. That is, the filtered estimate, $\hat{y}(t)$, from the *AR* model approximates the estimated signal, $\hat{y}(t) \approx \hat{s}(t)$:

$$\hat{y}(t) = \begin{bmatrix} e^{p_1 k\triangle T} & e^{p_2 k\triangle T} & \cdots & e^{p_{N_e} k\triangle T} \end{bmatrix} \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_{N_e} \end{bmatrix}$$

$$\text{for } t = k\triangle T, k = 0, \ldots, N-1 \tag{4.164}$$

**Table 4.9. Prony MBP**

Parameter Estimation

$$\hat{\mathbf{a}} = \left[\mathbf{Y}'\mathbf{Y}\right]^{-1}\mathbf{Y}'\mathbf{y}$$

Filter Response

$$\hat{\mathbf{y}} = \mathbf{Y}\hat{\mathbf{a}}$$

Pole (Root) Estimation

$$A(z) = \prod_{i=1}^{N_e}(1 - p_i z^{-1}) = \sum_{i=1}^{N_e}\hat{a}_i z^{-i}$$

Amplitude Estimation

$$\hat{\mathbf{A}} = \left[\mathbf{P}'\mathbf{P}\right]^{-1}\mathbf{P}'\hat{\mathbf{y}}$$

Signal Estimation

$$\hat{s}(t) = \sum_{i=1}^{N_e}\hat{A}_i e^{\hat{p}_i t}$$

to obtain

$$\mathbf{PA} = \hat{\mathbf{y}} \tag{4.165}$$

where the $\mathbf{P} \in R^{N\times N_e}$ is a Van der Monde matrix of exponentials given by

$$\mathbf{P} = \begin{bmatrix} 1 & \cdots & 1 \\ e^{p_1\triangle T} & & e^{p_{N_e}\triangle T} \\ \vdots & & \vdots \\ e^{p_1(N-1)\triangle T} & \cdots & e^{p_{N_e}(N-1)\triangle T} \end{bmatrix}$$

with

$$\mathbf{A}' = [A_1 \cdots A_{N_e}], \quad \hat{\mathbf{y}}' = [\hat{y}(1) \cdots \hat{y}(N_e)]$$

which can also be solved using least-squares techniques to give

$$\hat{\mathbf{A}} = \left[\mathbf{P}'\mathbf{P}\right]^{-1}\mathbf{P}'\hat{\mathbf{y}} \tag{4.166}$$

We summarize the Prony model-based parameter estimation technique here and in Table 4.9.

Criterion: $\mathcal{J}(\Theta) = \sum_{t=0}^{N-1} (y(t) - \hat{s}(t;\Theta))^2$

Models:

Measurement: $y(t) = s(t) + e(t)$

Signal: $s(t) = \sum_{i=1}^{N_e} A_i e^{p_i t}$

Noise: $R_{ee}$

Algorithm:

$$\hat{\mathbf{a}} = \left[\mathbf{Y}'\mathbf{Y}\right]^{-1}\mathbf{Y}'\mathbf{y}$$

$$\hat{\mathbf{A}} = \left[\mathbf{P}'\mathbf{P}\right]^{-1}\mathbf{P}'\hat{\mathbf{y}}$$

Quality: $\mathcal{J}_{\min} = \mathcal{E}'\mathcal{E}$

Note that the *orthogonality condition* of Eq. (4.159) can also be written in scalar form as

$$\frac{\partial J(t)}{\partial a_k} = 2\sum_{t=N_e}^{N-1} \varepsilon(t) y(t-k)$$

$$= 2\sum_{i=0}^{N_e} a_i \sum_{t=N_e}^{N-1} y(t-i)y(t-k) = 0 \qquad \text{for } k = 1, \ldots, N_e \tag{4.167}$$

where we have used the *AR* model of Eq. (4.155) and the fact that

$$\frac{\partial \varepsilon(t)}{\partial a_k} = \frac{\partial}{\partial a_k}\left(\sum_{i=0}^{N_e} a_i y(t-i)\right) = y(t-k) \tag{4.168}$$

If we assume that the $\{a_i\}$ are optimum, then the minimum *SSQE* is determined directly from Eq. (4.158) as

$$J_{\min} = \sum_{t=N_e}^{N-1} \varepsilon^2(t) = \sum_{t=N_e}^{N-1} \left(\sum_{k=0}^{N_e} a_k y(t-k)\right)\left(\sum_{i=0}^{N_e} a_i y(t-i)\right)$$

Taking the zeroth term from the first polynomial gives

$$J_{\min} = \sum_{t=N_e}^{N-1} \left(y(t) + \sum_{k=1}^{N_e} a_k y(t-k)\right)\left(\sum_{i=0}^{N_e} a_i y(t-i)\right)$$

Now rearranging the summations and multiplying through give the expanded expression

$$J_{\min} = \sum_{i=0}^{N_e} a_i \sum_{t=N_e}^{N-1} y(t)y(t-i) + \sum_{k=1}^{N_e} a_k \left[\sum_{i=0}^{N_e} a_i \sum_{t=N_e}^{N-1} y(t-i)y(t-k)\right]$$

Invoking the orthogonality condition of Eq. 4.167, the second term disappears, and we have the final result that

$$J_{\min} = \sum_{i=0}^{N_e} a_i \sum_{t=N_e}^{N-1} y(t)y(t-i) \tag{4.169}$$

Augmenting the orthogonality relations of Eq. (4.167) with this equation, we have that

$$\sum_{i=0}^{N_e} a_i \sum_{t=N_e}^{N-1} y(t-i)y(t-k) = \left\{ \begin{array}{ll} J_{\min} & \text{for } k = 0 \\ 0 & \text{for } k = 1, \ldots, N_e \end{array} \right\} \tag{4.170}$$

It is interesting to note that if we divide both sides of the Eq. (4.167) by $1/(N - N_e)$, then we obtain the *unbiased covariance estimator* as

$$\hat{C}_{yy}(i,k) = \frac{1}{N - N_e} \sum_{t=N_e}^{N-1} y(t-i)y(t-k) \qquad \text{for } \; k = 1, \ldots, N_e \tag{4.171}$$

which leads to the *Prony normal equations* as

$$\sum_{i=0}^{N_e} a_i \hat{C}_{yy}(i,k) = 0 \qquad \text{for } \; k = 1, \ldots, N_e \tag{4.172}$$

Therefore

$$J_{\min} = \sum_{k=0}^{N_e} a_k \hat{C}_{yy}(k,k) \tag{4.173}$$

Expanding over the indexes we obtain

$$\begin{bmatrix} \hat{C}_{yy}(1,1) & \hat{C}_{yy}(1,2) & \cdots & \hat{C}_{yy}(1,N_e) \\ \vdots & \vdots & & \vdots \\ \hat{C}_{yy}(N_e,1) & \hat{C}_{yy}(N_e,2) & \cdots & \hat{C}_{yy}(N_e,N_e) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{N_e} \end{bmatrix} = \begin{bmatrix} \hat{C}_{yy}(1,0) \\ \hat{C}_{yy}(2,0) \\ \vdots \\ \hat{C}_{yy}(N_e,0) \end{bmatrix} \tag{4.174}$$

which is the well-known *covariance method* of linear prediction theory ([12], [40]).

In compact vector matrix form we have the set of linear equations to be solved as

$$\mathbf{C}_{yy}\mathbf{a}(N_e) = \mathbf{c}_{yy} \tag{4.175}$$

They can be solved for the predictor coefficients as

$$\hat{\mathbf{a}}(N_e) = \mathbf{C}_{yy}^{-1}\mathbf{c}_{yy} \tag{4.176}$$

where $\mathbf{C}_{yy} \in \mathcal{R}^{N_e \times N_e}$ and $\mathbf{a}, \mathbf{c}_{yy} \in \mathcal{R}^{N_e \times 1}$.

The Prony technique can also be restricted to sinusoids yielding the *Prony harmonic decomposition* which follows directly from Eq. (4.148) with $\{d_i\}$ set to zero, that is,

$$s(t) = \sum_{i=1}^{N_s} A_i e^{j\Omega_i t} + A_i^* e^{-j\Omega_i t} = \sum_{i=1}^{N_s} A_i \cos(\Omega_i t + \phi_i) \tag{4.177}$$

The roots of the characteristic equation in this case are given by

$$A(z) = \prod_{i=1}^{N_s}(1 - z_i z^{-1})(1 - z_i^* z^{-1}) = \sum_{i=0}^{2N_s} a_i z^{2N_s - i} = 0 \tag{4.178}$$

The solution then follows as before by solving a set of linear vector-matrix equations except in this case the data matrix $\mathbf{Y}$ is both Toeplitz and Hankel (see [39] for details). Before we close this subsection, let us consider a simple example proposed in [41].

***Example 4.12*** Suppose that we have data in random noise generated by two complex exponential signals ($N_e = 2$) with unity amplitudes and poles at $-0.1 + j2\pi(0.52)$ and $-0.2 + j2\pi(0.42)$, that is,

$$s(t) = e^{(-0.1 + j2\pi(0.52))} + e^{(-0.2 + j2\pi(0.42))} \tag{4.179}$$

The noisy measurement data consists of the exponentials in additive zero-mean gaussian noise with variance of 0.001 at a *SNR* of 0 dB. We generate an ensemble of 101 realizations of 101 samples at $\triangle T = 1$ s and apply the extended Prony technique. We use the covariance method [40] to estimate the predictor coefficients $\{a_i\}, i = 1, \ldots, N_e$ and then estimate the corresponding set of poles, $\{p_i\}$, $i = 1, \ldots, N_e$. Since the data are contaminated by noise, we arbitrarily choose, $\hat{N}_e = 9$ corresponding to the two signal poles ($N_e = 2$) and the remaining ($\hat{N}_e - N_e$) noise poles. The pole and signal estimates are shown in Figure 4.20*a* and 4.20*b* with the average signal estimate in *c*. Here the signal estimates are given by the predictor outputs

$$\hat{s}(t) = \sum_{i=1}^{9} \hat{a}_i \hat{s}(t - i)$$

Each realization is plotted and overlayed in the figure. We note that the poles cluster about the true pole positions which are distinctly estimated. The corresponding

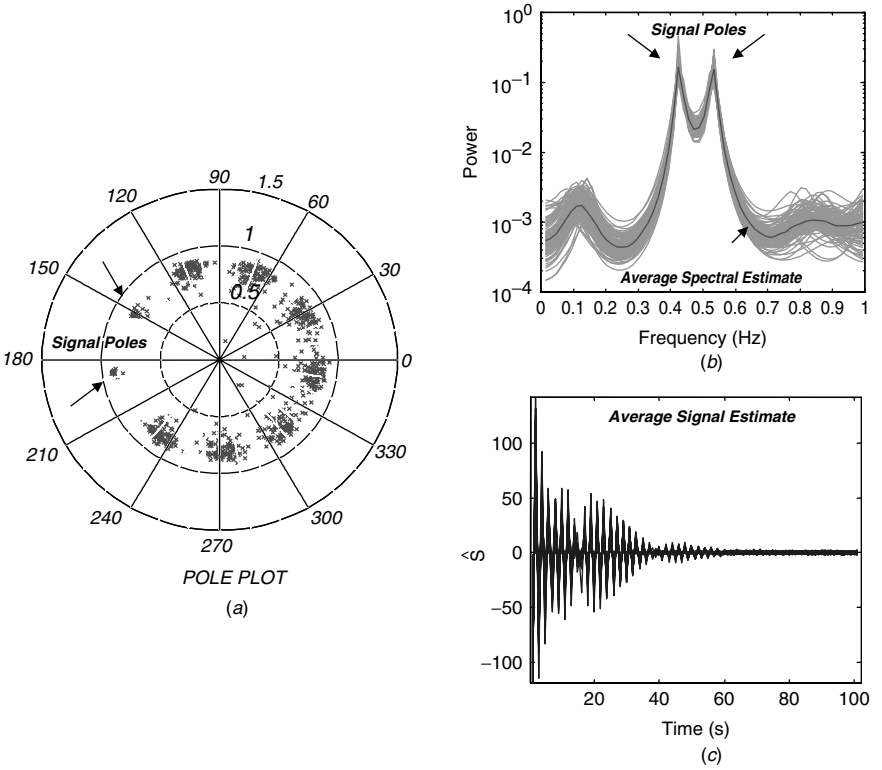



**Figure 4.20.** Model-based (parametric) estimation for a complex exponential model using the covariance method for 0 dB *SNR* and model order of 9 (2 signal poles): (*a*) Pole (signal and noise) estimation for 101 realizations. (*b*) Ensemble spectral estimates and average spectrum. (*c*) Average signal estimate.

noise poles are slightly concentrated and distributed uniformly about the unit circle. The signal estimates are reasonably concentrated and clearly dominated by the exponential signals. This completes the example.

This completes the subsection on the Prony techniques. In the next subsection we discuss an alternative method to solve this model-based problem.

### 4.8.2 *SVD* Exponential *MBP*

In this subsection we briefly discuss the *SVD* approach to the exponential model-based (parametric) estimation problem following the basic ideas of Kumaresan [41] and Cadzow [42]. As before the model-based estimation problem is based on finding the set of parameters defined by, $\Theta := \{A_i, d_i, \Omega_i, N_e\}$, the unknown complex amplitude, damping, angular frequency and model order that characterize the exponential model.

Let us return to the Prony problem as described earlier by Eq. (4.157) rewritten as the noisy data model

$$\mathbf{y} = \mathbf{Ya} + \mathcal{E} \tag{4.180}$$

The *SVD* approach is to choose a larger order $N_\epsilon$ (where $N_\epsilon > N/3$ for $N$ the number of data samples) and then perform an *SVD* of the data matrix to give

$$\left(U \Sigma V'\right) \mathbf{a} = \mathbf{x} \tag{4.181}$$

where $\mathbf{Y} := U \Sigma V'$ and $\mathbf{x} := \mathbf{y} - \mathcal{E}$.

Inverting the data matrix, we obtain the predictor coefficients

$$\hat{\mathbf{a}} = \left(V \Sigma^{-1} U'\right) \mathbf{x} \tag{4.182}$$

The idea is to estimate the order or the number of exponentials ($N_e$) by the largest of the singular values characterizing the data matrix, since

$$\Sigma = \left[ \; \sigma_{11}\sigma_{22} \; \cdots \; \sigma_{N_e N_e} \;\; | \;\; \sigma_{N_e+1 N_e+1} \; \cdots \; \sigma_{N_\mathcal{E} N_\mathcal{E}} \; \right]$$

One technique suggested by Golub [43] is the "best rank approximation" of $\mathbf{Y}$ defined by the ratio

$$\rho(N_e) = \frac{|\hat{Y}(N_e)|}{|Y(N_\mathcal{E})|} = \sqrt{\frac{\sum_{i=1}^{N_e} \sigma_{ii}}{\sum_{i=1}^{N_\mathcal{E}} \sigma_{ii}}} \tag{4.183}$$

Clearly, when $N_e \approx N_\mathcal{E}$, $\rho(N_e) \approx 1$ and the matrix $\hat{Y}(N_e) \longrightarrow Y(N_\mathcal{E})$ with high accuracy. The idea is to select a value of $N_e$ such that $\hat{Y}(N_e)$ approximates the data matrix. One way to accomplish a reasonable choice of $N_e$ is to choose a desired or target value of $\rho(N_e) = \rho^*$ for $0 < \rho^* < 1$ and continue to increase the estimated order $\hat{N}_e$ until $\rho(\hat{N}_e) \geq \rho^*$ is achieved. Once the number of exponentials is estimated, the predictor coefficients can then be estimated using the "truncated" *SVD* (see Chapter 3) as

$$\hat{\mathbf{a}}(\hat{N}_e) = V(\hat{N}_e)\Sigma^{-1}(\hat{N}_e)U'(\hat{N}_e)\mathbf{x} = \sum_{i=1}^{\hat{N}_e} \frac{\mathbf{v}_i}{\sigma_{\mathbf{ii}}} \mathbf{u}_i' \mathbf{x}(i) \tag{4.184}$$

The exponential parameters are then estimated as before in the Prony technique by rooting and least-squares estimation of the complex amplitudes.

The effect of employing the *SVD* of the data matrix and truncating to $\hat{N}_e$ is to increase the *SNR* of the data. In low *SNR* measurements its performance can be improved even further by using the correlation matrix instead of the data matrix [44]. That is, multiplying Eq. (4.181) by $Y'$ and defining $R := Y'Y'$ as before gives

$$R\mathbf{a} = Y'\mathbf{x} \tag{4.185}$$

Therefore

$$\hat{\mathbf{a}} = R^{-1} Y' \mathbf{x} \tag{4.186}$$

This method is called the autocorrelation matrix algorithm by Cadzow [44]. It can be improved as before in the data matrix case by estimating the number of exponentials using the best rank approximation and truncating the correlation matrices appropriately to give

$$\hat{\mathbf{a}}(N_e) = R^{-1}(N_e) Y'(N_e) \mathbf{x}(N_e) \tag{4.187}$$

Again, the predictor coefficients are estimated using the *SVD* approach, the resulting polynomial is factored into it roots and the corresponding amplitudes are estimated by solving the least-squares problem. We summarize the model-based *SVD* algorithm as follows:

Criterion: $\mathcal{J} = \mathcal{E}'\mathcal{E}$
Models:
  Measurement: $\mathbf{y} = \mathbf{Ya} + \mathcal{E}$
  Signal: $\mathbf{s} := \mathbf{Ya}$
  Noise: $R_{\epsilon\epsilon}$
Algorithm: $\hat{\mathbf{a}}(\hat{N}_e) = \sum_{i=1}^{\hat{N}_e} \frac{\mathbf{v}_i}{\sigma_{\mathbf{ii}}} \mathbf{u}_i' \mathbf{x}(i)$
Quality: $\mathcal{J}_{\min}(\hat{\mathbf{a}}) = \mathcal{E}'\mathcal{E}$

Let us revisit the previous example and apply the *SVD* methods.

***Example 4.13*** Using the data of Example 4.12 characterized by two complex exponential signals ($N_e = 2$) with unity amplitudes and poles at $-0.1 + j2\pi(0.52)$ and $-0.2 + j2\pi(0.42)$ at a *SNR* of 0 dB, we estimate the exponentials. We choose $\hat{N}_e = 9$ as before. The results for all three cases: modified covariance [39] matrix, correlation matrix, data matrix and truncated correlation matrix are shown in Figure 4.21*a*, *b*, *c* and *d*, respectively. In *a* and *b* we observe the pole estimation performance of the modified covariance matrix and correlation matrix *SVD* algorithms that appear quite similar to the covariance algorithm performance of the previous example. Next we apply the *SVD* data matrix and truncated *SVD* approach to each realization of the ensemble. The number of exponentials (model order 2) is estimated first using the best rank approximation and then the truncated equations of Eq. (4.181) are used to solve for the predictor coefficients. For the 101 realizations the dominant order selected was 5 corresponding to the two complex exponentials and three noise poles. As a result we observe that the poles cluster about the true pole positions (with a slight amplitude bias) that are distinctly estimated while the corresponding noise poles are more concentrated and distributed uniformly about the unit circle around the 3 pole noise model. We note that the truncated method approximates the solution compared to *c*. This completes the example.

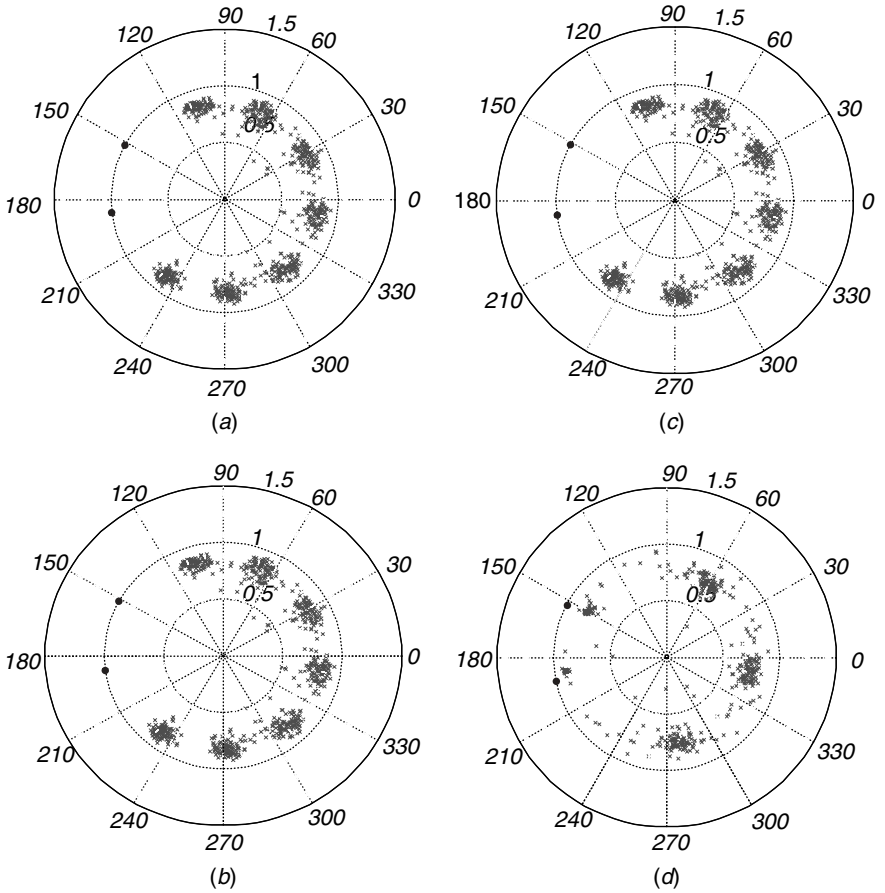



**Figure 4.21.** Model-based (parametric) estimation for a complex exponential model using the modified covariance, Cadzow's correlation, *SVD* and truncated *SVD* techniques for 0 dB *SNR* in 101 realizations (True position is circle, o): (*a*) Modified covariance method (9th order). (*b*) Correlation method (9th order). (*c*) *SVD* data matrix (9th order). (*d*) Truncated *SVD* (5th order).

In the next subsection we revisit the sinusoidal ($d_i = 0$) problem and investigate an alternate class of algorithms to estimate the model parameters.

### 4.8.3 Harmonic *MBP*

In this subsection we investigate a solution to the harmonics (sinusoids) in a noise model-based parameter estimation problem that is prevalent in a variety of applications, especially in spatial processing for direction-of-arrival and target localization problems [45].

As before in Chapter 2, recall that *complex harmonic model* is defined by

$$s(t) = \sum_{i=1}^{N_s} A_i e^{j\Omega_i t + \phi_i} \tag{4.188}$$

where $A_i$ is complex, $A_i = |A_i|e^{-j\theta}$; $\phi_i$ is random and uniformly distributed as $\phi \sim \mathcal{U}(-\pi, \pi)$ and $\Omega_i$ is the harmonic frequency. The set of $N_s$-complex amplitudes and harmonic frequencies, $[\{A_i\}, \{\Omega_i\}]$, $i = 1, \ldots, N_s$ are assumed deterministic but unknown.

The corresponding *measurement model* is characterized by the complex harmonic signal in additive random noise

$$y(t) = s(t) + n(t) \tag{4.189}$$

with $n(t)$, a zero-mean, random sequence with variance, $\sigma_n^2$.

Therefore its autocorrelation function is given by

$$R_{yy}(k) = R_{ss}(k) + R_{nn}(k) = P_i e^{j\Omega_i k} + \sigma_n^2 \qquad \text{for} \quad P_i := |A_i|^2 \tag{4.190}$$

since the signal is uncorrelated with the noise. Expanding this relation to incorporate the $N_s$-harmonics, we obtain

$$R_{yy}(k) = \begin{cases} \sum_{i=1}^{N_s} P_i + \sigma_n^2 \delta(k), & k = 0 \\ \sum_{i=1}^{N_s} P_i e^{j\Omega_i k}, & k \neq 0 \end{cases} \tag{4.191}$$

The model-based (parameter) estimation problem is therefore

> **GIVEN** the set of noisy measurements as in Eq. (4.189). **FIND** the best estimate of the signal $\hat{s}(t)$ characterized by the *harmonic model* of Eq. (4.188) and the set of unknown parameters, $\Theta := \{P_i, \Omega_i\}$.

There are a number of approaches by which this problem can be attacked, such as maximum likelihood parameter estimation, Prony's technique, and linear prediction. But here we choose to use the *power method* [46], that is, the cost function is based on maximizing the output power in the signal and the resulting function is related to its power spectrum. All peaks or roots of the spectrum locate the associated harmonic frequencies of the model.

We start by expanding the autocorrelation function of Eq. (4.191) over $k$ to obtain the associated $N \times N$ correlation matrices. The result gives a Hermitian Toeplitz matrix ($R^H = (R^*)'$) that has the following general properties [10]:

1. The set of distinct eigenvalues, $\{\lambda_i\}$ of a Hermitian matrix are *real*.
2. The set of eigenvectors, $\{\mathbf{e}_i\}$ of a Hermitian matrix corresponding to the distinct eigenvalues, $\{\lambda_i\}$, are *orthogonal* $\longrightarrow$ $\mathbf{e}_i^H \mathbf{e}_i = 0$ if $\lambda_i \neq \lambda_j$.

3. Any Hermitian matrix admits an eigen-decomposition such that

$$R = E\Lambda E^H \quad \text{with} \quad E \text{ unitary } (E^{-1} = E^H).$$

4. The inverse of a Hermitian matrix can be obtained for the eigen-decomposition as

$$R^{-1} = \left(E\Lambda E^H\right)^{-1} = E^{-H}\Lambda^{-1}E^{-1} = E\Lambda^{-1}E^H = \sum_{i=1}^{N} \frac{1}{\lambda_i}\mathbf{e}_i\mathbf{e}_i^H.$$

Now, performing the expansion of Eq. (4.191) over $k$, we obtain

$$R_{yy}(N) := R_{ss} + R_{nn} = V(\Omega)PV^H(\Omega) + \sigma_n^2 I \tag{4.192}$$

where $V(\Omega) \in \mathcal{C}^{N\times N_s}$ is the harmonic signal matrix with corresponding power matrix, $P \in \mathcal{C}^{N_s\times N_s}$ and $P = \text{diag}\ [P_1 \ \cdots \ P_{N_s}]$. The signal matrix, which is constructed from the harmonic model, is

$$V(\Omega) = \left[\mathbf{v}_1(\Omega) \mid \mathbf{v}_2(\Omega) \mid \cdots \mid \mathbf{v}_{N_s}(\Omega)\right] \qquad \text{with}$$
$$\mathbf{v}_i(\Omega) := [1 \ e^{j\Omega_i} \ \cdots \ e^{j(N-1)\Omega_i}]' \tag{4.193}$$

The output correlation matrix can be decomposed into the constituent eigenvalue-eigenvector systems as

$$R_{yy}(N) = E\begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_N \end{bmatrix} E^H = R_{ss} + R_{nn}$$
$$= V(\Omega)\begin{bmatrix} \lambda_1^s & & & 0 \\ & \ddots & & \\ & & \lambda_{N_s}^s & \\ 0 & & & 0 \end{bmatrix} V^H(\Omega) + \sigma_n^2\mathbf{I} \tag{4.194}$$

It follows from the fact that there are $N_s$-harmonics. Therefore we have that $\lambda_1 > \lambda_2 > \ \cdots \ > \lambda_N$ and

$$\lambda_i = \begin{cases} \lambda_i^s + \sigma_n^2, & i = 1, \ldots, N_s \\ \sigma_n^2, & i = N_s + 1, \ldots, N \end{cases} \tag{4.195}$$

The output correlation can also be written in partitioned form as

$$R_{yy}(N) = [E(N_s) \mid E(N - N_s)]\begin{bmatrix} \Lambda(N_s) & | & 0 \\ — & | & — \\ 0 & | & \Lambda(N - N_s) \end{bmatrix}\begin{bmatrix} E^H(N_s) \\ — \\ E^H(N - N_s) \end{bmatrix} \tag{4.196}$$

where both $E(N_s) \in \mathcal{C}^{N\times N_s}$ and $E(N-N_s) \in \mathcal{C}^{N\times(N-N_s)}$ are eigenvector matrices. Multiplying out the partitions gives the relation

$$R_{yy}(N) = E(N_s)\Lambda(N_s)E^H(N_s) + E(N-N_s)\Lambda(N-N_s)E^H(N-N_s) \quad (4.197)$$

or

$$R_{yy}(N) = [\mathbf{e}_1 \ \cdots \ \mathbf{e}_{N_s}] \begin{bmatrix} \lambda_1^s + \sigma_n^2 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N_s}^s + \sigma_n^2 \end{bmatrix} \begin{bmatrix} \mathbf{e}_1^H \\ \vdots \\ \mathbf{e}_{N_s}^H \end{bmatrix} + [\mathbf{e}_{N_s+1} \ \cdots \ \mathbf{e}_N] \begin{bmatrix} \sigma_n^2 & & 0 \\ & \ddots & \\ 0 & & \sigma_n^2 \end{bmatrix} \begin{bmatrix} \mathbf{e}_{N_s+1}^H \\ \vdots \\ \mathbf{e}_N^H \end{bmatrix} \quad (4.198)$$

This implies the decomposition

$$R_{yy}(N) = \sum_{i=1}^{N_s} \left(\lambda_i^s + \sigma_n^2\right) \mathbf{e}_i \mathbf{e}_i^H + \sum_{i=N_s+1}^{N} \sigma_n^2 \mathbf{e}_i \mathbf{e}_i^H \quad (4.199)$$

From this *spectral decomposition* it is possible to define various subspaces as follows:

1. The *signal subspace* is defined as the $N_s$-dimensional subspace spanned by the *signal eigenvectors*, $\{\mathbf{e}_i\}$, $i = 1, \ldots, N_s$, corresponding to the $N_s$-largest eigenvalues.
2. The *noise subspace* is defined as the $(N - N_s)$-dimensional subspace spanned by the *noise eigenvectors*, $\{\mathbf{e}_i\}$, $i = N_s + 1, \ldots, N$, corresponding to the remaining $(N - N_s)$-eigenvalues.
3. The *signal* and *noise subspaces* are *orthogonal*.

Applying this decomposition to our harmonic model-based estimation problem, we see that the *harmonic signal vectors*, $\mathbf{v}_i(\Omega)$ lie in the signal subspace spanned by the signal eigenvectors, $\{\mathbf{e}_i\}$; $i = 1, \ldots, N_s$, and the corresponding *noise vectors* lie in the orthogonal noise subspace spanned by $\{\mathbf{e}_i\}$, $i = N_s + 1, \ldots, N$. Thus the harmonic signal vectors lie in the signal subspace spanned by the first $N_s$-eigenvectors of $R_{yy}(N)$ and are *orthogonal* to each noise eigenvector, that is,

$$\mathbf{v}_i^H(\Omega)\mathbf{e}_j = 0 \qquad \text{for } \ i = 1, \ldots, N_s;\ j = N_s + 1, \ldots, N \quad (4.200)$$

which also implies that

$$V^H(\Omega)\mathbf{e}_j = 0 \qquad \text{for } \ j = N_s + 1, \ldots, N;\ V(\Omega) = [\mathbf{v}_1(\Omega) \mid \cdots \mid \mathbf{v}_{N_s}(\Omega)] \quad (4.201)$$

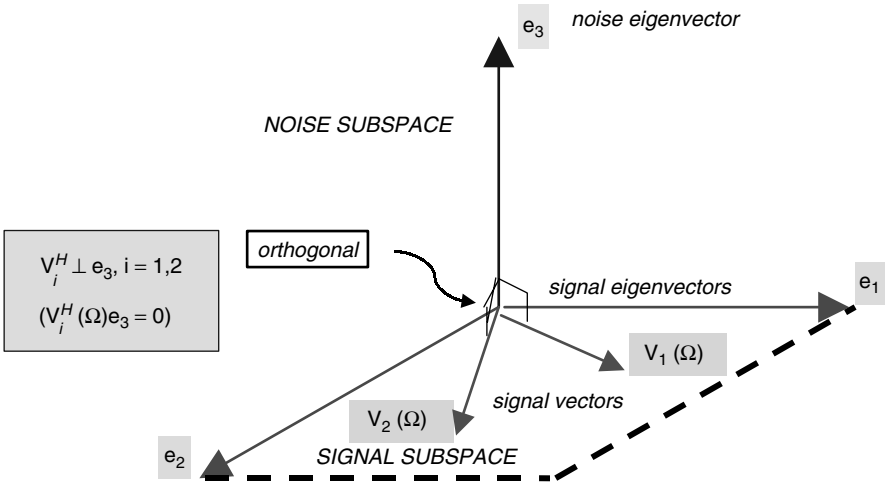



**Figure 4.22.** Signal and noise subspaces with two harmonic signal vectors and one noise eigenvector.

***Example 4.14*** Consider the following illustration of the orthogonality property of the signal and noise subspaces. Suppose that we have two harmonic signal vectors ($N_s = 2$) and one noise vector ($N = 3$), then the signal vectors, $\{\mathbf{v}_1(\Omega), \mathbf{v}_2(\Omega)\}$, lie in the subspace spanned by the two signal eigenvectors, $\{\mathbf{e}_1, \mathbf{e}_2\}$. The noise eigenvector is orthogonal to this space as shown in Figure 4.22

The harmonic frequency *power estimator* [46] follows directly from the orthogonality property of Eq. (4.200) as

$$P(\Omega) := \frac{1}{\sum_{j=N_s+1}^{N} |\mathbf{v}^H(\Omega)\mathbf{e}_j|^2} \text{ for } P(\Omega) \longrightarrow \infty \tag{4.202}$$

whenever $\Omega = \Omega_i,\ i = 1, \ldots, N_s$, where we have defined the *frequency vector*, $\mathbf{v}(\Omega)$, to represent the corresponding temporal Fourier frequencies.[11] So we see that if we calculate the power estimator over a set of frequencies, $\{\Omega\}$, then whenever this estimator passes through a harmonic frequency, $\Omega = \Omega_i$, a peak will occur in the power function. Simple peak detection then enables the estimation of the desired harmonic frequency. Another approach to estimating these frequencies is obtained by defining

$$E(\Omega) := \mathbf{v}^H(\Omega)\mathbf{e}(N) = \begin{bmatrix} 1 & e^{-j\Omega} & \cdots & e^{-j(N-1)\Omega} \end{bmatrix} \begin{bmatrix} e(0) \\ e(1) \\ \cdots \\ e(N-1) \end{bmatrix} = \sum_{t=0}^{N-1} e(t)e^{-j\Omega t} \tag{4.203}$$

[11] We use this frequency vector convention throughout. Note also that it differs from the harmonic signal vector in that it spans over all $\Omega$ not just the one corresponding to the estimated harmonic signal.

which is simply the *DtFT* of the eigen-filter, $\{e(t)\}$. Using the $Z$-transform relations, we see that the harmonic frequencies correspond to the *roots* of

$$E(z) = \sum_{t=0}^{N-1} e(t) z^{-t} = \prod_{k=1}^{N_s} (1 - e^{j\Omega_k} z^{-1}) \tag{4.204}$$

We summarize the generic harmonic model-based processor and follow with a discussion of more specific solutions:

| | |
|---|---|
| Criterion: | $J = \mathbf{w}' \mathbf{R}_{yy} \mathbf{w}$ |
| Models: | |
| Measurement: | $y(t) = s(t) + n(t)$ |
| Signal: | $s(t) = \sum_{i=1}^{N_s} A_i e^{j\Omega_i t + \phi_i}$ |
| Noise: | $R_{nn}$ |
| Algorithm: | $P(\Omega) = 1/\lvert E(\Omega)\rvert^2$ |
| Quality: | $R_{ss}$ |

This approach to harmonic model-based parameter estimation has evolved from the initial work of Pisarenko and Schmidt [47] to the more modern approaches of subspace estimation [48]. We briefly mention the particular methods and refer the reader to the detailed derivations and performance properties of the referenced papers. However, it should be noted in passing that the derivations typically follow the eigen-decomposition approach we have outlined in this subsection.

***Pisarenko Harmonic Decomposition*** The Pisarenko harmonic decomposition (*PHD*) method is based on the assumption that the *number* of $N_s$-harmonics in white additive noise are known a priori and it is desired to extract them from the data. It is further assumed that the dimension of the noise subspace is *unity* so that the data length is $N = N_s + 1$. Thus the $N_s$ largest eigenvalues correspond to the signal eigenvectors, $\mathbf{e}_i$, $i = 1, \ldots, N_s$, and the smallest or minimum eigenvalue $(N_s + 1)$ corresponds to the noise eigenvector, $\mathbf{e}_{N_s+1}$ with $\lambda_{\min} = \sigma_N^2$. As before, the signal eigenvectors are orthogonal to the noise eigenvector, that is,

$$\mathbf{v}_i(\Omega) \perp \mathbf{e}_{N_s+1} \Rightarrow \mathbf{v}_i^H(\Omega) \mathbf{e}_{N_s+1} = 0 \tag{4.205}$$

which can also be written as the *DtFT* relation of Eq. (4.203) to obtain $E_{\min}(\Omega)$. Using the power method, we have that

$$P_{PHD}(\Omega) := \frac{1}{\lvert \mathbf{v}^H(\Omega) \mathbf{e}_{N_s+1} \rvert^2} = \frac{1}{\lvert E_{\min}(\Omega) \rvert^2}$$
$$\text{for } P(\Omega) \longrightarrow \infty, \quad \Omega = \Omega_i, \ i = 1, \ldots, N_s \tag{4.206}$$

with the peaks corresponding to a set of harmonic frequencies or equivalently finding the roots of Eq. (4.204).

The corresponding power coefficients are then found by solving a set of linear equations developed from the eigen-equations

$$R_{yy}\mathbf{e}_i = \lambda_i \mathbf{e}_i, \qquad i = 1, \ldots, N_s \tag{4.207}$$

Multiplying both sides by $\mathbf{e}_i^H$ and substituting Eq. (4.192), we have that

$$\begin{aligned}
\mathbf{e}_i^H R_{yy}\mathbf{e}_i &= \mathbf{e}_i^H \left( \sum_{k=1}^{N_s} P_k \mathbf{v}_k(\Omega)\mathbf{v}_k^H(\Omega) + \sigma_n^2 I \right) \mathbf{e}_i \\
&= \mathbf{e}_i^H \lambda_i \mathbf{e}_i = \lambda_i, \qquad i = 1, \ldots, N_s \\
&= \sum_{k=1}^{N_s} P_k \mathbf{e}_i^H \mathbf{v}_k(\Omega)\mathbf{v}_k^H(\Omega)\mathbf{e}_i + \sigma_n^2 \mathbf{e}_i^H \mathbf{e}_i \\
&= \sum_{k=1}^{N_s} P_k |\mathbf{v}_k^H(\Omega)\mathbf{e}_i|^2 + \sigma_n^2 = \lambda_i
\end{aligned}$$

Then, solving, we have

$$\sum_{k=1}^{N_s} P_k |\mathbf{v}_k^H(\Omega)\mathbf{e}_i|^2 = \lambda_i - \sigma_n^2 \tag{4.208}$$

Defining $H_k(\Omega_i) = \mathbf{v}_k^H(\Omega)\mathbf{e}_i$ and expanding over $i$ and $k$, we obtain

$$\begin{bmatrix} |H_1(\Omega_1)|^2 & \cdots & |H_1(\Omega_{N_s})|^2 \\ \vdots & \ddots & \vdots \\ |H_{N_s}(\Omega_1)|^2 & \cdots & |H_{N_s}(\Omega_{N_s})|^2 \end{bmatrix} \begin{bmatrix} P_1 \\ \vdots \\ P_{N_s} \end{bmatrix} = \begin{bmatrix} \lambda_1 - \sigma_n^2 \\ \vdots \\ \lambda_{N_s} - \sigma_n^2 \end{bmatrix}$$

or compactly

$$H\mathbf{P} = \Lambda - \sigma_n^2 I \tag{4.209}$$

which is easily solved.

This is the *PHD* technique of estimating harmonics in white noise. This method can be generalized further by incorporating more of the noise eigenvectors and averaging which we discuss in the next subsection. Note, however, that the *PHD* method provides the foundation for the eigen-decomposition approaches to harmonic retrieval in noise.

***Multiple Signal Classification*** The MUltiple SIgnal Classification (*MUSIC*) method was developed by Schmidt [48] as an extension of the *PHD* technique and evolved from the spatial (array) signal processing area. It follows the derivation presented in this subsection with the steps summarized as follows:

1. The eigen-decomposition of the autocorrelation matrix is performed, $R = E\Sigma E^H$.
2. The number of harmonics, $N_s$, are determined using the *order estimation* or *best rank approximation* methods with the noise subspace dimensioned as $N - N_s$.
3. The *signal subspace* is defined by the set of largest eigenvalues and corresponding eigenvectors, $[\{\lambda_i\}, \{\mathbf{e}_i\}]$, $i = 1, \ldots, N_s$.
4. The *noise subspace* is defined by the set of the remaining (smallest) eigenvalues and corresponding eigenvectors, $[\{\lambda_i\}, \{\mathbf{e}_i\}]$, $i = N_s + 1, \ldots, N$.
5. The *power estimator* of Eq. 4.210 (to follow) is based on averaging over all of the *noise eigenvectors* to reduce the effects of spurious peaks, $|\mathbf{v}_i^H(\Omega)\mathbf{e}_j|^2 \longrightarrow \sum_{j=N_s+1}^{N} |\mathbf{v}^H(\Omega)\mathbf{e}_j|^2$.
6. The power is found by solving the linear least-squares problem of Eq. (4.211).

Using the noise averaging we have that the *MUSIC* power estimator is

$$\begin{aligned} P_{MUSIC}(\Omega) :&= \frac{1}{\mathbf{v}^H(\Omega)\left[E(N-N_s)E^H(N-N_s)\right]\mathbf{v}(\Omega)} \\ &= \frac{1}{\sum_{j=N_s+1}^{N} |\mathbf{v}^H(\Omega)\mathbf{e}_j|^2} \end{aligned} \tag{4.210}$$

Note also that root-*MUSIC* avoids calculating the power estimator and just "roots" the resulting polynomial as in Eq. (4.204).

The power is found by solving Eq. (4.192) using least-squares for $P$ as

$$\hat{P} = [V^H(\Omega)V(\Omega)]^{-1}V^H(\Omega)\left(R_{yy} - \sigma_N^2 I\right)V(\Omega)[V^H(\Omega)V(\Omega)]^{-H} \tag{4.211}$$

It should also be noted that the *Eigenvector method* (*EV*) of Johnson [49] is a weighted version of *MUSIC*, that is, the technique is identical except that instead of weighting by the identity as in *MUSIC*, the inverse of the noise eigenvalues is used. The result of this weighting is to reduce the effects of spurious peaks or roots even further. The corresponding *EV*-method has the following power estimator

$$\begin{aligned} P_{\mathrm{EV}}(\Omega) :&= \frac{1}{\mathbf{v}^H(\Omega)\left[E(N-N_s)\Lambda^{-1}(N-N_s)E^H(N-N_s)\right]\mathbf{v}(\Omega)} \\ &= \frac{1}{\sum_{j=N_s+1}^{N} \frac{1}{\lambda_j}|\mathbf{v}^H(\Omega)\mathbf{e}_j|^2} \end{aligned} \tag{4.212}$$

This completes the subsection on *MUSIC* and *EV* methods of model-based harmonic estimation using the power methods, we summarize as follows:

Criterion: $J = \mathbf{w}'\mathbf{R}_{yy}\mathbf{w}$
Models:
Measurement: $y(t) = s(t) + n(t)$
Signal: $s(t) = \sum_{i=1}^{N_s} A_i e^{j\Omega_i t + \phi_i}$
Noise: $R_{nn}$
Algorithm: $P_{\mathrm{EV}}(\Omega) = \dfrac{1}{\sum_{j=N_s+1}^{N} \frac{1}{\lambda_j} |\mathbf{v}^H(\Omega)\mathbf{e}_j|^2}$
Quality: $R_{ss}$

***Minimum Norm*** Another method related to the eigen-decomposition techniques is Minimum NORM method developed by Tufts [50]. It is the solution to a constrained optimization problem and is related to the subspace decomposition methods [51].

The *MNORM* method is based on defining a vector, say $\mathbf{a}$, constrained such that:

1. The vector $\mathbf{a}$ lies in the "noise" subspace (roots on the unit circle).
2. The vector $\mathbf{a}$ has a minimum norm (spurious roots inside the unit circle).
3. The first component of $\mathbf{a}$ is unity (nonzero solution guaranteed).

These constraints are easily explained in terms of the projection operators of Chapter 3. The projection operator, $P_N$, projects an arbitrary vector $\mathbf{n}$ onto the noise subspace as

$$\mathbf{a} = P_N\mathbf{n} \qquad \text{where} \quad P_N := E(N - N_s)E^H(N - N_s) \tag{4.213}$$

and $E(N - N_s)$ is the $C^{N\times(N-N_s)}$ noise eigenvector matrix.

The corresponding norm of $\mathbf{a}$ can be placed in terms of the projection of $\mathbf{n}$, that is,

$$\| \mathbf{a} \|^2 = \| P_N\mathbf{n} \|^2 = \mathbf{n}^H P_N^H P_N\mathbf{n} = \mathbf{n}^H P_N^2\mathbf{n} = \mathbf{n}^H P_N\mathbf{n} \tag{4.214}$$

since $P_N$ is a Hermitian projection matrix ($P_N = P_N^H$) and idempotent ($P_N^2 = P_N$).

The "unity constraint" can be placed in terms of the projection matrix as well

$$\mathbf{a}^H\mathbf{u}_1 = (P_N\mathbf{n})\mathbf{u}_1 = \mathbf{n}^H(P_N\mathbf{u}_1) = 1 \qquad \text{for } \mathbf{u}_1 \text{ a unit vector} \tag{4.215}$$

Using these constraints, we can now define the minimum norm estimation problem as *finding* an $\mathbf{n}$ such that

$$\min_{\mathbf{n}} \mathbf{n}^H P_N\mathbf{n} \qquad \text{such that} \qquad \mathbf{n}^H P_N\mathbf{u}_1 = 1 \tag{4.216}$$

The solution to this problem is developed as a constrained optimization problem which evolves from the *Lagrange multiplier* approach [52]. Define the cost function as

$$\min_{\mathbf{n}} \mathcal{J} = \mathbf{n}^H P_N \mathbf{n} + \gamma \left(1 - \mathbf{n}^H P_N \mathbf{u}_1\right) \tag{4.217}$$

where $\gamma$ is the Lagrange multiplier. Using the chain rule and differentiating Eq. (4.217), we have

$$\nabla_{\mathbf{n}} \mathcal{J} = P_N \mathbf{n} - \gamma (P_N \mathbf{u}_1) = 0 \tag{4.218}$$

Solving for $\mathbf{n}$, we obtain

$$\mathbf{n} = \gamma P_N^{-1} (P_N \mathbf{u}_1) = \gamma \mathbf{u}_1 \tag{4.219}$$

Substituting this result into Eq. (4.215) and solving for the multiplier gives

$$\gamma = \frac{1}{\mathbf{u}_1^H P_N \mathbf{u}_1} \tag{4.220}$$

Now, substituting for $\gamma$, we have

$$\mathbf{n} = \frac{\mathbf{u}_1}{\mathbf{u}_1^H P_N \mathbf{u}_1} \tag{4.221}$$

which leads to the *minimum norm* solution

$$\mathbf{a}_{\min} = P_N \mathbf{n} = \frac{P_N \mathbf{u}_1}{\mathbf{u}_1^H P_N \mathbf{u}_1} = \frac{E(N - N_s) E^H (N - N_s) \mathbf{u}_1}{\mathbf{u}_1^H E(N - N_s) E^H (N - N_s) \mathbf{u}_1} \tag{4.222}$$

The corresponding power function is therefore

$$P_{\min}(\Omega) := \frac{1}{|\mathbf{v}^H(\Omega) \mathbf{a}_{\min}|^2} \qquad \text{for } \mathbf{a}_{\min} = P_N \mathbf{n} \tag{4.223}$$

This completes the development of the *minimum norm* solution to the harmonic model-based estimation problem, we summarize this *MBP* as follows:

Criterion: $J = \mathbf{w}' \mathbf{R}_{yy} \mathbf{w}$
Models:
Measurement: $y(t) = s(t) + n(t)$
Signal: $s(t) = \sum_{i=1}^{N_s} A_i e^{j\Omega_i t + \phi_i}$
Noise: $R_{nn}$

Algorithm:

$$\mathbf{a}_{\min} = \frac{E(N-N_s)E^H(N-N_s)\mathbf{u}_1}{\mathbf{u}_1^H E(N-N_s)E^H(N-N_s)\mathbf{u}_1}$$

$$P_{\min}(\Omega) = \frac{1}{|\mathbf{v}^H(\Omega)\mathbf{a}_{\min}|^2}$$

Quality: $R_{ss}$

It should also be noted that development based on the *principal components* (eigenvectors) using the signal rather than noise subspace are also available. The Prony SVD algorithm discussed earlier can be applied to this problem and it can be considered a signal subspace approach. Power estimators using the signal subspace approach follow the same power function relations with the output correlation matrix approximated by the truncated SVD as before, that is,

$$\hat{R}_{yy}(N_s) \approx \sum_{i=1}^{N_s} \lambda_i \mathbf{e}_i \mathbf{e}_i^H = E(N_s)\Lambda(N_s)E^H(N_s) \tag{4.224}$$

Typical algorithms simply estimate the power function (spectrum) using the truncated correlation matrix such as the classical *Bartlett* estimator

$$P_{\text{BART}}(\Omega) = \frac{1}{N}\mathbf{v}^H(\Omega)\hat{R}_{yy}(N_s)\mathbf{v}(\Omega) \tag{4.225}$$

We demonstrate the performance of these power estimators in the following example.

***Example 4.15*** Suppose that we have data in random noise generated by two harmonic signals ($N_e = 2$) with unity amplitudes and poles at $j2\pi(0.52)$ and $j2\pi(0.42)$, that is,

$$s(t) = e^{(j2\pi(0.52))} + e^{(j2\pi(0.42))}$$

The noisy measurement data consists of the exponentials in additive zero-mean gaussian noise with variance of 0.001 at a *SNR* of 10 dB. We generate an ensemble of 101 realizations of 128 samples at $\triangle T = 2$ sec and apply the harmonic model-based estimation techniques. Since the data are contaminated by noise, we arbitrarily choose, $\hat{N}_e = 9$ corresponding to the two signal poles ($N_e = 2$) and the remaining ($\hat{N}_e - N_e$) noise poles. We first apply the covariance method of the previous subsection using the power method to identify the harmonics. The ensemble and median power estimates are shown in Figure 4.23*a*, respectively. Each realization is plotted and overlayed in the figure. We note that the spectral peaks cluster about the true pole positions that are distinctly estimated. The eigen-methods were

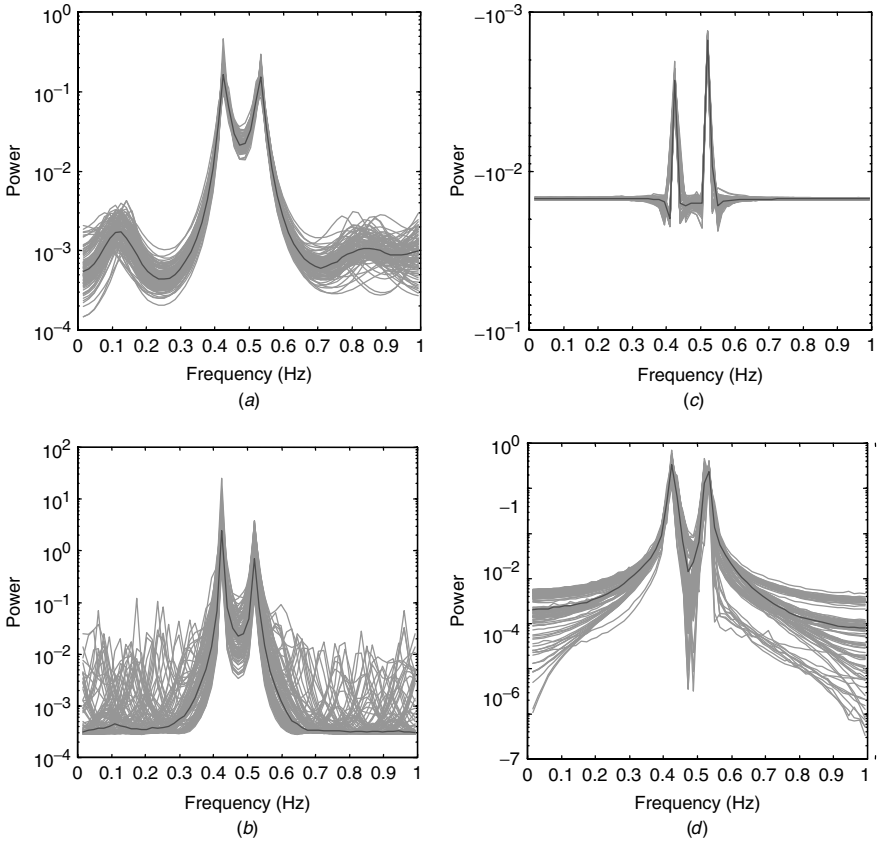



**Figure 4.23.** Model-based (parametric) estimation for a harmonic model using the power methods for 10 dB *SNR*: (*a*) COVARIANCE method ensemble and median spectra ($\hat{f}_1 = 0.425$ Hz, $\hat{f}_1 = 0.50$ Hz). (*b*) *MUSIC* method ensemble and median spectra ($\hat{f}_1 = 0.425$ Hz, $\hat{f}_1 = 0.50$ Hz). (*c*) *MNORM* method ensemble and median spectra ($\hat{f}_1 = 0.425$ Hz, $\hat{f}_1 = 0.50$ Hz). (*d*) BARTLETT principal component method ensemble and median spectra ($\hat{f}_1 = 0.425$ Hz, $\hat{f}_1 = 0.50$ Hz).

also applied to this problem for comparison. The *MUSIC* and *MNORM* approaches were applied and the results are shown in Figure 4.23*b* and 4.23*c* respectively. We note that the *MUSIC* spectrum shows many false noise peaks, while the *MNORM* does not. However, both methods appear to perform well estimating the signal harmonics. Finally, we applied the principle component approach using the Bartlett estimator and the results are shown in Figure 4.23*d*. Here the harmonic estimates are not quite as good, but certainly reasonable at this *SNR*. This completes the example.

This completes the discussion of harmonic estimators, next we consider similar solutions to the *wave-type* problems.

## 4.9 WAVE *MBP*

In this section we develop the concept of MBP for spatiotemporal signals or simply propagating waves. From the wave theory background and models of Chapter 2, we discuss the development of wave model-based processors, that is, the development of a processor to estimate the parameters capturing the propagating wave structure.

We start with the narrowband, *spatiotemporal wave model* defined by the noisy $L$-element sensor array measurement model

$$\mathbf{y}(t) = D(\kappa)\mathbf{s}(t) + \eta(t) \tag{4.226}$$

where $\mathbf{y}, \eta \in \mathcal{C}^{L\times 1}$ are the array measurement and noise vectors, $\mathbf{s} \in \mathcal{C}^{N_s\times 1}$ is the signal vector and $D \in \mathcal{C}^{L\times N_s}$ direction matrix. $D$ is parameterized with complex amplitude and phase elements defined by $d_{\ell m} := \alpha_\ell(\kappa_m)e^{j\omega_o\tau_\ell(\kappa_m)}$ for $\ell$ the sensor array element and $m$ the signal vector component with the parameters $\{\alpha_\ell(\kappa_m), \tau_\ell(\kappa_m)\}$ defined uniquely for planar or spherical wavefronts and $\kappa_m$ the corresponding spatial frequency or wavenumber ($\kappa_o = \omega_o/c = 2\pi/\lambda_o$) with $c$ the propagation speed. It should be noted that the wavenumber is a vector specifying the direction of arrival (*DOA*) of a wavefront as it impinges on the sensor array. For instance, the angle of incidence, $\theta_o$, that the direction vector makes with the array vertical reference, is defined in terms of the wavenumber and sensor locations. For the 2D case, let the horizontal and vertical wavenumbers be defined by $\kappa_\mathbf{o} = [\kappa_x \;\; \kappa_y]$ and the location vector be $\mathbf{r} = [x \;\; y]$. Therefore the complex wavenumber vector has magnitude $|\kappa| = \sqrt{\kappa_x^2 + \kappa_y^2}$, and the angle is $\angle\kappa_\mathbf{o} = [\sin\theta_o \;\; \cos\theta_o]$. Thus in this case the wavenumber vector is defined by its components

$$\kappa_o = [\kappa_x \;\; \kappa_y] = [|\kappa_o|\sin\theta_o \;\; |\kappa_o|\cos\theta_o] \tag{4.227}$$

With this in mind a harmonic plane wave signal vector can be represented by

$$\mathbf{s}(\mathbf{r};t) = \alpha_o e^{j\omega_o t}e^{-j\kappa_o\cdot\mathbf{r}} = \alpha_o e^{j\omega_o t}e^{-j(\kappa_x x+\kappa_y y)} = \alpha_o e^{j(\omega_o t-(|\kappa_o|\sin\theta_o x+|\kappa_o|\cos\theta_o y))} \tag{4.228}$$

We see that in this case the plane wave is characterized by the parameter vector, $\Theta_o = \{\alpha_o, \kappa_o\}$, or equivalently $\Theta_o = \{\alpha_o, \omega_o, \theta_o\}$.

The *model-based wave estimation problem* can succinctly be stated as follows:

> **GIVEN** a set of noisy $L$-element array measurements, $\{\mathbf{y}(t)\}$, $t = 0, \ldots, N-1$, **FIND** the best estimate of the set of wave (source or target) parameters, $\Theta = \{\alpha_m, \omega_m, \theta_m\}$, $m = 1, \ldots, N_s$.

There have been a wide variety of methods applied to this problem that is usually limited to estimating the number of signals (harmonics), $N_s$, as well as the associated arrival angles, $\{\theta_i\}$, $i = 1, \ldots, N_s$ ([48], [49], [50]). In the signal processing literature the problem is called the *direction-of-arrival* (*DOA*) estimation

problem (for plane waves) or spatial localization problem (for spherical waves). Techniques usually assume that the spatiotemporal signal is separable into both spatial, $\mathbf{s}(\theta)$ and temporal, $\mathbf{s}(t)$, parts. *Array signal processing* is involved with processing the multichannel sensor array data to localize source (target) positions.

Here we limit our discussion to the array signal processing approach, applying the *power method* developed for harmonic signals of the previous section. Since this is currently a research area of high interest in signal processing, we refer the reader to the subspace parameter estimation approach discussed in [51] and all of the references therein.

Statistically, assuming the processes measured by the array are zero-mean and *WSS* leads directly to the corresponding measurement spatial covariance matrix

$$R_{yy} = E\{\mathbf{y}(t)\mathbf{y}^H(t)\} = D(\kappa)R_{ss}D^H(\kappa) + R_{\eta\eta} \tag{4.229}$$

for $R_{ss} \in \mathcal{C}^{N_s \times N_s}$, the *signal covariance matrix* and $R_{\eta\eta} \in \mathcal{C}^{L\times L}$, *noise covariance matrix*.

Performing an eigen-decomposition, replacing the wavenumber parameter vector with the angle of incidence, and assuming the measurement noise is white, we write this relation as

$$\begin{aligned} R_{yy} &= D(\theta)R_{ss}D^H(\theta) + \sigma_{\eta\eta}^2 I \\ \text{with } \; d_{\ell n} &= e^{j\omega_o \kappa_\ell \sin\theta_n}, \ell = 1, \ldots, L; n = 1, \ldots, N_s \end{aligned} \tag{4.230}$$

Note that in practice the spatial covariance matrix is usually not available in analytical form. Therefore it must be estimated form the measured sensor array data. The *maximum likelihood* estimator [49] is obtain by averaging over the array temporal snapshots using

$$\hat{R}_{yy} = \frac{1}{N}\sum_{t=1}^{N} \mathbf{y}(t)\mathbf{y}^H(t) \qquad \text{for } \mathbf{y} \in \mathcal{C}^{L\times 1} \tag{4.231}$$

By performing the eigen-decomposition of the estimated spatial covariance matrix, we obtain the identical spectral decomposition of Eq. (4.198) in the spatial domain, that is,

$$R_{yy}(L) = \sum_{i=1}^{N_s} \left(\lambda_i^s + \sigma_{\eta\eta}^2\right)\mathbf{e}_i\mathbf{e}_i^H + \sum_{i=N_s+1}^{L} \sigma_{\eta\eta}^2 \mathbf{e}_i\mathbf{e}_i^H \tag{4.232}$$

Therefore the signal and noise subspaces are defined (as before)

$$R_{yy}(L) = E(N_s)\Lambda(N_s)E^H(N_s) + E(L-N_s)\Lambda(L-N_s)E^H(L-N_s) \tag{4.233}$$

In the spatial case the "harmonic signal vectors" are the columns of the direction matrix, $D(\theta)$ with the *signal direction vectors* defined by $\{\mathbf{d}(\theta_i)\}, i = 1, \ldots, N_s$,

and $\mathbf{d} \in \mathcal{C}^{L\times 1}$. Thus we have the *orthogonality condition* of the signal and noise subspaces as

$$\mathbf{d}(\theta_i) \perp \mathbf{e}_j = \mathbf{d}^H(\theta_i)\mathbf{e}_j = 0, \qquad i = 1, \ldots, N_s;\ j = N_s + 1, \ldots, N \tag{4.234}$$

Expanding over $i$, we obtain

$$D^H(\theta)\mathbf{e}_j = 0, \qquad j = N_s + 1, \ldots, L \tag{4.235}$$

The $L = 3$, $N_s = 2$ case is shown in Figure 4.22 (with $\Theta \to \Omega$) demonstrating the decomposition. Using the *power method* [46], we can generate the *MUSIC* pseudospectrum

$$P_{\text{MUSIC}}(\theta_i) := \frac{1}{\mathbf{d}^H(\theta_i)\left[E(L-N_s)E^H(L-N_s)\right]\mathbf{d}(\theta_i)} = \frac{1}{\sum_{j=N_s+1}^{L} |\mathbf{d}^H(\theta_i)\mathbf{e}_j|^2} \tag{4.236}$$

Using the same arguments as before, we have from Eq. (4.212) that the *EV* pseudospectrum can be generated in a similar manner as

$$\begin{aligned} P_{\text{EV}}(\theta_i) : &= \frac{1}{\mathbf{d}^H(\theta_i)\left[E(L-N_s)\Lambda^{-1}(L-N_s)E^H(L-N_s)\right]\mathbf{d}(\theta_i)} \\ &= \frac{1}{\sum_{j=N_s+1}^{L} \frac{1}{\lambda_j}|\mathbf{d}^H(\theta_i)\mathbf{e}_j|^2} \end{aligned} \tag{4.237}$$

along with any of the other eigen-based methods such as the *MNORM* of Eq. (4.238) given by

$$P_{\min}(\theta_i) := \frac{1}{|\mathbf{d}^H(\theta_i)\mathbf{a}_{\min}|^2} \qquad \text{for } \mathbf{a}_{\min} = P_N\mathbf{n} \tag{4.238}$$

Of course, the *DOA* estimates are obtained by locating the pseudo spectral peaks. We summarize the wave models in the usual framework as follows:

Criterion: $J = \mathbf{w}'\mathbf{R}_{yy}\mathbf{w}$
Models:
- Measurement: $y(t) = D(\theta)s(t) + \eta(t)$
- Signal: $s(r;t) = \sum_{i=1}^{N_s} A_i e^{j\omega_i t - \kappa\cdot\mathbf{r}}$
- Noise: $R_{\eta\eta}$

Algorithm: $P(\Theta_i) = 1/\sum_{j=N_s+1}^{L} |\mathbf{d}^H(\theta_i)\mathbf{e}_j|^2$
Quality: $R_{ss}$

Consider the following example demonstrating the spatial power method.

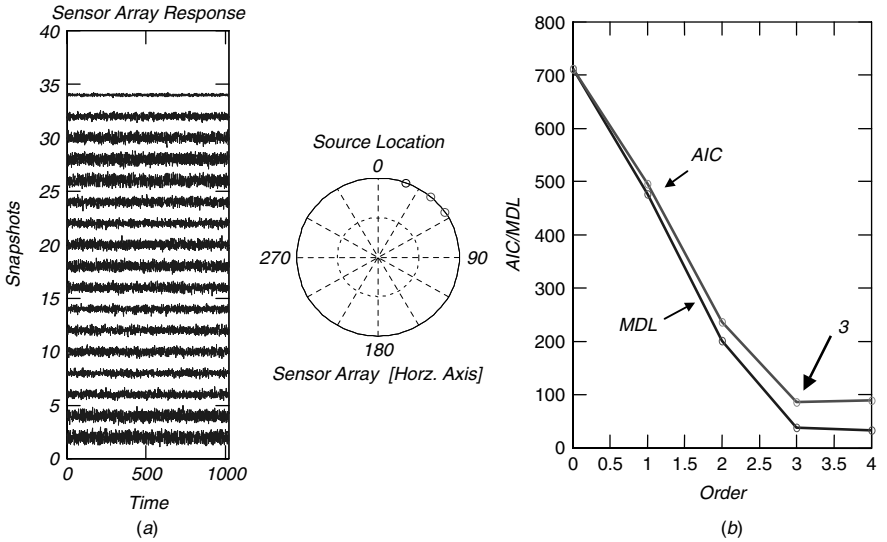



**Figure 4.24.** Plane wave simulation at 20°, 40°, and 55° arrival angles and 0 dB *SNR* for a 16 element array: (*a*) Noisy received sensor data. (*b*) Number of estimated sources using *AIC* and *MDL* techniques.

***Example 4.16*** Suppose that a 16-element, uniformly spaced at 2.5 m, linear array is impinged upon by a planar wavefront generated from three sources ($N_s = 3$) emanating from 20°, 40°, and 55° incidence angles. The temporal frequency is $\omega_o = 300$ Hz with corresponding propagation speed of $c = 1500$ m/s. The sensor data are generated at 0 dB *SNR* and shown in Figure 4.24*a*. The order or number of sources are estimated using the *AIC* and *MDL* techniques giving an $\hat{N}_s = 3$ with the corresponding plots shown in Figure 4.24*b*. The results of applying the *MUSIC* and *MNORM* power estimators are shown in Figure 4.25*a* and 4.25*b*, where we observe the ensemble results of the 16 sensor channels and the corresponding mean and median spectral estimates. The results demonstrate that the algorithms are capable of providing reasonable *DOA* estimates. This completes the example.

So we see that the pseudospectral estimation techniques developed for temporal harmonic model-based parametric estimation map over to the wave-type problems directly with the temporal harmonic signal vectors replaced by the spatial direction signal vectors.

These techniques can be generalized even further by replacing the direction vectors with "matching" vectors generated by a forward propagation model. The technique which has evolved from the work of Hinich [53] and Bucker [54] and is called model-based *matched-field processing* (*MBMFP*). Here a set of (passive) sensor array measurements sampling the field are compared to an equivalent set generated by a forward propagation model producing the corresponding matching (field) vector under an assumed source position. The vectors are compared by producing a statistic which generates the corresponding pixel value (in 2D).

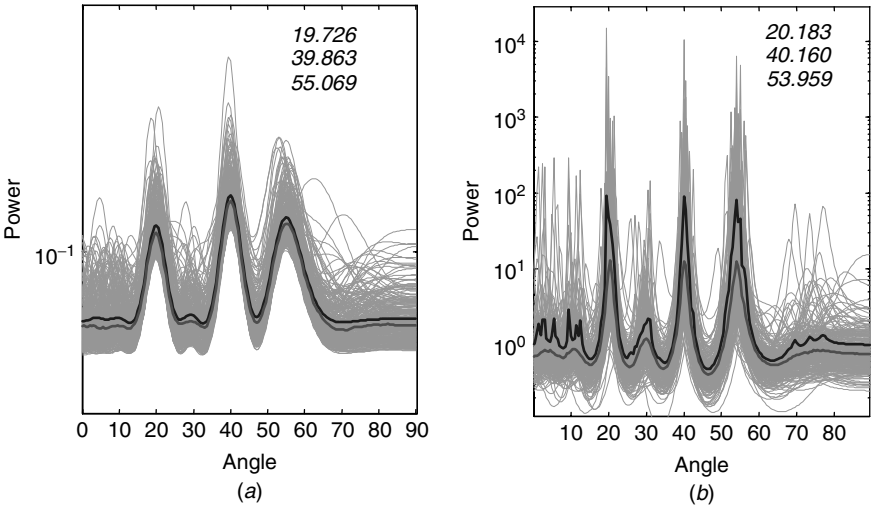



**Figure 4.25.** Plane wave *DOA* estimation: (*a*) *MUSIC* estimates (19.7°, 39.9°, 55.1°). (*b*) *MNORM* estimates (20.2°, 40.2°, 54.0°).

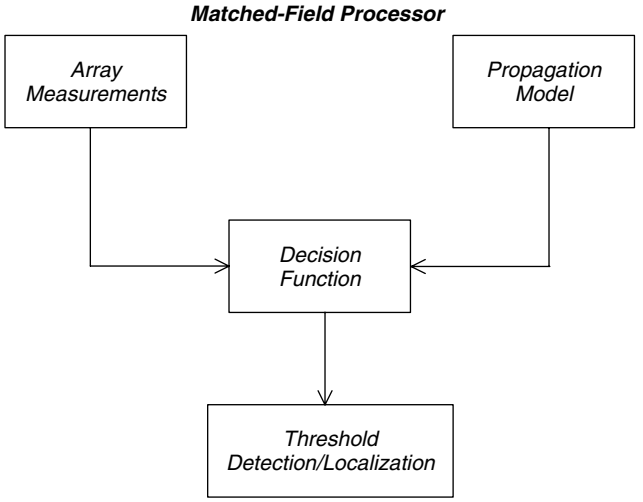



**Figure 4.26.** Model-based matched-field processing (*MBMFP*) approach to the *DOA* estimation problem.

Localization can be performed by thresholding the image and locating its peaks. The approach is depicted in Figure 4.26.

More formally, model-based matched-field processing (*MBMFP*) evolves as the solution to a hypothesis-testing problem [45]. The *MBMFP* problem can be stated as follows:

**GIVEN** the set of noisy field (array) measurements, $\{\mathbf{y}(t)\}$. **DETECT** the presence of a source (target) and **FIND** the best estimate of the set of parameters, $\Theta = \{\alpha_m, \omega_m, \theta_m\}, m = 1, \ldots, N_s$.

The usual application of this approach is to find the location of the source or direction which is represented by the *DOA* or coordinate position, $\theta_i$. A particular solution to this problem is based on testing the following hypotheses:

$$\begin{aligned} \mathcal{H}_o: &\quad \mathbf{y}(t) = \mathbf{n}(t) && \text{(Noise)} \\ \mathcal{H}_1: &\quad \mathbf{y}(t) = \mathbf{m}(t;\Theta) + \mathbf{n}(t) && \text{(Source + noise)} \end{aligned} \tag{4.239}$$

where $\mathbf{m}(t;\theta), \mathbf{n}(t) \in \mathcal{C}^{L\times 1}$ are the respective matching vector and noise vector assumed white, gaussian, that is, $\mathbf{n} \sim \mathcal{N}(0, \Sigma_n)$. The unknown parameters are represented by $\Theta$. The usual approach to the hypothesis testing problem is to apply the Neyman-Pearson theorem [45] and calculate the *likelihood ratio*, that is,

$$\begin{aligned} \mathcal{L}(\Theta) &= \frac{\Pr(\mathbf{y}(t)|\Theta, \mathcal{H}_1)}{\Pr(\mathbf{y}(t)|\Theta, \mathcal{H}_o)} \\ &= \frac{\exp\left(-\frac{1}{2}\left(\mathbf{y}(t) - \mathbf{m}(t;\Theta)\right)' \Sigma_n^{-1}\left(\mathbf{y} - \mathbf{m}(t;\Theta)\right)\right)}{\exp\left(-\frac{1}{2}\mathbf{y}'(t)\Sigma_n^{-1}\mathbf{y}(t)\right)} \underset{\mathcal{H}_o}{\overset{\mathcal{H}_1}{\gtrless}} \tau_\theta \end{aligned} \tag{4.240}$$

where the matching vector, $\mathbf{m}$ is considered deterministic, but unknown. If we take the natural logarithm of both sides, we obtain the *loglikelihood ratio*

$$\begin{aligned} \Lambda(\Theta) :&= \ln \mathcal{L}(\Theta) = \ln \Pr(\mathbf{y}(t)|\Theta, \mathcal{H}_1) - \ln \Pr(\mathbf{y}(t)|\Theta, \mathcal{H}_o) \underset{\mathcal{H}_o}{\overset{\mathcal{H}_1}{\gtrless}} \ln \tau_\theta \\ &= -\frac{1}{2}\left(\mathbf{y}(t) - \mathbf{m}(t;\Theta)\right)' \Sigma^{-1}\left(\mathbf{y} - \mathbf{m}(t;\Theta)\right) + \frac{1}{2}\mathbf{y}'(t)\Sigma^{-1}\mathbf{y}(t) \end{aligned} \tag{4.241}$$

The problem here is that the parameter vector, $\Theta$, is unknown. Therefore the hypothesis test above is a *composite* hypothesis, and the parameter vector must be estimated before a decision can be made [45].

Thus the solution to this problem is to estimate the parameter vector, $\hat{\Theta}$, and then calculate the loglikelihood to perform the test. This is called the *generalized likelihood ratio test* (*GLRT*) and is defined by Eq. (4.241) above. The *GLRT* is

$$\max_{\Theta} \Lambda(\Theta) = \max_{\Theta}\left[\ln \Pr(\mathbf{y}(t)|\Theta, \mathcal{H}_1) - \ln \Pr(\mathbf{y}(t)|\Theta, \mathcal{H}_o)\right] \tag{4.242}$$

If we further assume that the noise of Eq. 4.239 is uncorrelated, so that $\Sigma = \sigma_n^2\mathbf{I}$, then expanding the loglikelihood and performing the indicated operations gives

$$\Lambda(\Theta) = \frac{1}{2}\mathbf{y}'(t)\Sigma_n^{-1}\mathbf{y}(t) - \frac{1}{2}\left[\mathbf{y}'(t)\Sigma_n^{-1}\mathbf{y}(t) - 2\mathbf{m}'(t;\Theta)\Sigma_n^{-1}\mathbf{y}(t) + \mathbf{m}'(t;\Theta)\Sigma_n^{-1}\mathbf{m}(t;\Theta)\right] \underset{\mathcal{H}_o}{\overset{\mathcal{H}_1}{\gtrless}} \ln\tau_\theta$$

or moving all of the "knowns" into the threshold gives the resulting decision function

$$\Lambda(\Theta) = \mathbf{m}'(t;\Theta)\Sigma_n^{-1}\mathbf{y}(t) \underset{\mathcal{H}_o}{\overset{\mathcal{H}_1}{\gtrless}} \ln\tau_\theta + \frac{1}{2}\mathbf{m}'(t;\Theta)\Sigma_n^{-1}\mathbf{m}(t;\Theta) := T_\Theta \tag{4.243}$$

It can be shown ([55], [56]) that Eq. (4.243) is the *matched filter* solution to this multichannel problem, however, instead of the replica being a transmitted signal, it is the *matching vector*, $\mathbf{m}(t;\hat{\Theta})$, generated from a propagation model with estimated parameter vector, $\hat{\Theta}$. Therefore the wave based *MBMFP* has evolved. For example, if we consider the matching vector to be a spherical wavefront, then the solution is simply the spatial filter or beamformer discussed previously. This can be made made more clear if we calculate the maximum output *SNR* power for the following hypothesis test:

$$\max_{\Theta} \quad P(\Theta) = \underset{\mathcal{H}_o}{\overset{\mathcal{H}_1}{\gtrless}} T_\Theta \tag{4.244}$$

where

$$P(\Theta) = E\{\left(\mathbf{m}'(t;\Theta)\Sigma_n^{-1}\mathbf{y}(t)\right)\left(\mathbf{m}'(t;\Theta)\Sigma_n^{-1}\mathbf{y}(t)\right)'\}$$

If we let the additive noise be spatially uncorrelated with unit variance, then $\Sigma_n^{-1} = \mathbf{I}$. The power is given by

$$P(\Theta) = E\{\left(|\mathbf{m}'(t;\Theta)\mathbf{y}(t)|^2\right) = \mathbf{m}'(t;\Theta)R_{yy}\mathbf{m}(t;\Theta) \tag{4.245}$$

which is precisely the conventional *matched-field processor* of Bucker [54]. So we see that at each estimated value of the parameter, $\hat{\Theta}$, a decision is made based on the underlying hypothesis test of Eq. (4.244). From the practical viewpoint a normalized version of this processor is applied as

$$P(\Theta) = \frac{\mathbf{m}'(t;\Theta)R_{yy}\mathbf{m}(t;\Theta)}{\mathbf{m}'(t;\Theta)\mathbf{m}(t;\Theta)} \tag{4.246}$$

For localization of a source or target, the parameter vector is the unknown source position, say, $\Theta = \Theta_s$, and a search over a prespecified grid is performed by estimating the power at each pixel (2D) and creating an image. The image is usually thresholded and the resulting maximum is selected as the source position. Next let us consider an example of active sonar imaging which can be used to demonstrate the *MBMFP* for a set of spherical sources. We summarize the *MBMFP* using *MUSIC* in the model-based framework as follows:

Criterion: $\Lambda(\Theta)$

Models:

Measurement: $\mathbf{y}(t) = \mathbf{m}(t;\Theta) + \mathbf{n}(t)$

Signal: $\mathbf{m}(t;\Theta)$

Noise: $R_{nn}$

Algorithm: $P(\Theta_i) = \dfrac{1}{\sum_{j=N_s+1}^{L} |\mathbf{m}^H(\theta_i)\mathbf{e}_j|^2}$

Quality: $P(\Theta)$

***Example 4.17*** Consider an example of a low frequency, active sonar system searching for targets in the ocean. We assume a simple homogeneous (geometric spreading and time delay) ocean model with three (3) scattering targets in two dimensions (2D). The transmitted pulse, $p(t)$, has a bandwidth of 40 Hz centered at 50 Hz. The receiving array consists of 64 hydrophones spaced 14.8 m apart (spatial Nyquist sampling) with the received temporal signals sampled every 2 ms. The targets are located at xz-positions: (0.2 km, 0.3 km), (0.5 km, 0.9 km), and (0.9 km, 0.5 km) from the array center. The spatiotemporal signals arriving at the array are governed by spherical wave propagation in a homogeneous ocean medium and satisfy

$$s(r_{\ell jk};t) = \frac{1}{|\triangle r_{\ell jk}|}\, p(t-\tau_{\ell jk}) \qquad \text{for } \triangle r_{\ell jk} = |r_\ell - r_{jk}|;\ \tau_{\ell jk} = \frac{|\triangle r_{\ell jk}|}{c}$$

for $\ell$ the array element, $j$ the $x$-position index, and $k$ the $z$-position index. The signals are contaminated by white gaussian noise at a *SNR* of 40dB, that is,

$$\mathbf{y}(t) = \mathbf{s}(\mathbf{r};t) + \mathbf{n}(t)$$

The simulated data are shown in Figure 4.27 along with the transmitted pulse (inset).

The *MBMFP* is implemented in cartesian coordinates with the unknown parameter vector given by

$$\triangle r_{\ell jk} = |r_\ell - r_{jk}| = \sqrt{(x_\ell - x_{jk})^2 + (z_\ell - z_{jk})^2}$$

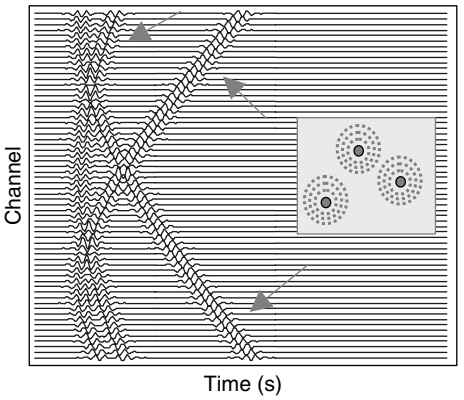



**Figure 4.27.** Active sonar simulation at 40 dB *SNR* for a 64-element hydrophone array: Received sensor data with transmit pulse inset.

The corresponding matching function is therefore

$$m(x_{\ell jk}, z_{\ell jk}; t) = \frac{1}{|\triangle r_{\ell jk}|} p(t - \tau_{\ell jk})$$

with the power at each pixel given by

$$P(x_{\ell jk}, z_{\ell jk}) = \frac{|\mathbf{m}'(\mathbf{r}; t)\mathbf{y}(t)|}{\mathbf{m}'(\mathbf{r}; t)\mathbf{m}(\mathbf{r}; t)} = \frac{\sum_{\ell jk} |m(x_{\ell jk}, z_{\ell jk}; t) y_\ell(t)|^2}{\sum_{\ell jk} |m(x_{\ell jk}, z_{\ell jk}; t)|^2}$$

So we see that the scatterer locations are estimated by the following steps:

1. *Varying* the assumed scatterer positions (location parameter vector).
2. *Calculating* the matching vector.
3. *Calculating* the corresponding power at the specified pixel location. A power image can be generated over desired range of pixels, $j = 1, \ldots, N_x$; $k = 1, \ldots, N_z$.
4. *Thresholding* the image and selecting the dominant peaks

In our problem the resulting scattered power field is shown in Figure 4.28*a* with the thresholded image shown in Figure 4.28*b*. The estimated scatterer positions are (0.207 km,0.297 km), (0.498 km, 0.898 km) and (0.903 km, 0.502 km), which are quite good and can be attributed to the high *SNR*. This completes the example and the section.

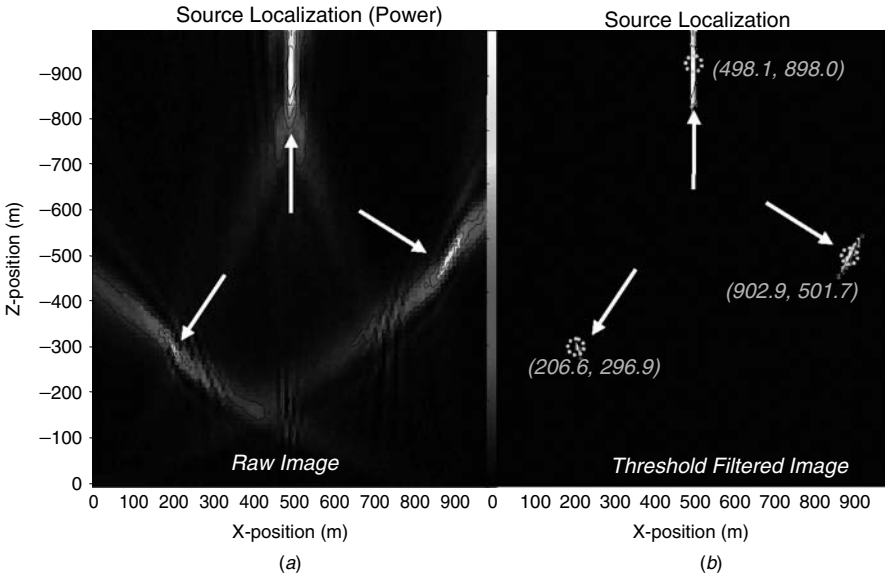



**Figure 4.28.** *MBMFP* localization for active sonar simulation: (*a*) Estimated power image. (*b*) Thresholded power image and peak (location) estimates consisting of (0.207 km, 0.297 km), (0.498 km, 0.898 km), and (0.903 km, 0.502 km).

## 4.10 SUMMARY

In this chapter we discussed the fundamental concepts underlying modern model-based (parametric) signal processing techniques combining the equivalent methods from signal processing, time-series analysis, and system identification and array processing areas. After introducing the basic approach, we developed the all-pole, all-zero, and lattice *MBP*. In each these cases we derived a set of recursions evolving from the Toeplitz structure of the underlying covariance method, while the lattice processor was developed directly from the model structure itself. We demonstrated how a variety of important problems (spectral estimation, deconvolution, time delay estimation, etc.) can be cast into this framework and solved using these model sets. Next we invested the pole-zero *MBP* based on prediction error filter designs and demonstrated its performance on a variety of problems especially in the electromagnetic signal processing case study. We developed the idea of order estimation for these representations and applied directly in the case study to demonstrate its utility. Next we developed the exponential *MBP* and the important special case of harmonic processors. Using the *SVD* approach, we showed how it could be used to efficiently solve the model-based processing problem. Using the *SVD* algorithm led to subspace methods of signal processing and especially the popular *MUSIC* algorithm which was developed along with its variants. We extended these temporal ideas to the spatial domain where we showed how they could be applied to solve the direction-of-arrival and associated localization problem. This led up

to the popular model-based matched-field processing approach, which was derived from the detection theory perspective completing the chapter.

## *MATLAB* NOTES

*MATLAB* and its *Signal Processing Toolbox* can be used to design model-based parametric signal processors. *AR*-models can be estimated from data using a variety of algorithms including the Levinson-Durbin recursion (**levinson**), the linear predictor forward filter (**lpc**), Burg forward-backward approach (**arburg**), the Covariance method (**arcov**), the modified Covariance method (**armcov**), the maximum entropy method (**pmem**), the Yule-Walker approach (**aryule**) and if the *System Identification Toolbox* (*ID*) is available the (**ar**) command has a suite of "approach" flags (burg, yw, ls, etc.). Note also that these algorithms accommodate the lattice forms as well. *ARMAX*-models are available using the Prony technique (**Prony**) and the Steiglitz-McBride optimization algorithm (**stmcb**). In the *ID* toolbox *ARMAX* models can be further estimated in batch mode and well as the *ARX* and general linear models using the Prediction Error Method (**PEM**), which can accommodate multivariable models. A wide suite of models for model-based parameter estimation are available in the *ID* toolbox. Recursive forms can be applied for *ARMAX* (**rarmax**), *ARX* (**rarx**), and the recursive PEM (**rpem**). Order estimation techniques are available (**aic**) and (**fpe**), but other approaches discussed in this chapter are easily programmed in *MATLAB* (e.g., best rank approximation). The transfer function is estimated using the (**tfe**) command which solves the Wiener filter problem in the frequency domain. A wealth of model-based spectral estimators are available to solve for the harmonic temporal as well as spatial models. They are the Burg lattice algorithm (**pburg**), *AR* covariance and modified covariance methods (**pcov**), (**pcovm**) as well as the Yule-Walker approach (**pyulear**). The spatial and temporal eigenvector methods from harmonic estimation are also available such as the MUSIC method (**pmusic**), its eigenvalue weighting variant, (**peig**) as well as their root variants discussed in the chapter (**rootmusic**), (**rooteig**). Classical *PSD* methods are available in the periodogram (**periodogram** and (**pwelch**)), and the multiple taper window method using prolate spheroidal functions (**pmtm**). In fact a very nice interactive cross section of the *PSD* tools are available in the interactive (**sptool**), which is part of the *Signal Processing Toolbox*.

As before, many of the other algorithms mentioned in this chapter can easily be implemented in *MATLAB* using its vector-matrix language.

## REFERENCES

1. J. Markel and A. Gray, *Linear Prediction of Speech*, New York: Springer-Verlag, 1976.
2. L. Rabiner and R. Shafer, *Digital Processing of Speech Signals,* Englewood Cliffs, NJ: Prentice-Hall, 1978.
3. D. Childers, Ed., *Modern Spectral Analysis*, New York: IEEE Press, 1978.

4. G. Box and G. Jenkins, *Time Series Analysis: Forcasting and Control*, San Francisco: Holden-Day, 1976.
5. R. Kalman, "A new approach to linear filtering and prediction problems," *J. Basic Eng.*, **82**, 35–43 1960.
6. K. Astrom and P. Eykhoff, "System identification—A survey," *Automatica*, **7**, 123–162 1971.
7. L. Ljung and T. Soderstrom, *Theory and Practice of Recursive Identification*, Cambridge: MIT Press, 1983.
8. G. Goodwin and K. Sin, *Adaptive Filtering, Prediction and Control* Prentice-Hall, New Jersey, 1984.
9. J. Candy, *Signal Processing: The Model-Based Approach*, New York: McGraw-Hill, 1986.
10. S. Haykin, *Adaptive Filtering Theory*, Englewood Cliffs, NJ: Prentice-Hall, 1996.
11. MathWorks, *MATLAB Users Manual*, Boston: MathWorks Press, 1986.
12. S. Orfanidis, *Optimum Signal Processing*, New York: Macmillan, 1988.
13. R. Wiggins and E. Robinson, "Recursive solution to the multichannel filtering problem," *J. Geophysics, Res.*, **20**, 1885–1891 1965.
14. M. Silvia and E. Robinson, *Deconvolution of Geophysical Time Series in the Exploration of Oil and Natural Gas*, New York: Elsevier, 1979.
15. B. Widrow and S. Sterns, *Adaptive Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1985.
16. A. Meyer and J. Candy, "Iterative processing of ultrasonic measurements to characterize flaws in critical optical components," *IEEE Trans. Ultra. Ferro. Freq. Contr.*, **49** (8), 1124–1137 2002.
17. P. Bogler, *Radar Principles with Applications to Tracking Systems*, New York: Wiley, 1990.
18. W. Hedrick, D. Hykes and D. Starchman, *Ultrasound Physics and Instrumentation*, St. Louis: Mosby, 1995.
19. B. Friedlander, "Lattice filters for adaptive processing," *Proc. IEEE*, **70**, 829–867 1982.
20. M. Honig and D. Messerschmitt, *Adaptive Filters: Structures, Algorithms, and Applications*, Boston: Kluwer, 1984.
21. A. Giordano and F. Hsu, *Least Squares Estimation with Applications to Digital Signal Processing*, New York: Wiley, 1985.
22. H. Akaike, "A new look at the statistical model identification," *IEEE Trans. Autom. Control*, **19**, 716–723 1974.
23. Y. Sakamoto, M. Ishiguro, and G. Kitagawa, *Akaike Information Criterion Statistics*, Boston: Reidel/Kluwer Academic, 1986.
24. J. Candy, M. Warren and T. Bullock, "Realization of invariant system descriptions from infinite Markov sequences," *IEEE Trans. Autom. Contr.*, **23** (1), 93–96 1978.
25. J. Candy, T. Bullock, and M. Warren, "Invariant system description of the stochastic realization," *Automatica*, **15**, 493–495 1979.
26. E. Tse and H. Weinert, "Structure determination and parameter identification for multivariable stochastic linear systems," *IEEE Trans. Autom. Contr.*, **20** (5), 603–613 1975.
27. N. Mohanty, *Random Signals, Estimation, and Identification*, New York: Van Nostrand, 1986.

28. R. Jategaonkar, J. Raol, and S. Balakrishna, "Determination of model order for dynamical systems," *IEEE Trans. Sys., Man, Cyber.*, **12**, 1982.
29. R. Isermann, Ed., "Identification and System Parameter estimation," *Automatica*, **17**, 1–253 1981.
30. J. Candy and J. Zicker, "Electromagnetic signal processing: An estimation/ identification application," *Automatica*, **23** (2), 175–187 1987.
31. R. Bevensee, F. Deadrick, E. Miller, and J. Okada, "Validation and calibration of the LLL transient electromagnetic measurement facility," *LLNL Report*, UCRL-52225, 1977.
32. E. Miller, ed., *Transient Electromagnetic Measurements*, New York: Van Nostrand, 1986.
33. C. Baum, "On the singularity expansion method for the solution of EM interaction problems," *AFWL Interaction Note 88*, 1971.
34. B. Steinberg, *Principles of Aperture and Array System Design*, New York: Wiley, 1976.
35. D. Goodman, "NLS: A system identification package for transient signals," *LLNL Report*, UCID-19767, 1983.
36. S. Tretter, *Introduction to Discrete-Time Signal Processing*, New York: Wiley, 1976.
37. F. Tesche, "On the analysis of scattering and antenna problems using the singularity expansion technique," *IEEE Trans. Ant. Propag.*, **21**, 53–62 1973.
38. S. Kay and S. Marple, "Spectrum analysis-a modern perspective," *Proc. IEEE* , **69** (11), 1380–1419 1981.
39. S. Kay, *Modern Spectral Estimation*, Englewood Cliffs, NJ: Prentice-Hall, 1988.
40. M. Hayes, *Statistical Digital Signal Processing and Modeling*, New York: Wiley, 1996.
41. R. Kumaresan and D. Tufts, "Estimating the parameters of exponentially damped sinusoids and pole-zero modeling in noise," *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. ASSP-30, 6, 833–840 1982.
42. J. Cadzow and M. Wu, "Analysis of transient data in noise," *IEE Proc.*, **134** (Pt. F), 69–78 1987.
43. G. Golub and C. Van Loan, *Matrix Computations*, Baltimore: Johns Hopkins University Press, 1990.
44. J. Cadzow, B. Baseghi, and T. Hsu, "Singular-value decomposition approach to time series modeling," *IEE Proc.*, **130** (Pt. F), 202–210 1983.
45. H. Van Trees, *Optimum Array Processing*, New York: Wiley, 2002.
46. R. Klem, "Use of generalized resolution methods to locate sources in random dispersive media," *IEE Proc.*, **127** (Pt. F), 34–40 1980.
47. V. Pisarenko, "The retrieval of harmonics from a covariance function," *Geophysical J. Royal Astro. Soc.*, **33**, 347–366 1973.
48. R. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE Trans. Ant. Propag.*, **34**, (3) 276–280 1986.
49. D. Johnson and D. Dudgeon, *Array Signal Processing: Concepts and Techniques*, Englewood Cliffs, NJ: Prentice-Hall, 1993.
50. D. Tufts and R. Kumaresan, "Estimation of frequencies of multiple sinusoids: making linear prediction perform like maxiumum likelihood," *IEEE Proc.*, **70** (9) 975–989 1982.

51. H. Krim and M. Viberg, "Two decades of array signal processing research," *IEEE Signal Proc. Mag.*, **13**, 67–94 1996.
52. P. Gill, W. Murray and M. Wright, *Practical Optimization*, New York: Academic Press, 1983.
53. M. Hinich, "Maximum likelihood signal processing for a vertical array," *J. Acoust. Soc. Am.*, **54**, 499–503 1973.
54. H. Bucker, "Use of calculated sound fields and matched-field detection to locate sound in shallow water," *J. Acoust. Soc. Am.*, **59**, 329–337 1976.
55. E. Sullivan and D. Middleton, "Estimation and detection issues in matched-field processing," *IEEE J. Oceanic Eng.*, **18**, 156–167 1993.
56. A. Baggeroer, W. Kuperman and H. Schmidt, "Matched-field processing: source localization in correlated noise as an optimum parameter estimation problem," *J. Acoust. Soc. Am.*, **83**, 571–587 1988.

## PROBLEMS

**4.1** Suppose that we are given a measurement characterized by

$$y(t) = x(t) + n(t)$$

where $x$ is exponentially correlated $R_{xx}(k) = (\frac{1}{2})^{|k|}$ and $n$ is zero-mean, white noise with variance $R_{nn} = 5$.

(**a**) Determine the optimal realizable Wiener filter.

(**b**) Determine the optimal realizable Wiener filter for one-step prediction, that is, $x(t+1|Y_t)$.

(**c**) Simulate the process and obtain the Wiener solution in both time and frequency domain.

**4.2** We are asked to design a Wiener filter to estimate a random signal with covariance $R_{ss}(k) = (\frac{1}{2})^{|k|}$ from a noisy measurement contaminated with unit variance white noise.

(**a**) Determine the optimal Wiener "noncausal" solution.

(**b**) Determine the causal Wiener solution.

**4.3** Suppose that a random signal $s$ is zero-mean with covariance

$$R_{ss}(k) = a\ \exp(-b|k|)$$

and is to be estimated using an instrument with the following model:

$$y(t) = cs(t) + n(t)$$

with $n$ zero-mean and white with variance $R_{nn}$, $a = b = 1$, $c = 2$, and $R_n = 4$, then

(**a**) Find the noncausal Wiener solution $K_{NC}(z)$ and $k_{NC}(t)$.

(**b**) Find the causal Wiener solution $K(z)$ and $k(t)$.

(**c**) Verify the results using the whitening filter approach.

**4.4** Given the covariance of $R_{yy}(k) = (\frac{1}{2})^{|k|}$.

(**a**) Calculate the second-order all-pole Wiener filter. Explain the result.

(**b**) Assume that $R_{yy}(k) = G(\frac{1}{2})^{|k|}$. What effect does $G$ have on the parameter estimates in part $(a)$?

(**c**) Explain how to estimate the gain $G$ in part $(b)$, and in general.

**4.5** Suppose that we have the autoregressive model

$$y(t) = -a_1 y(t-1) - a_2 y(t-2) + e(t)$$

where $\{e(t)\}$ is white.

(**a**) Derive $\hat{y}(t|t-1)$.

(**b**) Derive $\hat{y}(t|t-2)$.

**4.6** Suppose that we have the moving average model

$$y(t) = e(t) + c_1 e(t-1)$$

where $\{e(t)\}$ is white.

(**a**) Derive $\hat{y}(t|t-1)$.

(**b**) Derive $\hat{y}(t|t-2)$.

**4.7** We are given a sequence $\{y(t)\}$ generated by passing white noise with variance $R_{ee}$ through a second order all-pole digital filter with transfer function

$$H(z) = \frac{1}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

Find estimates for $\{\hat{a}_1, \hat{a}_2, \hat{R}_{ee}\}$ in terms of the statistics of $y(t)$ using (i) batch approach, (ii) Levinson-Durbin recursion.

**4.8** Repeat the previous problem using a second-order *FIR* model

$$H(z) = 1 + b_1 z^{-1} + b_2 z^{-2}$$

using the (i) batch approach, (ii) *LWR* recursion.

**4.9** Starting with the Levinson-Durbin recursion

$$a(i+1, i+1) = -k_{i+1} \text{ for } i = 1, \ldots, N$$
$$a(j, i+1) = a(j, i) - k_{i+1} a(i+1-j, 1) \text{ for } j = 1, \ldots, i$$

expand over the noted indexes and take $Z$-transforms of both sides to show:

**(a)** $A_{i+1}(z) = A_i(z) - k_{i+1}z^{-(i+1)}A_i(z^{-1})$
where $A_i(z) = 1 + a(i,1)z^{-1} + \cdots + a(i,i)z^{-i}$.

**(b)** Define the *reverse* polynomial $A_i^R(z) = z^{-i}A_i(z^{-1})$, and using $(a)$, show that

$$A_{i+1}^R(z) = z^{-1}A_i^R(z) - k_{i+1}A_i(z)$$

**(c)** Combine $(a)$ and $(b)$ and show that the predictor polynomials satisfy the recursion

$$\begin{bmatrix} A_{i+1}(z) \\ A_{i+1}^R(z) \end{bmatrix} = \begin{bmatrix} 1 & -k_{i+1}z^{-1} \\ -k_{i+1} & z^{-1} \end{bmatrix} \begin{bmatrix} A_i(z) \\ A_i^R(z) \end{bmatrix}$$

which is initialized with $i = 0$, $A_0(z) = A_0^R(z) = 1$. This is the *Levinson forward recursion* on both forward and backward prediction polynomials.

**(d)** Starting with the *AR* models and the forward recursion, derive the recursion.

**(e)** Suppose that we are given the reflection coefficient sequence $\{k_1 \cdots k_4\} = \{1/2, -1/2, 1/2, -1/2\}$. Find $A_4(z)$.

**(f)** The mean-squared error can also be found recursively using the recursion $J(i+1) = (1 - k_{i+1}^2)J(i)$ with initial conditions $J(0) = R_{yy}(0) = 40.5$. Find $J(4)$.

**(g)** Using the fact that the 4th-order predictor satisfies the relation

$$R_{yy}(i) = -\sum_{j=1}^{i} a(i,j)R_{yy}(i-j)$$

find the corresponding covariances $\{R_{yy}(1) \cdots R_{yy}(4)\}$ using the results of $(e)$ above.

**(h)** The *Levinson reverse recursion* can be derived directly from the forward recursion of the previous exercise. Show that the recursion is given by

$$\begin{bmatrix} A_i(z) \\ A_i^R(z) \end{bmatrix} = \frac{1}{1 - k_{i+1}^2} \begin{bmatrix} 1 & k_{i+1} \\ k_{i+1}z & z \end{bmatrix} \begin{bmatrix} A_{i+1}(z) \\ A_{i+1}^R(z) \end{bmatrix}$$

**(i)** Starting with the *AR* models and the backward recursion derive the recursion.

**(j)** Given the 4th-order predictor and corresponding mean-squared error

$$A_4(z) = 1 - 1.25z{-}1 + 1.3125z^{-2} + 0.5z^{-4}, \quad J(4) = 0.81$$

Find the corresponding reflection coefficients $\{k_1, \ldots, k_4\}$, mean-squared errors, $\{J(0), \ldots, J(3)\}$, and lower order predictors.

**4.10** Show that the following properties of lattice filters hold:

(**a**) Both forward and backward prediction-errors are orthogonal to the input sequence $\{y(t)\}$, that is,

$$E\{e_f(t,i)y(t-j)\} = 0, \qquad 1 \le j \le i$$
$$E\{e_b(t,i)y(t-j)\} = 0, \qquad 0 \le j \le i-1$$

(**b**) The cross-covariance of both forward and backward prediction errors with the input satisfies

$$E\{e_f(t,i)u(t)\} = E\{e_b(t,i)u(t-i)\} = J(i)$$

for $J(i)$ the corresponding mean-squared error (power).

(**c**) The backward prediction errors are *uncorrelated* (white),that is,

$$E\{e_b(t,i)e_b(t,j)\} = \begin{cases} J(i), & i = j \\ 0, & \text{otherwise} \end{cases}$$

(**d**) The forward prediction errors satisfy

$$E\{e_f(t,i)e_f(t,j)\} = J(i), \qquad i \ge j$$

Therefore they are *correlated* (colored).

(**e**) The forward and backward prediction errors are correlated,

$$E\{e_f(t,i)e_b(t,j)\} = \begin{cases} k_j J(i), & i \ge j \\ 0, & i < j \end{cases}$$

**4.11** Using the recursion for the backward prediction errors in terms of the predictor coefficients, that is,

$$e_b(t,i) = \sum_{j=0}^{i} a(i-j,i)y(t-k)$$

show that

(**a**) $\underline{e}_b(t) = L\underline{y}(t)$ with $L$ lower triangular

(**b**) Show that

$$\mathbf{R}_{e_b e_b} = L\mathbf{R}_{yy}L'$$

with $\mathbf{R}_{e_b e_b} = \text{diag}[J(0)\cdots J(i)]$.

(**c**) Calculate the inverse of (*b*) and show that this is equivalent to a Cholesky decomposition

$$\mathbf{R}_{yy}^{-1} = \mathbf{S}'\mathbf{S}$$

with $\mathbf{S} = \mathbf{D}^{-1/2}L$.

**4.12** Simulate the following *AR* system using *MATLAB*

$$(1 + 1.5q^{-1} + 0.5625q^{-2})y(t) = e(t) + \delta(t - 1.2)$$

where $e \sim \mathcal{N}(0, 0.01)$ and $\delta(t - d)$ is a unit impulse function at delay-time $d$ for $\triangle T = 0.1$ s and $N = 64$. Design the following sets of parametric processors:

(**a**) Levinson all-pole filter

(**b**) Levinson all-zero filter

(**c**) Lattice all-pole filter

In each case determine the impulse response of the designed filter and compare it to the true (signal estimate). Also obtain the filtered output $\hat{y}$ and compare to the measured.

**4.13** Simulate the "standard" system using *MATLAB*

$$\begin{aligned} &y(t) - 1.5y(t-1) + 0.7y(t-2) \\ &\quad = u(t-1) + 0.5u(t-2) + e(t) - e(t-1) + 0.2e(t-2) \end{aligned}$$

where $e$ is zero-mean with $R_{ee} = 0.25$, $u$ is pseudorandom binary ($\pm 1$) over 500 samples.

(**a**) Develop the *RELS* estimator for this problem. What are the final parameter estimates?

(**b**) Show the parameter estimate plots.

# 5

# LINEAR STATE-SPACE MODEL-BASED PROCESSORS

## 5.1 STATE-SPACE *MBP* (KALMAN FILTER)

We develop the model-based processors for the dynamic estimation problem, that is, the estimation of processes that vary with time. Here the state-space representation is used as the basic model. We discuss the operation of the filter as a predictor-corrector algorithm.

First, we describe the algorithm and attempt to give some insight into the operations. Then we develop the algorithm. The *MBP*, or equivalently the Kalman filter, can be thought of as an estimator that produces three types of outputs given a noisy measurement sequence and the associated models. The filter is depicted in Figure 5.1. It can be thought of as a *state estimator or reconstructor*; that is, it reconstructs estimates of the state $x(t)$ from noisy measurements $y(t)$. Second, the *MBP* can be thought of as a *measurement filter* that, on input, accepts the noisy sequence $\{y(t)\}$ and, on output, produces a filtered measurement sequence $\{\hat{y}(t|t)\}$. Finally the processor can be thought of as a *whitening filter* that accepts noisy correlated measurements $\{y(t)\}$ and produces uncorrelated or equivalent white measurements $\{e(t)\}$, the innovation sequence. All these properties of the filter have been exploited in many different applications ([1]–[5]). Next we present the algorithm in the predictor-corrector form. We use this form because it provides much insight into the operation of the state-space *MBP*.

*Model-Based Signal Processing*, by James V. Candy

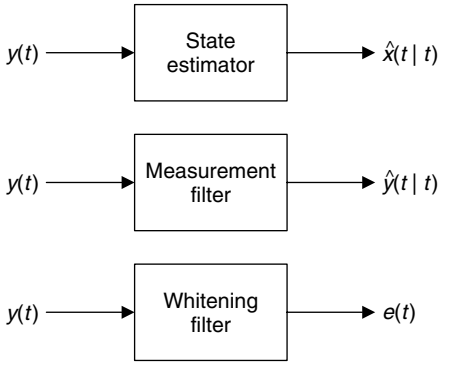


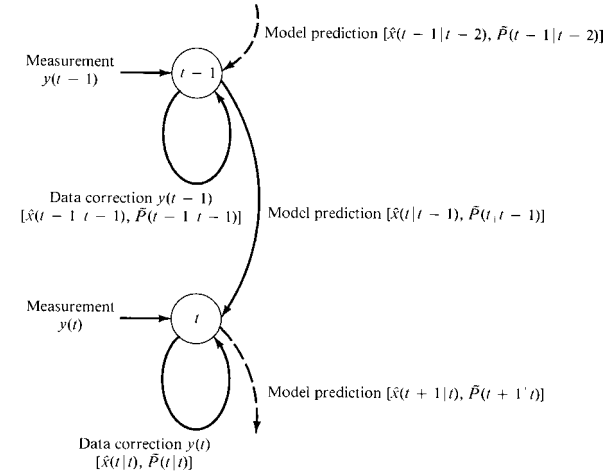

**Figure 5.1.** Various representations of the *MBP* (Kalman filter estimator).

**Table 5.1. State-Space *MBP* (Kalman Filter) Algorithm (Predictor-Corrector Form)**

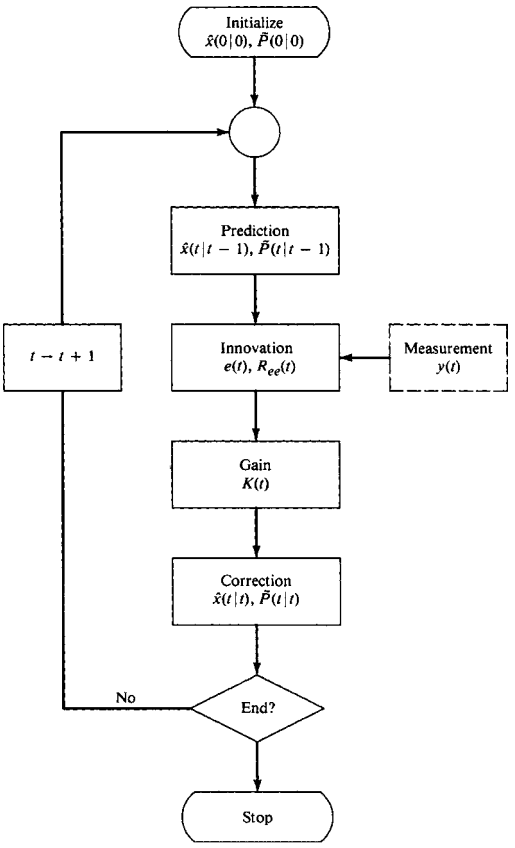
| | |
|---|---|
| Prediction | |
| $\hat{x}(t\|t-1) = A(t-1)\hat{x}(t-1\|t-1) + B(t-1)u(t-1)$ | (State prediction) |
| $\tilde{P}(t\|t-1) = A(t-1)\tilde{P}(t-1\|t-1)A'(t-1) + R_{ww}(t-1)$ | (Covariance prediction) |
| Innovation | |
| $e(t) = y(t) - \hat{y}(t\|t-1) = y(t) - C(t)\hat{x}(t\|t-1)$ | (Innovation) |
| $R_{ee}(t) = C(t)\tilde{P}(t\|t-1)C'(t) + R_{vv}(t)$ | (Innovation covariance) |
| Gain | |
| $K(t) = \tilde{P}(t\|t-1)C'(t)R_{ee}^{-1}(t)$ | (Gain or weight) |
| Correction | |
| $\hat{x}(t\|t) = \hat{x}(t\|t-1) + K(t)e(t)$ | (State correction) |
| $\tilde{P}(t\|t) = [I - K(t)C(t)]\tilde{P}(t\|t-1)$ | (Covariance correction) |
| Initial conditions | |
| $\hat{x}(0\|0), \quad \tilde{P}(0\|0)$ | |

The operation of the *MBP* algorithm can be viewed as a predictor-corrector algorithm as in standard numerical integration. Referring to the algorithm in Table 5.1[1] and Figure 5.2, we see the inherent timing in the algorithm Figure 5.2*a* and the

[1]Note that the preferred numerical technique of implementing this algorithm is the U-D factorization method (see Appendix B for details).




(*a*)




(*b*)

**Figure 5.2.** Predictor-corrector form of the *MBP* or Kalman filter. (*a*) Timing diagram. (*b*) Computational flow diagram.

computational flow in Figure 5.2*b*. First, suppose that we are currently at time $t$ and have not received a measurement, $y(t)$ as yet. We have available to us the previous filtered estimate $\hat{x}(t-1|t-1)$ and error covariance $\tilde{P}(t-1|t-1)$ and would like to obtain the best estimate of the state based on $[t-1]$ data samples. We are in the "prediction phase" of the algorithm. We use the state-space model to predict the state estimate $\hat{x}(t|t-1)$ and its associated error covariance $\tilde{P}(t|t-1)$. Once the prediction based on the model is completed, we then calculate the innovation covariance $R_{ee}(t)$ and gain $K(t)$. As soon as the measurement at time $t$, that is, $y(t)$, becomes available, then we determine the innovation $e(t)$. Now we enter the "correction phase" of the algorithm. Here we correct or update the state based on the new information in the measurement—the innovation. The old, or predicted, state estimate $\hat{x}(t|t-1)$ is used to form the filtered, or corrected, state estimate $\hat{x}(t|t)$ and $\tilde{P}(t|t)$. Here we see that the error, or innovation, is the difference between the actual measurement and the predicted measurement $\hat{y}(t|t-1)$. The innovation is weighted by the gain matrix $K(t)$ to correct the old state estimate (predicted) $\hat{x}(t|t-1)$. The associated error covariance is corrected as well. The algorithm then awaits the next measurement at time $(t+1)$. Observe that in the absence of a measurement, the state-space model is used to perform the prediction, since it provides the best estimate of the state.

The covariance equations can be interpreted in terms of the various signal (state) and noise models (see Table 5.1). The first term of the predicted covariance $\tilde{P}(t|t-1)$ relates to the uncertainty in predicting the state using the model $A$. The second term indicates the increase in error variance due to the contribution of the process noise ($R_{ww}$) or model uncertainty. The corrected covariance equation indicates the predicted error covariance or uncertainty due to the prediction, decreased by the effect of the update ($KC$), thereby producing the corrected error covariance $\tilde{P}(t|t)$. We derive the state-space *MBP* algorithm from the innovations viewpoint in the next section.

## 5.2 INNOVATIONS APPROACH TO THE *MBP*

In this section we derive the *MBP* or equivalently (linear) Kalman filter algorithm from the innovations perspective following the approach by Kailath ([6], [7]). First recall from Chapter 2 that a model of a stochastic process can be characterized by the state-space representation

$$x(t) = A(t-1)x(t-1) + B(t-1)u(t-1) + w(t-1) \tag{5.1}$$

where $w$ is assumed zero-mean and white with covariance $R_{ww}$ and $x$ and $w$ are uncorrelated. The measurement model is given by

$$y(t) = C(t)x(t) + v(t) \tag{5.2}$$

where $v$ is a zero-mean, white sequence with covariance $R_{vv}$ and $v$ is uncorrelated with $x$ and $w$.

The model-based processing problem can be stated in terms of the preceding state-space model. The linear *state estimation problem* is as follows:

> **GIVEN** a set of noisy measurements $\{y(i)\}$, for $i = 1, \ldots, t$, characterized by the measurement model of Eq. (5.2). **FIND** the linear minimum (error) variance estimate of the state characterized by the state-space model of Eq. (5.1). That is, find the best estimate of $x(t)$ given the measurement data up to time $t$, $Y(t) := \{y(1), \ldots, y(t)\}$.

First, we develop the batch minimum variance estimator for this problem using the results derived in Section 3.2. Then we develop an alternative solution using the innovations sequence. The recursive solution follows almost immediately from the innovations. Next we derive equations for the predicted state, gain, and innovations. The corrected, or filtered, state equation then follows.

Constraining the estimator to be linear, we see that for a batch of $N$ data, the minimum variance estimator is given by[2]

$$\underline{\hat{X}}_{MV} = K_{MV}\underline{Y} = R_{xy}R_{yy}^{-1}\underline{Y} \tag{5.3}$$

where $\underline{\hat{X}}_{MV} \in R^{N_x N \times 1}$, $R_{xy}$ and $K_{MV} \in R^{N_x N \times N_y N}$, $R_{yy} \in R^{N_y N \times N_y N}$, and $\underline{Y} \in R^{N_y N \times 1}$. Similarly the linear estimator can be expressed in terms of the $N$ data samples as

$$\underline{\hat{X}}_{MV}(N) = R_{xy}(N)R_{yy}^{-1}(N)\underline{Y}(N) = K_{MV}(N)\underline{Y}(N) \tag{5.4}$$

where $\underline{\hat{X}}_{MV}(N) = [\hat{x}'(1) \cdots \hat{x}'(N)]'$, $\underline{Y}(N) = [y'(1) \cdots y'(N)]'$, $\hat{x} \in R^{N_x \times 1}$, and $y \in R^{N_y \times 1}$.

Here we are investigating a "batch" solution to the state estimation problem, since all the $N_y$-vector data $\{y(1) \cdots y(N)\}$ are processed in one batch. However, we require a recursive solution (see Chapter 3) to this problem of the form

$$\hat{X}_{\text{new}} = \hat{X}_{\text{old}} + K E_{\text{new}} \tag{5.5}$$

In order to achieve the recursive solution, it is necessary to transform the covariance matrix $R_{yy}$ to be block diagonal, since

$$R_{yy}(N) = \begin{bmatrix} E\{y(1)y'(1)\} & \cdots & E\{y(1)y'(N)\} \\ \vdots & & \vdots \\ E\{y(N)y'(1)\} & \cdots & E\{y(N)y'(N)\} \end{bmatrix}$$

$$= \begin{bmatrix} R_{yy}(1,1) & \cdots & R_{yy}(1,N) \\ \vdots & & \vdots \\ R_{yy}(1,N) & \cdots & R_{yy}(N,N) \end{bmatrix}$$

[2]Note that the complete form of the linear minimum variance estimator (see Appendix A) for nonzero mean is given by

$$\underline{\hat{X}}_{MV} = \underline{m}_x + R_{xy}R_{yy}^{-1}(\underline{y} - \underline{m}_y)$$

$R_{yy}$ block diagonal implies that all the off-diagonal block matrices $R_{yy}(t, j) = 0$, for $i$ not equal to $j$, which in turn implies that the $\{y(t)\}$ must be uncorrelated or equivalently orthogonal. Therefore we must construct a sequence of independent $N_y$-vectors, say $\{e(t)\}$, such that

$$E\{e(t)e'(k)\} = 0 \qquad \text{for} \quad t \neq k \tag{5.6}$$

The sequence $\{e(t)\}$ can be constructed using the orthogonality property of the minimum variance estimator:

$$[y(t) - E\{y(t)|Y(t-1)\}] \perp Y(t-1)$$

We define the *innovation* or new information [6] as

$$e(t) := y(t) - \hat{y}(t|t-1) \tag{5.7}$$

with the orthogonality property that

$$\text{cov}\,[y(T), e(t)] = 0 \qquad \text{for} \quad T \leq t-1 \tag{5.8}$$

Since $\{e(t)\}$ is a time-uncorrelated $N_y$-vector sequence, we have

$$R_{ee}(N) = \begin{bmatrix} R_{ee}(1) & & 0 \\ & \ddots & \\ 0 & & R_{ee}(N) \end{bmatrix} \qquad \text{for each} \quad R_{ee}(i) \in R^{N_y \times N_y}$$

The correlated measurement vector can be transformed to an uncorrelated innovation vector through a linear transformation, say $L$, given by

$$\underline{Y}(N) = L\underline{e}(N) \tag{5.9}$$

where $L \in R^{N_y N \times N_y N}$ is a nonsingular transformation matrix and $\underline{e} := [e'(1) \cdots e'(N)]'$. Multiplying Eq. (5.9) by its transpose and taking expected values, we obtain

$$R_{yy}(N) = LR_{ee}(N)L'$$

or inverting, we obtain

$$R_{yy}^{-1}(N) = (L')^{-1}R_{ee}^{-1}(N)L^{-1}$$

Similarly we obtain

$$R_{xy}(N) = R_{xe}(N)L'$$

Substituting these results into Eq. (5.4) gives

$$\underline{\hat{X}}_{MV}(N) = R_{xy}(N)R_{yy}^{-1}(N)\underline{Y}(N) = [R_{xe}(N)L'][(L)^{'-1}R_{ee}^{-1}(N)L^{-1}](L\underline{e}(N))$$

or

$$\underline{\hat{X}}_{MV}(N) = R_{xe}(N)R_{ee}^{-1}(N)\underline{e}(N) \tag{5.10}$$

Since the $\{e(t)\}$ are time-uncorrelated by construction, $R_{ee}(N)$ is block diagonal. From the orthogonality properties of $e(t)$ it can be shown that $R_{xe}(N)$ is lower block triangular; that is,

$$R_{xe}(N) = \begin{cases} R_{xe}(t,i), & t > i \\ R_{xe}(t,t), & t = i \\ 0, & t < i \end{cases}$$

Substituting into Eq. (5.10), we obtain

$$\underline{\hat{X}}_{MV}(N) = \begin{bmatrix} R_{xe}(1,1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ R_{xe}(N,1) & \cdots & R_{xe}(N,N) \end{bmatrix} \begin{bmatrix} R_{ee}^{-1}(1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & R_{ee}^{-1}(N) \end{bmatrix} \begin{bmatrix} e(1) \\ \vdots \\ e(N) \end{bmatrix} \tag{5.11}$$

where $e(i) \in R^{N_y \times 1}$, $R_{xe}(t,i) \in R^{N_x \times N_y}$, and $R_{ee}(i) := R_{ee}(i,i) \in R^{N_y \times N_y}$. The recursive filtered solution follows easily, if we realize that we want the best estimate of $x(t)$ given $Y(t)$. Therefore, any block row of Eq. 5.11 can be written (for $N = t$) as[3]

$$\hat{x}(t|t) = \sum_{i=1}^{t} R_{xe}(t,i)R_{ee}^{-1}(i)e(i)$$

If we extract the last ($t^{\text{th}}$) term out of the sum (recall from Chapter 3), we get

$$\hat{X}_{\text{new}} = \hat{x}(t|t) = \underbrace{\sum_{i=1}^{t-1} R_{xe}(t,i)R_{ee}^{-1}(i)e(i)}_{\hat{X}_{\text{old}}} + \underbrace{R_{xe}(t,t)R_{ee}^{-1}(t)}_{K}\, e(t) \tag{5.12}$$

or

$$\hat{X}_{\text{new}} = \hat{x}(t|t) = \hat{x}(t|t-1) + K(t)e(t) \tag{5.13}$$

where $K(t) = R_{xe}(t)R_{ee}^{-1}(t)$ and $R_{xe}(t,t) = R_{xe}(t)$. So we see that the recursive solution using the innovations sequence instead of the measurement sequence has reduced the computations to inverting a $N_y \times N_y$ matrix $R_{ee}(t)$ instead of a $N_yN \times N_yN$ matrix, $R_{yy}(N)$. Before we develop the expression for the filtered estimate of Eq. (5.13), let us investigate the innovations sequence more closely. Recall that the

[3]Recall that $\hat{x}(t|t) := E\{x(t)|Y(t)\}$ is the state estimate at time $t$ based on all of the data up to and including time $t$.

minimum variance estimate of $y(t)$ is just a linear transformation of the minimum variance estimate of $x(t)$; that is,

$$\hat{y}(t|t-1) = C(t)\hat{x}(t|t-1) \tag{5.14}$$

Thus the innovation can be decomposed using Eqs. (5.2) and (5.14) as

$$e(t) = y(t) - C(t)\hat{x}(t|t-1) = C(t)[x(t) - \hat{x}(t|t-1)] + v(t)$$

or

$$e(t) = C(t)\tilde{x}(t|t-1) + v(t) \tag{5.15}$$

for $\tilde{x}(t|t-1) := x(t) - \hat{x}(t|t-1)$—the predicted *state estimation error*. Consider the innovation covariance $R_{ee}(t)$ using this equation:

$$\begin{aligned} R_{ee}(t) = {} & C(t)E\{\tilde{x}(t|t-1)\tilde{x}'(t|t-1)\}C'(t) + E\{v(t)\tilde{x}'(t|t-1)\}C'(t) \\ & + C(t)E\{\tilde{x}(t|t-1)v'(t)\} + E\{v(t)v'(t)\} \end{aligned}$$

This gives the following, since $v$ and $\tilde{x}$ are uncorrelated:

$$R_{ee}(t) = C(t)\tilde{P}(t|t-1)C'(t) + R_{vv}(t) \tag{5.16}$$

The cross-covariance $R_{xe}$ is obtained using Eq. (5.15) by

$$R_{xe}(t) = E\{x(t)e'(t)\} = E\{x(t)[C(t)\tilde{x}(t|t-1) + v(t)]'\}$$

or

$$R_{xe}(t) = E\{x(t)\tilde{x}'(t|t-1)\}C'(t)$$

Using the definition of the estimation error $\tilde{x}(t)$ and substituting for $x(t)$, we obtain

$$R_{xe}(t) = E\{[\hat{x}(t|t-1) + \tilde{x}(t|t-1)]\tilde{x}'(t|t-1)\}C'(t)$$

or

$$R_{xe}(t) = E\{\hat{x}(t|t-1)\tilde{x}'(t|t-1)\}C'(t) + E\{\tilde{x}(t|t-1)\tilde{x}'(t|t-1)\}C'(t) \tag{5.17}$$

From the orthogonality property of the estimation error for dynamic variables [5], that is,

$$E\{f(Y(T))\tilde{x}'(t|t-1)\} = 0 \quad \text{for} \quad T \leq t-1 \tag{5.18}$$

the first term of Eq. (5.37) is zero because $\hat{x}(t|t-1) = f(Y(t-1))$, giving

$$R_{xe}(t) = \tilde{P}(t|t-1)C'(t) \tag{5.19}$$

Thus we see that the weight, or gain matrix is given by

$$K(t) = R_{xe}(t)R_{ee}^{-1}(t) = \tilde{P}(t|t-1)C'(t)R_{ee}^{-1}(t) \tag{5.20}$$

Before we can calculate the corrected state estimate, we require the predicted, or old estimate; that is,

$$\hat{X}_{\text{old}} = \hat{x}(t|t-1) = E\{x(t)|Y(t-1)\}$$

If we employ the state-space model of Eq. (5.1), then we have from the linearity properties of the conditional expectation

$$\hat{x}(t|t-1) = E\{A(t-1)x(t-1) + B(t-1)u(t-1) + w(t-1)\big|Y(t-1)\}$$

or

$$\hat{x}(t|t-1) = A(t-1)\hat{x}(t-1|t-1) + B(t-1)u(t-1) + \hat{w}(t-1|t-1)$$

However, we have

$$\hat{w}(t|T) = E\{w(t)|Y(T)\} = 0 \qquad \text{for } t \leq T$$

which is not surprising, since the best estimate of zero-mean, white noise is zero (unpredictable). Thus the prediction is given by

$$\hat{X}_{\text{old}} = \hat{x}(t|t-1) = A(t-1)\hat{x}(t-1|t-1) + B(t-1)u(t-1) \tag{5.21}$$

To complete the algorithm, the expressions for the predicted and corrected error covariances must be determined. From the definition of predicted estimation error covariance, we see that $\tilde{x}(t|t-1)$ satisfies

$$\begin{aligned}\tilde{x}(t|t-1) = {} & [A(t-1)x(t-1) + B(t-1)u(t-1) + w(t-1)] \\ & - [A(t-1)\hat{x}(t-1|t-1) + B(t-1)u(t-1)]\end{aligned} \tag{5.22}$$

or

$$\tilde{x}(t|t-1) = A(t-1)\tilde{x}(t-1|t-1) + w(t-1) \tag{5.23}$$

The predicted error covariance $\tilde{P}(t|t-1) := \text{cov}\,(\tilde{x}(t|t-1))$ is

$$\begin{aligned}\tilde{P}(t|t-1) = {} & A(t-1)E\{\tilde{x}(t-1|t-1)\tilde{x}'(t-1|t-1)\}A'(t-1) \\ & + E\{w(t-1)\tilde{x}'(t-1|t-1)\}A'(t-1) \\ & + A(t-1)E\{\tilde{x}(t-1|t-1)w'(t-1)\} \\ & + E\{w(t-1)w'(t-1)\}\end{aligned}$$

However, since $w$ and $\tilde{x}$ are uncorrelated, we have

$$\tilde{P}(t|t-1) = A(t-1)\tilde{P}(t-1|t-1)A'(t-1) + R_{ww}(t-1) \tag{5.24}$$

The corrected error covariance $\tilde{P}(t|t)$ is calculated using the corrected *state-estimation error* and the corresponding state estimate of Eq. (5.13) as

$$\tilde{x}(t|t) := x(t) - \hat{x}(t|t) = x(t) - \hat{x}(t|t-1) - K(t)e(t)$$

or

$$\tilde{x}(t|t) = \tilde{x}(t|t-1) - K(t)e(t) \tag{5.25}$$

From this expression we calculate the required error covariance as

$$\begin{aligned}\tilde{P}(t|t) = \tilde{P}(t|t-1) &- K(t)E\{e(t)\tilde{x}'(t|t-1)\} \\ &- E\{\tilde{x}(t|t-1)e'(t)\}K'(t) + K(t)R_{ee}(t)K'(t)\end{aligned} \tag{5.26}$$

However, from the orthogonality property, $E\{\hat{x}(t|t-1)e'(t)\} = 0$, and Eq. (5.19), we have

$$\begin{aligned}\tilde{P}(t|t) = \tilde{P}(t|t-1) &- K(t)C(t)\tilde{P}(t|t-1) \\ &- \tilde{P}(t|t-1)C'(t)K'(t) + K(t)R_{ee}(t)K'(t)\end{aligned} \tag{5.27}$$

Factoring $\tilde{P}(t|t-1)$ from the first two terms and substituting the expression for the gain of Eq. (5.20) for the first gain in the last term gives

$$\begin{aligned}\tilde{P}(t|t) = [I - K(t)C(t)]\,\tilde{P}(t|t-1) \\ - \tilde{P}(t|t-1)C'(t)K'(t) + \tilde{P}(t|t-1)C'(t)R_{ee}^{-1}(t)R_{ee}(t)K'(t)\end{aligned} \tag{5.28}$$

yielding the final expression for the corrected error covariance as

$$\tilde{P}(t|t) = [I - K(t)C(t)]\,\tilde{P}(t|t-1) \tag{5.29}$$

This completes the derivation, we summarize the *state-space model-based processor* or equivalently the *Kalman filter* as follows:

*Criterion*: $J(t|t) = \text{trace}\ \tilde{P}(t|t)$

*Models*:

Signal: $x(t) = A(t-1)x(t-1) + B(t-1)u(t-1) + w(t-1)$

Measurement: $y(t) = C(t)x(t) + v(t)$

Noise: $w$ and $v$ are zero-mean and white with covariances $R_{ww}$ and $R_{vv}$

Initial state: $x(0)$ has mean $\hat{x}(0|0)$ and covariance $\tilde{P}(0|0)$

*Algorithm MBP* (Kalman filter): $\hat{x}(t|t) = \hat{x}(t|t-1) + K(t)e(t)$
*Quality*: $\tilde{P}(t|t) = [I - K(t)C(t)]\tilde{P}(t|t-1)$

This completes the derivation of the *MBP*[4] estimator based on the innovations sequence. It is clear to see from this derivation that the innovations sequence holds the "key" to unlocking the mystery of *MBP* (Kalman filter) design. In the following section we investigate the statistical properties of the innovation sequence that will enable us to develop a procedure to "tune" the model-based processor.

## 5.3 INNOVATIONS SEQUENCE OF THE *MBP*

In this section we investigate the properties of the innovations sequence that were used to develop the state-space *MBP*. It is interesting to note that since the innovations sequence depends directly on the measurement and is linearly related to it, then it spans the measurement space in which all of our data resides. In contrast, the states or internal (hidden) variables are usually *not* measured directly and are usually *not* available. Therefore our designs are accomplished using the innovation sequence along with its statistical properties to assure optimality. We call this design procedure *minimal (error) variance design*.

Recall that the innovations or equivalently the one-step prediction error is given by

$$e(t) = y(t) - \hat{y}(t|t-1) = \tilde{y}(t|t-1) + v(t) \tag{5.30}$$

where we define $\tilde{y}(t|t-1) := C(t)\tilde{x}(t|t-1)$. Using these expressions, we can now analyze the statistical properties of the innovations sequence based on its orthogonality to the measured data. We will state the property first and then prove it.

The innovation sequence is *zero mean*:

$$E\{e(t)\} = E\{\tilde{y}(t|t-1) + v(t)\} = E\{\tilde{y}(t|t-1)\} + E\{v(t)\} = 0 \tag{5.31}$$

Clearly, the second term is null by the definition of the Gauss-Markov model. Now the first term is null based on the fact that $\hat{x}(t|t-1)$ is an unbiased estimator. This follows since

$$E\{\tilde{y}(t|t-1)\} = C(t)E\{\tilde{x}(t|t-1)\} = C(t)E\{x(t) - \hat{x}(t|t-1)\}$$

the unbiasedness of the estimate, that is,

$$E\{\hat{x}(t|t)\} = E\{\hat{x}(t|t-1)\} = E\{x(t)\}$$

[4]Note that if the Gauss-Markov model is time invariant with stationary noise processes, then the gain converges to "steady state"; that is, the gain matrix becomes a constant resulting in the *steady-state (Kalman) filter*.

which proves the assertion. The innovation sequence is *white* because

$$R_{ee}(t,k) = E\{e(t)e'(k)\} = E\{\tilde{y}(t|t-1)\tilde{y}'(t|t-1)\} + E\{v(t)\tilde{y}'(t|t-1)\} + E\{\tilde{y}(t|t-1)v'(t) + E\{v(t)v'(t)\} \quad (5.32)$$

For $t \neq k$, we have from the orthogonality property of $\tilde{y}(t|t-1)$ and $Y(t-1)$ and the whiteness of $v$ that $R_{ee}(t,k) = 0$. As before, from Eq. (5.16) the innovation sequence is *white* by construction, since

$$R_{ee}(t,k) = \begin{cases} C(t)\tilde{P}(t|t-1)C'(t) + R_{vv}(t), & t = k \\ 0, & t \neq k \end{cases}$$

or

$$\text{cov}\,(e(t), e(k)) = R_{ee}(t,t)\delta(t-k)$$

The innovation sequence is also *uncorrelated with the deterministic input* $u(t-1)$ of Eq. (5.1), since

$$R_{eu}(t) = E\{e(t)u'(t-1)\} = E\{\tilde{y}(t|t-1) + v(t)\}u(t-1) = 0$$

which follows from Eq. (5.31). Assuming that the measurement evolves from a Gauss-Markov process as well, then the innovation sequence is merely a linear transformation of gaussian vectors and are therefore *gaussian* with $e \sim \mathcal{N}(0, R_{ee}(t))$.

Finally, recall that the innovation sequence is related to the measurement by an invertible linear transformation. Therefore it is an *equivalent sequence under linear transformations*, since either sequence can be constructed from knowledge of the second order statistics of the other.

We summarize these properties of the *innovation sequence* as follows:

1. Innovation sequence $\{e(t)\}$ is *zero-mean*.
2. Innovation sequence $\{e(t)\}$ is *white*.
3. Innovation sequence $\{e(t)\}$ is *uncorrelated* in time and with input $u(t-1)$.
4. Innovation sequence $\{e(t)\}$ is *gaussian* with statistics, $\mathcal{N}(0, R_{ee}(t))$, under the Gauss-Markov assumptions.
5. Innovation $\{e(t)\}$ and measurement $\{y(t)\}$ sequences are *equivalent under linear invertible transformations*.

The innovation sequence spans the measurement or data space, but in the model-based processor design problem we are concerned with the state space. Analogous to the innovation sequence in the output space is the predicted state estimation error $\tilde{x}(t|t-1)$ in the state space. It is clear form the orthogonality condition of the innovation, that is,

$$\text{cov}\,(y(T), e(t)) = 0 \qquad \text{for} \quad T \leq t-1$$

that the estimation error is also orthogonal to $y(T)$, since it follows from Eq. (5.15) that

$$E\{y(T)\tilde{x}'(t|t-1)\}C'(t) + E\{y(T)v'(t)\} = 0 \qquad \text{for} \quad T \leq t-1$$

Therefore we have the *state orthogonality condition*

$$\text{cov}\,(y(T), \tilde{x}(t|t-1)) = 0 \qquad \text{for} \quad T \leq t-1$$

which implies that

$$\text{cov}\,(\hat{x}(t|T), \tilde{x}(t|s)) = 0 \qquad \text{for} \quad T \leq s$$

Note also that the state orthogonality condition leads to the following orthogonal decomposition in the state space as

$$x(t) = \hat{x}(t|t-1) + \tilde{x}(t|t-1) \tag{5.33}$$

which in turn gives the covariance expression

$$P(t) = \hat{P}(t|t-1) + \tilde{P}(t|t-1) \qquad \text{for} \quad \hat{P}(t|t-1) := \text{cov}\,(\hat{x}(t|t-1)) \tag{5.34}$$

Unfortunately, the only time the estimation error can be calculated is if $x(t)$ is *known* a priori, as in a simulation study or in comparing a reference model to an estimated output. Thus it is clear that *orthogonality* plays a key role in understanding the operation of the state-space *MBP* processor or Kalman filter. In fact we may consider the *MBP* algorithm as an "orthogonal decomposer" in the sense that it takes correlated measurements and states and returns uncorrelated innovations and estimation errors (if $x$ is known). Note also that the state space can also be characterized similar to the output space. We obtain identical results if we replace $y(t)$, $e(t)$ with $x(t)$, $\tilde{x}(t|t-1)$ in all of the previous equations and use the relations of Eq. (5.12).

To see this, let us go back to our general minimum variance "formula"; that is,

$$x(t) = \sum_{i=1}^{t} R_{x\tilde{x}}(t,i) R_{\tilde{x}\tilde{x}}^{-1}(i,i) \tilde{x}(i|i-1) \tag{5.35}$$

Now we can easily derive the batch relations equivalent to Eq. (3.97) of Chapter 3 with $y(t)$, $e(t)$ replaced by $x(t)$, $\tilde{x}(t|t-1)$. However, the utility is only of theoretical interest because with knowledge of $x(t)$ an estimate would not be required.

It is interesting to investigate how the optimal estimator performs the orthogonalization. The key is to investigate how the innovations can be determined directly from the state estimates. Using the Gram-Schmidt arguments, as before, we have that

$$\hat{x}(t|t) = \sum_{i=1}^{t} K(t,i)e(i) = \sum_{i=1}^{t-1} K(t,i)e(i) + K(t,t)e(t) \tag{5.36}$$

where $K \in R^{N_x \times N_y}$ and therefore can be considered a transformation from the state to output space. The optimal corrected estimate is given by Eq. (5.13), which can be rewritten in the notation above as

$$\hat{x}(t|t) = \hat{x}(t|t-1) + K(t,t)e(t) \tag{5.37}$$

where $K(t,i) = R_{xe}R_{ee}^{-1}$ and $\hat{x}(t|t-1) = \sum_{i=1}^{t-1} K(t,i)e(i)$.

Using the Gram-Schmidt orthogonalization of Eq. (5.36) with the optimal estimator of Eq. (5.37) to solve the batch problem, and expanding, we have

$$\begin{bmatrix} \hat{x}(1|1) \\ \hat{x}(2|2) \\ \vdots \\ \hat{x}(N|N) \end{bmatrix} = \begin{bmatrix} K(1,1) & & & \\ K(2,1) & K(2,2) & & 0 \\ \vdots & & \ddots & \\ K(N,1) & K(N,2) & \cdots & K(N,N) \end{bmatrix} \begin{bmatrix} \hat{e}(1) \\ \hat{e}(2) \\ \vdots \\ \hat{e}(N) \end{bmatrix} \tag{5.38}$$

which gives

$$\underline{\hat{X}}_{MV}(N) = K_{MV}(N)\underline{e}(N)$$

So we see that all the $K(t,i)$ for $t \neq i$ are contained in the past estimates of Eq. (5.37). That is, examining any block row of Eq. (5.38), we have

$$\hat{x}(t|t) = [K(t,1) \quad K(t,2) \quad \cdots \quad K(t,t-1) \quad 0] \begin{bmatrix} e(1) \\ e(2) \\ \vdots \\ e(t-1) \\ 0 \end{bmatrix} + K(t,t)e(t) \tag{5.39}$$

or

$$\hat{x}(t|t) = \underbrace{|<-------\hat{x}(t|t-1)------->|}_{\text{Past}} + \underbrace{K(t)e(t)}_{\text{New}} \tag{5.40}$$

The $K(t,t)$ or $K(t)$ are the "gains" calculated at each time step. The optimal recursive estimator calculates the minimum variance estimate $\hat{x}(t|t)$ or, equivalently, a block row of the $K_{MV}$ matrix. Summarizing this result, the optimal estimator implicitly calculates the $K(t,i)$ for $t > i$, and that information is aggregated into $\hat{x}(t|t-1)$. Thus the estimator need only determine the block diagonal elements of $K$ or the gains at each time step. Therefore the innovations sequence is determined recursively using the *MBP* or Kalman filter, as a "Gram-Schmidt orthogonal decomposer."

This completes the discussion on the orthogonality properties of the optimal estimator and the theoretical development. Next we investigate the alternate statistical approach to develop the *MBP*.

## 5.4 BAYESIAN APPROACH TO THE *MBP*

In this section we briefly derive the Bayesian approach to model-based signal processing primarily in the interest of completeness and for other applications such as optimal Neyman-Pearson detection theory that use the Bayesian approach to develop likelihood ratio detectors ([8], [9], [10]). Therefore we would like to develop the *maximum a posteriori* (*MAP*) estimate of the state vector under the Gauss-Markov model assumptions (see Chapter 3). That is, we would like the *MAP* estimator for the state estimation problem where the underlying Gauss-Markov model assuming $w(t)$, $v(t)$, and $x(0)$ are gaussian distributed. We know that the corresponding state estimate will also be Gaussian because it is a linear transformation of gaussian variables. The *state estimation problem* is therefore defined by:

> **GIVEN** a set of noisy measurements of Eq. (5.2) specified by $Y(t)$, where $Y(t) := \{y(1), \ldots, y(t)\}$, and a Gauss-Markov model of Eq. (5.1). **FIND** the maximum a posteriori (*MAP*) estimator of the state vector $x(t)$ at time $t$, that is, $\hat{X}_{\text{map}}(t)$ that maximizes the a posteriori probability $\Pr(x(t)|Y(t))$.

Applying Bayes' rule (see Chapter 3) to the a posteriori probability, we have that

$$\Pr(x(t)|Y(t)) = \frac{\Pr(x(t), Y(t))}{\Pr(Y(t))} = \frac{\Pr(x(t), y(t), Y(t-1))}{\Pr(Y(t))} \tag{5.41}$$

We also have from Bayes' rule that the denominator can be expressed as

$$\Pr(Y(t)) = \Pr(y(t), Y(t-1)) = \Pr(y(t)|Y(t-1)) \times \Pr(Y(t-1)) \tag{5.42}$$

Therefore, substituting into the denominator of Eq. (5.41) and expanding the numerator again using Bayes' rule, we obtain

$$\Pr(x(t)|Y(t)) = \frac{\Pr(y(t)|x(t), Y(t-1)) \times \Pr(x(t), Y(t-1))}{\Pr(y(t)|Y(t-1)) \times \Pr(Y(t-1))} \tag{5.43}$$

The first term in the numerator can be simplified based on the Gauss-Markov model of Eq. (5.1). Since knowledge of the given $x(t)$ contains all of the information available about $y(t)$ ($y(t) = C(t)x(t) + v(t)$, $v(t)$ zero-mean, white), this term is equivalent to

$$\Pr(y(t)|x(t), Y(t-1)) = \Pr(y(t)|x(t)) \tag{5.44}$$

The second term in the numerator of Eq. (5.43) can be expanded as

$$\Pr(x(t), Y(t-1)) = \Pr(x(t)|Y(t-1)) \times \Pr(Y(t-1)) \tag{5.45}$$

which upon substituting into Eq. (5.43) and canceling like terms yields the following final expression for the *a posteriori probability:*

$$\Pr(x(t)|Y(t)) = \frac{\Pr(y(t)|x(t)) \times \Pr(x(t)|Y(t-1))}{\Pr(y(t)|Y(t-1))} \tag{5.46}$$

Under the Gauss-Markov model assumptions, we know that each of the conditional distributions can be expressed in terms of the gaussian distribution as follows:

1. $\Pr(y(t)|x(t)) \quad : \mathcal{N}(C(t)x(t), R_{vv}(t))$
2. $\Pr(x(t)|Y(t-1)) \; : \mathcal{N}\left(\hat{x}(t|t-1), \tilde{P}(t|t-1)\right)$
3. $\Pr(y(t)|Y(t-1)) \; : \mathcal{N}(\hat{y}(t|t-1), R_{ee}(t))$

Substituting these probabilities into Eq. (5.46) and combining all constants into a single constant $\kappa$, we obtain

$$\begin{aligned}\Pr(x(t)|Y(t)) = \kappa \times \exp\left[-\frac{1}{2}(y(t) - C(t)x(t))' R_{vv}^{-1}(t)(y(t) - C(t)x(t))\right] \\ \times \exp\left[-\frac{1}{2}(x(t) - \hat{x}(t|t-1))' \tilde{P}^{-1}(t|t-1)(x(t) - \hat{x}(t|t-1))\right] \\ / \exp\left[+\frac{1}{2}(y(t) - \hat{y}(t|t-1))' R_{ee}^{-1}(t)(y(t) - \hat{y}(t|t-1))\right]\end{aligned}$$

Recognizing the measurement noise, state estimation error and innovation vectors, we have simply that the a posteriori probability in terms of the Gauss-Markov model is given by

$$\begin{aligned}\Pr(x(t)|Y(t)) = \kappa \times \exp\left[-\frac{1}{2}v'(t)R_{vv}^{-1}(t)v(t)\right] \\ \times \exp\left[-\frac{1}{2}\tilde{x}'(t|t-1)\tilde{P}^{-1}(t|t-1)\tilde{x}(t|t-1)\right] \\ \times \exp\left[+\frac{1}{2}e'(t)R_{ee}^{-1}(t)e(t)\right]\end{aligned} \tag{5.47}$$

Taking natural logarithms of each side, which will not alter the final *MAP* estimate (see Chapter 3), gives

$$\begin{aligned}\ln \Pr(x(t)|Y(t)) = \ln \kappa - \tfrac{1}{2}v'(t)R_{vv}^{-1}(t)v(t) - \tfrac{1}{2}\tilde{x}'(t|t-1)\tilde{P}^{-1}(t|t-1)\tilde{x}(t|t-1) \\ + \tfrac{1}{2}e'(t)R_{ee}^{-1}(t)e(t)\end{aligned} \tag{5.48}$$

The *MAP* estimate is then obtained by differentiating Eq. (5.48), setting it to zero, and solving, that is,

$$\nabla_x \ln \Pr(x(t)|Y(t))\Big|_{x=\hat{X}_{\text{map}}} = \mathbf{0} \tag{5.49}$$

Using the *chain rule* of the gradient operator (see Chapter 3), we obtain the expression (note that the last term of Eq. 5.48 is not a function of $x(t)$, but just the data)

$$\nabla_x \ln \Pr(x(t)|Y(t)) = C'(t)R_{vv}^{-1}(t)\left[y(t) - C(t)x(t)\right] - \tilde{P}^{-1}(t|t-1)\tilde{x}(t|t-1) \tag{5.50}$$

Setting Eq. (5.50) to zero and solving for $x(t)$ gives the *MAP* estimate

$$\begin{aligned}\hat{X}_{\text{map}}(t) = &\left[C'(t)R_{vv}^{-1}(t)C(t) + \tilde{P}^{-1}(t|t-1)\right]^{-1} \\ &\times \left[\tilde{P}^{-1}(t|t-1)\hat{x}(t|t-1) + C'(t)R_{vv}^{-1}(t)y(t)\right]\end{aligned} \tag{5.51}$$

This relation can be simplified by using a form of the *matrix inversion lemma* [5] defined by the following equation:

$$\left(A + BD'\right)^{-1} = A^{-1} - A^{-1}B(I + D'A^{-1}B)^{-1}D'A^{-1} \tag{5.52}$$

Defining the following terms for the lemma, $A = \tilde{P}^{-1}(t|t-1)$, $B = C'(t)R_{vv}^{-1}(t)$, and setting $D' = C(t)$, we find that

$$\begin{aligned}&\left[\tilde{P}^{-1}(t|t-1) + C'(t)R_{vv}^{-1}(t)C(t)\right]^{-1} = \tilde{P}(t|t-1) \\ &\quad - \tilde{P}(t|t-1)C'(t)R_{vv}^{-1}(t)\left(I + C(t)\tilde{P}(t|t-1)C'(t)R_{vv}^{-1}(t)\right)^{-1}C(t)\tilde{P}(t|t-1)\end{aligned} \tag{5.53}$$

Making the observation that the term in parenthesis on the right hand side of Eq. (5.53) can be rewritten by factoring out $R_{vv}^{-1}(t)$ as

$$\left(I + C(t)\tilde{P}(t|t-1)C'(t)R_{vv}^{-1}(t)\right)^{-1} = R_{vv}(t)\left(R_{vv}(t) + C(t)\tilde{P}(t|t-1)C'(t)\right)^{-1} \tag{5.54}$$

Then Eq. (5.54) can also be expressed as

$$\begin{aligned}&\left[\tilde{P}^{-1}(t|t-1) + C'(t)R_{vv}^{-1}(t)C(t)\right]^{-1} = \tilde{P}(t|t-1) \\ &\quad - \tilde{P}(t|t-1)C'(t)\left(R_{vv}(t) + C(t)\tilde{P}(t|t-1)C'(t)\right)^{-1}C(t)\tilde{P}(t|t-1)\end{aligned} \tag{5.55}$$

From Table 5.1 we have the definition of the innovations covariance

$$R_{ee}(t) = C(t)\tilde{P}(t|t-1)C'(t) + R_{vv}(t)$$

Therefore Eq. (5.55) becomes

$$\left[\tilde{P}^{-1}(t|t-1) + C'(t)R_{vv}^{-1}(t)C(t)\right]^{-1} = \tilde{P}(t|t-1)$$
$$- \tilde{P}(t|t-1)C'(t)R_{ee}^{-1}(t)C(t)\tilde{P}(t|t-1) = (I - K(t)C(t))\,\tilde{P}(t|t-1) \quad (5.56)$$

where recall that $K(t) = \tilde{P}(t|t-1)C'(t)R_{ee}^{-1}(t)$ is the *MBP* gain. Referring again to the table, we see that Eq. (5.56) is simply the *corrected error covariance*, $\tilde{P}(t|t)$, equivalent to

$$\tilde{P}(t|t) \equiv \left[\tilde{P}^{-1}(t|t-1) + C'(t)R_{vv}^{-1}(t)C(t)\right]^{-1} \quad (5.57)$$

Thus we can eliminate the first bracketed term in Eq. (5.51) to give

$$\hat{X}_{\text{map}}(t) = \tilde{P}(t|t) \times \left[\tilde{P}^{-1}(t|t-1)\hat{x}(t|t-1) + C'(t)R_{vv}^{-1}(t)y(t)\right]$$

Solving Eq. (5.57) for $\tilde{P}^{-1}(t|t-1)$ and substituting the result gives

$$\hat{X}_{\text{map}}(t) = \tilde{P}(t|t)$$
$$\times \left[\left(\tilde{P}^{-1}(t|t) - C'(t)R_{vv}^{-1}(t)C(t)\right)\hat{x}(t|t-1) + C'(t)R_{vv}^{-1}(t)y(t)\right] \quad (5.58)$$

Multiplying out, regrouping terms and factoring, this relation can be rewritten as

$$\hat{X}_{\text{map}}(t) = \hat{x}(t|t-1) + \left(\tilde{P}(t|t)C'(t)R_{vv}^{-1}(t)\right)\left[y(t) - C(t)\hat{x}(t|t-1)\right] \quad (5.59)$$

or finally

$$\hat{X}_{\text{map}}(t) = \hat{x}(t|t) = \hat{x}(t|t-1) + K(t)e(t) \quad (5.60)$$

Now we only need to show the equivalence of the gain expression using the corrected instead of predicted error covariance, that is,

$$K(t) = \tilde{P}(t|t-1)C'(t)R_{ee}^{-1}(t) = \tilde{P}(t|t)\tilde{P}^{-1}(t|t)\left(\tilde{P}(t|t-1)C'(t)R_{ee}^{-1}(t)\right)$$
$$= \tilde{P}(t|t)\left[C'(t)R_{vv}^{-1}(t)C(t) + \tilde{P}^{-1}(t|t-1)\right]\tilde{P}(t|t-1)C'(t)R_{ee}^{-1}(t)$$
$$= \tilde{P}(t|t)C'(t)R_{vv}^{-1}(t)\left[C(t)\tilde{P}(t|t-1)C'(t) + R_{vv}(t)\right]R_{ee}^{-1}(t) \quad (5.61)$$

where we have solved Eq. (5.57) for $\tilde{P}^{-1}(t|t)$ and substituted giving the desired result from the definition of innovations covariance. Thus we now have two *equivalent* expressions that can be used to calculate the gain

$$K(t) = \tilde{P}(t|t)C'(t)R_{vv}^{-1}(t) \equiv \tilde{P}(t|t-1)C'(t)R_{ee}^{-1}(t) \tag{5.62}$$

which completes the Bayes' approach to *MBP*.

## 5.5 TUNED *MBP*

In this section we heuristically develop a feel for the operation of the model-based processor using the state-space model. These results coupled with the theoretical points developed in the previous section lead to the proper adjustment or "tuning" of the *MBP*. Tuning the processor is considered an art, but with proper statistical tests, the performance can readily be evaluated and adjusted. This approach is called *minimum (error) variance* design. In contrast to standard filter design procedures in signal processing, the minimum variance design adjusts the statistical parameters (e.g., covariances) of the processor and examines the innovations sequence to determine if the *MBP* is properly tuned. Once tuned, then all of the statistics (conditional means and variances) are valid and may be used as valid estimates. Here we discuss how the parameters can be adjusted and what statistical tests must be performed to evaluate *MBP* performance.

Heuristically the *MBP* can be viewed simply by its correction equation

$$\hat{X}_{\text{new}} = \overbrace{\underbrace{\hat{X}_{\text{old}}}_{\text{State-space model}}}^{\text{Prediction}} + \overbrace{\underbrace{K \times E_{\text{new}}}_{\text{Measurement}}}^{\text{Correction}}$$

where $\hat{X}_{\text{old}} \approx f(\text{model})$ and $E_{\text{new}} \approx f(\text{measurement})$.

Using this model of the *MBP*, we see that we can view the old, or predicted estimate $\hat{X}_{\text{old}}$ as a function of the state-space model $(A, B)$ and the prediction error or innovation $E$ as a function primarily of the new measurement, as indicated in Table 5.2. Consider the new estimate under the following cases:

$$\begin{aligned} K \longrightarrow \text{small} \quad & \hat{X}_{\text{new}} = \hat{X}_{\text{old}} \quad = f(\text{model}) \\ K \longrightarrow \text{large} \quad & \hat{X}_{\text{new}} = K E_{\text{new}} = f(\text{measurement}) \end{aligned}$$

So we can see that the operation of the processor is pivoted about the values of the gain or weighting matrix $K$. For small $K$, the processor "believes" the *model*, and for large $K$, the processor believes the *measurement*.

Let us investigate the gain matrix and see if its variations are consistent with these heuristic notions. First, it was shown in Eq. (5.62) that the alternate form of the gain equation is given by

$$K(t) = \tilde{P}(t|t)C'(t)R_{vv}^{-1}(t)$$

**Table 5.2. Model-Based Processor Heuristic Notions**

| Condition | Gain | Parameter |
|---|---|---|
| Believe *model* | Small | $\tilde{P}$ small (model adequate)<br>$R_{vv}$ large (measurement noisy) |
| Believe *measurement* | Large | $\tilde{P}$ large (model inadequate)<br>$R_{vv}$ small (measurement good) |

Thus the condition where $K$ is *small* can occur in two cases: (1) $\tilde{P}$ is small (fixed $R_{vv}$), which is consistent because small $\tilde{P}$ implies that the model is adequate, and (2) $R_{vv}$ is large ($\tilde{P}$ fixed), which is also consistent because large $R_{vv}$ implies that the measurement is noisy, so again believe the model.

For the condition where $K$ is *large*, two cases can also occur: (1) $K$ is large when $\tilde{P}$ is large (fixed $R_{vv}$), implying that the model is inadequate, so believe the measurement; and (2) $R_{vv}$ is small ($\tilde{P}$ fixed), implying the measurement is good (high *SNR*). So we see that our heuristic notions are based on specific theoretical relationships between the parameters in the *MBP* algorithm of Table 5.1. We summarize the heuristic *MBP* operation as follows:

A *MBP* (Kalman filter) is not functioning properly when the *gain* becomes small and the measurements still contain information necessary for the estimates. The filter is said to *diverge* under these conditions. In this case it is necessary to detect how the filter is functioning and how to adjust it if necessary, but first we consider the tuned *MBP*.

When the processor is "tuned," it provides an optimal or minimum (error) variance estimate of the state. The innovations sequence, which was instrumental in deriving the processor, also provides the starting point to check the *MBP* operation. A *necessary and sufficient condition* for a *MBP* to be optimal is that the innovation sequence is zero-mean and white (see Section 5.3). These are the first properties that must be evaluated to insure that the processor is operating properly. If we assume that the innovation sequence is ergodic and gaussian, then we can use the sample mean as the test statistic to estimate $m_e$, the population mean. The sample mean for the $i$th component of $e_i$ is given by

$$\hat{m}_e(i) = \frac{1}{N}\sum_{t=1}^{N} e_i(t) \qquad \text{for} \quad i = 1, \ldots, N_y \tag{5.63}$$

where $\hat{m}_e(i) \sim \mathcal{N}(m_e, R_{ee}(i)/N)$ and $N$ is the number of data samples. We perform a statistical hypothesis test to "decide" if the innovation mean is null [2]. We test that the mean of the $i$th component of the innovation vector $e_i(t)$ is

$$H_0 : m_e(i) = 0$$
$$H_1 : m_e(i) \neq 0$$

As our test statistic we use the sample mean. At the $\alpha$-significance level, the probability of rejecting the null hypothesis $H_0$ is given by

$$\Pr\left(\left|\frac{\hat{m}_e(i) - m_e(i)}{\sqrt{R_{ee}(i)/N}} > \frac{\tau_i - m_e(i)}{\sqrt{R_{ee}(i)/N}}\right|\right) = \alpha \tag{5.64}$$

Therefore the *zero-mean test* [11] on each component innovation $e_i$ is given by

$$\hat{m}_e(i) \begin{array}{c} > \\ \\ < \end{array} \begin{array}{l} \text{Reject } H_0 \\ \tau_i \\ \text{Accept } H_0 \end{array} \tag{5.65}$$

Under the null hypothesis $H_0$, each $m_e(i)$ is assumed zero. Therefore, at the 5% significance level ($\alpha = 0.05$), we have that the threshold is

$$\tau_i = 1.96\sqrt{\frac{\hat{R}_{ee}(i)}{N}} \tag{5.66}$$

where $\hat{R}_{ee}(i)$ is the *sample variance* (assuming ergodicity) estimated by

$$\hat{R}_{ee}(i) = \frac{1}{N}\sum_{t=1}^{N} e_i^2(t) \tag{5.67}$$

Under the same assumptions, we can perform a *whiteness test* [12] (see Example 2.10), that is, check statistically that the innovations covariance corresponds to that of an uncorrelated (white) sequence. Again, assuming ergodicity of the innovations sequence, we use the sample covariance function as our test statistic with the $i$th component covariance given by

$$\hat{R}_{ee}(i,k) = \frac{1}{N}\sum_{t=k+1}^{N} (e_i(t) - \hat{m}_e(i))\,(e_i(t+k) - \hat{m}_e(i)) \tag{5.68}$$

We actually use the *normalized covariance* test statistic

$$\hat{\rho}_{ee}(i,k) = \frac{\hat{R}_{ee}(i,k)}{\hat{R}_{ee}(i)} \tag{5.69}$$

Asymptotically for large $N$ it can be shown that (see [13]) that

$$\hat{\rho}_{ee}(i,k) \sim \mathcal{N}\left(0, \frac{1}{N}\right)$$

Therefore the 95% confidence interval estimate is

$$I_{\rho_{ee}} = \hat{\rho}_{ee}(i,k) \pm \frac{1.96}{\sqrt{N}} \qquad \text{for } N > 30 \tag{5.70}$$

Hence, under the null hypothesis, 95% of the $\hat{\rho}_{ee}(i,k)$ values must lie within this confidence interval to accept $H_0$. That is, for each *component* innovation sequence to be considered statistically white. Similar tests can be constructed for the *cross-covariance* properties of the innovations [13] as well, that is,

$$\text{cov}\,(e(t), e(k)) = 0 \quad \text{and} \quad \text{cov}\,(e(t), u(t-1)) = 0$$

The whiteness test of Eq. (5.70) is very useful to detect modeling inaccuracies from individual component innovations. However, for complex systems with a large number of measurement channels, it becomes computationally burdensome to investigate each innovation componentwise. A statistic containing *all* of the innovation information is the *weighted sum-squared residual* (*WSSR*) [14]. It aggregates all of the innovation *vector* information over some finite window of length $N$. It can be shown that the *WSSR* is related to a maximum-likelihood estimate of the normalized innovations variance [14], [15]. The *WSSR* test statistic is given by

$$\hat{\rho}(\ell) := \sum_{k=\ell-N+1}^{\ell} e'(k) R_{ee}^{-1}(k) e(k) \qquad \text{for} \quad \ell \geq N \tag{5.71}$$

The *WSSR* hypothesis test is based on

$$H_0 : \rho(\ell) \textit{ is } \text{white}$$
$$H_1 : \rho(\ell) \textit{ is not } \text{white}$$

and is given by

$$\hat{\rho}(\ell) \begin{array}{l} > \\ \\ < \end{array} \begin{array}{l} \text{Reject } H_0 \\ \tau \\ \text{Accept } H_0 \end{array} \tag{5.72}$$

Under the null hypothesis, the *WSSR* is chi-squared distributed, $\rho(\ell) \sim \chi^2(N_y N)$. However, for $N_y N > 30$, $\rho(\ell)$ is approximately gaussian $\mathcal{N}(N_y N, 2N_y N)$ (see [1], [16] for more details). At the $\alpha$-significance level, the probability of rejecting the null hypothesis is given by

$$\Pr\left( \left| \frac{\rho(\ell) - N_y N}{\sqrt{2N_y N}} > \frac{\tau - N_y N}{\sqrt{2N_y N}} \right| \right) = \alpha \tag{5.73}$$

For a level of significance of $\alpha = 0.05$, we have

$$\tau = N_y N + 1.96\sqrt{2N_y N} \tag{5.74}$$

So we see that the *WSSR* can be considered a "whiteness test" of the innovations *vector* over a finite window of length $N$. Note that since $[\{e(t)\}, \{R_{ee}(t)\}]$ are obtained

from the state-space *MBP* algorithm directly, they can be used for both *stationary* as well as *nonstationary* processes. In fact, in practice, for a large number of measurement components, the *WSSR* is used to "tune" the filter, and then the component innovations are individually analyzed to detect model mismatches. Note also that the adjustable parameter of the *WSSR* statistic is the window length $N$, which essentially controls the width of the window sliding through the innovations sequence.

Another set of "reasonableness" tests can be performed using the covariances estimated by the *MBP* algorithm and sample variances estimated using Eq. (5.67). The *MBP* provides estimates of the respective processor covariances $R_{ee}$ and $\tilde{P}$ from the relations given in Table 5.1. Using sample variance estimators when the filter reaches steady state (process is stationary), that is, when $\tilde{P}$ is a constant, we can compare the estimates to ensure that they are reasonable. Thus we have

$$\hat{R}_{ee}(i) \approx R_{ee}(i) \quad \text{and} \quad \hat{\tilde{P}}(i) \approx \tilde{P}(i) \tag{5.75}$$

Plotting the $\pm 1.96\sqrt{R_{e_i e_i}(t)}$ and $\pm 1.96\sqrt{\tilde{P}_{ii}(t|t)}$ about the component innovations $\{e_i(t)\}$ and component state estimation errors $\{\tilde{x}_i(t|t)\}$, when the true state is known provides an accurate estimate of the *MBP* performance especially when simulation is used. If the covariance estimates of the processor are reasonable, then 95% of the sequence samples should lie within the constructed bounds. Violation of these bounds clearly indicate inadequacies in modeling the processor statistics. We summarize these results in Table 5.3 and conclude this section with an example of a "tuned" *MBP*.

**Table 5.3. State-Space *MBP* Tuning Tests**

| Data | Property | Statistic | Test | Assumptions |
|---|---|---|---|---|
| Innovation | $m_e = 0$ | Sample mean | Zero mean | Ergodic, gaussian |
| | $R_{ee}(t)$ | Sample covariance | Whiteness | Ergodic, gaussian |
| | $\rho(l)$ | WSSR | Whitness | Gaussian |
| | $R_{ee}(t,k)$ | Sample cross-covariance | Cross-covariance | Ergodic, gaussian |
| | $R_{eu}(t,k)$ | Sample cross-covariance | Cross-covariance | Ergodic, gaussian |
| Covariance | Innovation | Sample variance | $R_{ee} = \tilde{R}_{ee}$ | Ergodic |
| | Innovation | $R_{ee}$ | Confidence interval about $\{e(t)\}$ | |
| | Estimation error | Sample variance | $\tilde{P} = \hat{\tilde{P}}$ | Ergodic, $X_{\text{true}}$ known |
| | Estimation error | $\tilde{P}$ | Confidence interval about $\{\tilde{x}(t\|t)\}$ | $X_{\text{true}}$ known |

***Example 5.1*** Suppose that we have the $RC$ circuit discussed in Chapter 1 (see Figure 1.3). We measure the voltage across the capacitor with a high-impedance voltmeter as shown. Since these measurements are noisy and the component values imprecise ($\pm\triangle$), we require an improved estimate of the output voltage. We develop a *MBP* to solve this problem from first principles—a typical approach. Writing the Kirchoff current equations at the node, we have

$$I_{\text{in}}(t) - \frac{e(t)}{R} - C\frac{de(t)}{dt} = 0$$

where $e_o$ is the initial voltage and $R$ is the resistance with $C$ the capacitance. The measurement equation for a voltmeter of gain $K_e$ is simply

$$e_{out}(t) = K_e e(t)$$

We choose to use the discrete *MBP* formulation; therefore, approximating the derivatives with first differences and substituting for them, we have

$$C\frac{e(t) - e(t-1)}{\triangle T} = -\frac{e(t-1)}{R} + I_{\text{in}}(t-1)$$

or

$$e(t) = \left(1 - \frac{\triangle T}{RC}\right) e(t-1) + \frac{\triangle T}{C} I_{\text{in}}(t-1)$$

where the measurement is given above. Suppose that for this circuit the parameters are $R = 3.3\ k\Omega$ and $C = 1000\ \mu F$, $\triangle T = 100$ ms, $e_o = 2.5\ V$, $K_e = 2.0$, and the voltmeter is precise to within $\pm 4\ V$. Then, transforming the physical circuit model into state-space form by defining $x = e$, $y = e_{\text{out}}$, and $u = I_{\text{in}}$, we obtain

$$x(t) = 0.97x(t-1) + 100u(t-1) + w(t-1)$$
$$y(t) = 2x(t) + v(t)$$

The process noise covariance is used to model the circuit parameter uncertainty with $R_{ww} = 0.0001$, since we assume standard deviations, $\triangle R$, $\triangle C$ of 1%. Also, $R_{vv} = 4$, since two standard deviations are $\triangle V = 2\left(\frac{1}{2}\ 4V\right)$. We also assume initially that the state is $x(0) \sim \mathcal{N}(2.5, 10^{-12})$, and that the input current is a step function of $u(t) = 300\mu A$. *SSPACK_PC* (see Appendix C) is used to simulate this system [17]. The results are shown in Figure 5.3. The simulated and true (mean) states (voltages) are shown in Figure 5.3*a* along with the corresponding confidence limits. We see that the process samples (state and process noise) lie within the bounds (3.5% out). Therefore, the data statistically satisfy the underlying Gauss-Markov model assumptions. If it does not, then choose another seed for the simulation. That is, we perform another realization (different *seed* in random number generator) until the samples lie within the bounds. Similarly the simulated and true (mean) measured voltages are shown in Figure 5.3*b*. Again the

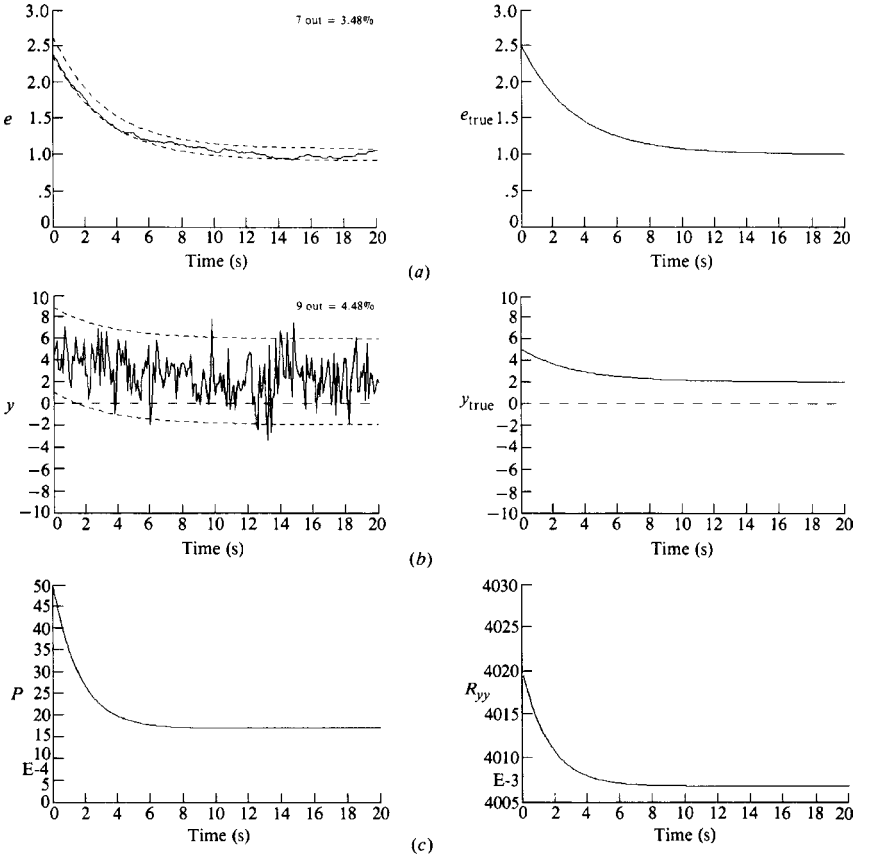



**Figure 5.3.** *RC*-circuit problem Gauss-Markov simulation. (*a*) Simulated and true (mean) output voltage. (*b*) Simulated and true (mean) measurement. (*c*) Simulated state and measurement variances.

data (measurement and noise) statistically satisfy the underlying models with only 4.5% of the samples exceeding the prescribed bounds. The state and measurement variances used to construct the confidence intervals about the means, that is, $\left[m_x(t) \pm 1.96\sqrt{P(t)}\right]$ and $\left[m_y(t) \pm 1.96\sqrt{R_{yy(t)}}\right]$ are shown in Figure 5.3*c*.

With the data simulated, we now consider the design of the *MBP*. In the ideal *MBP* problem, we are given the model set $\Sigma := \{A, B, C, R_{ww}, R_{vv}, x(0), P(0)\}$, the known input $\{u(t)\}$, and the set of noisy measurements, $\{y(t)\}$ to construct the processor. The *RC* model-based processor for this problem can simply be written as

$$\hat{x}(t|t-1) = 0.97\hat{x}(t-1|t-1) + 100u(t-1) \qquad \text{(Predicted state)}$$

$$\tilde{P}(t|t-1) = 0.94\tilde{P}(t-1|t-1) + 0.0001 \qquad \text{(Predicted covariance)}$$

$$e(t) = y(t) - 2\hat{x}(t|t-1) \qquad \text{(Innovation)}$$

$$R_{ee}(t) = 4\tilde{P}(t|t-1) + 4 \qquad \text{(Innovations covariance)}$$

$$K(t) = 2\frac{\tilde{P}(t|t-1)}{4\tilde{P}(t|t-1)+4} \qquad \text{(Gain)}$$

$$\hat{x}(t|t) = \hat{x}(t|t-1) + K(t)e(t) \qquad \text{(Corrected state)}$$

$$\tilde{P}(t|t) = \frac{\tilde{P}(t|t-1)}{\tilde{P}(t|t-1)+1} \qquad \text{(Corrected covariance)}$$

The estimator is also designed using *SSPACK_PC* and the results are shown in Figure 5.4. In Figure 5.4*a* we see the estimated state (voltage) and estimation error as well as the corresponding confidence bounds. Note that the processor "optimally" estimates the voltage, since our models are exact. That is, it provides the minimum error variance estimate in the gaussian case. Also, since we have the true (mean) state, we can calculate the estimation error and use the corresponding error covariance to specify the bounds as shown. Note that the error is small and no samples exceed the bound as indicated by the overestimation of the variance compared with the sample variance ($0.0017 > 0.0002$). In Figure 5.4*b* we see the filtered measurement ($\hat{y}(t|t-1)$) and corresponding innovation sequence as well as the confidence limits provided by the processor. Here we see that only 4.5% of the samples exceed the bounds and that the variance predicted by the filter is close to the sample variance estimate ($4.0 \sim 3.7$). The *WSSR*, zero-mean, and whiteness tests are shown in Figure 5.4*c*. Here we see that using a window of 75 samples, the threshold is not exceeded, indicating a statistically white sequence. The innovation mean is small and well within the bound ($0.11 < 0.27$). The sequence is statistically white, since 0% of the normalized sample covariances exceed the bound. Finally, we see the gain and corrected error covariance as monotonically decreasing functions that reach a steady-state (constant) value at approximately 8 seconds. This completes the example. We summarize the results as follows:

Criterion: $J = \text{trace}\ \ \tilde{P}(t|t)$

Models:

Signal: $x(t) = 0.97x(t-1) + 100u(t-1) + w(t-1)$

Measurement: $y(t) = 2x(t) + v(t)$

Noise: $w \sim \mathcal{N}(0, 10^{-12})$ and $v \sim \mathcal{N}(0, 4)$

Algorithm: $\hat{x}(t|t) = \hat{x}(t|t-1) + K(t)e(t)$

Quality: $\tilde{P}(t|t) = \dfrac{\tilde{P}(t|t-1)}{\tilde{P}(t|t-1)+1}$

This completes the example, next we investigate circumstances where the processor is *not* tuned or the models used do *not* match the statistics.

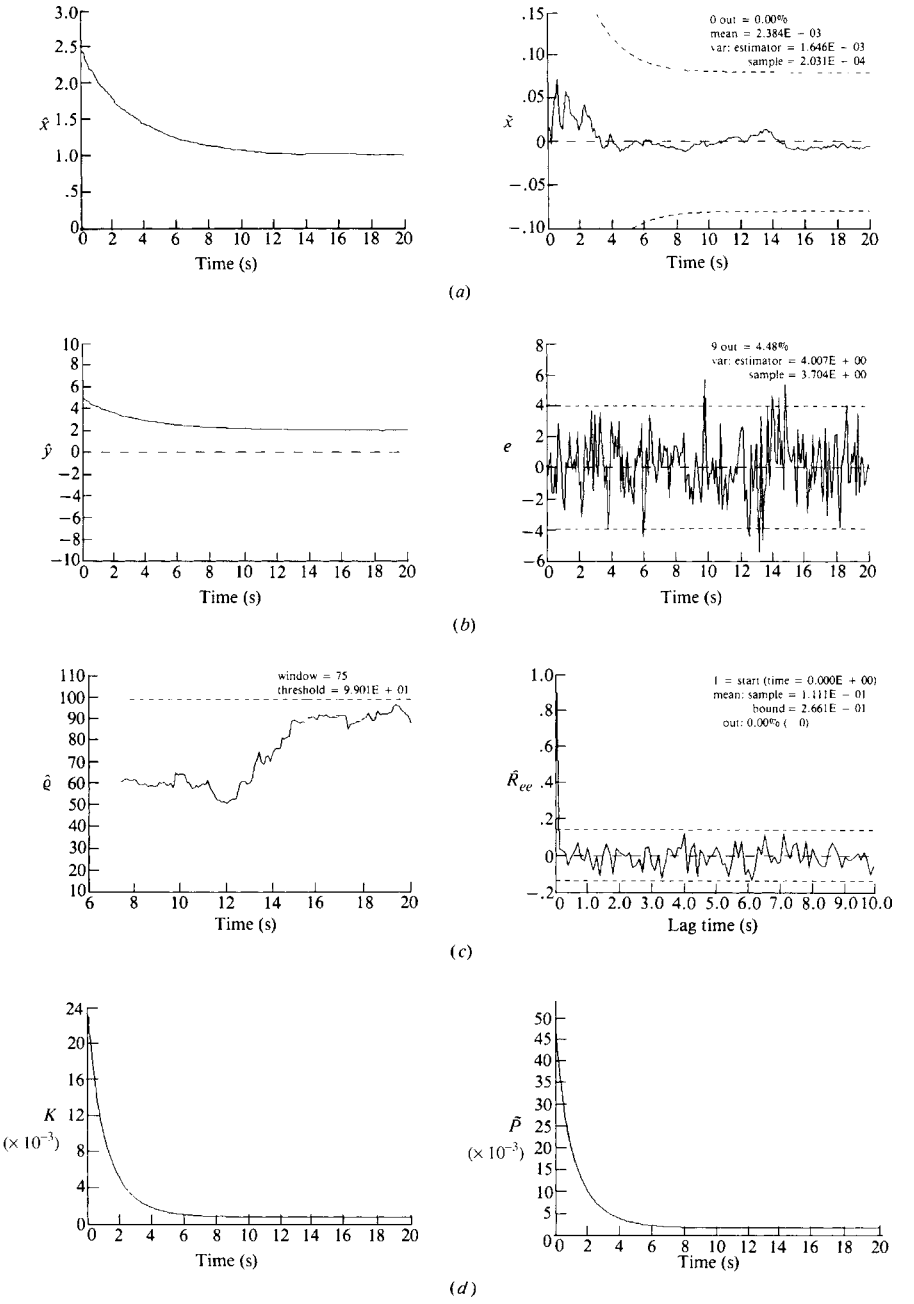



**Figure 5.4.** *MBP* design for *RC*-circuit problem. (*a*) Estimated state (voltage) and error. (*b*) Filtered voltage measurement and error (innovations). (*c*) *WSSR* and zero-mean/whiteness tests. (*d*) Gain and corrected error covariance.

## 5.6 TUNING AND MODEL MISMATCH IN THE *MBP*

### 5.6.1 Tuning with State-Space *MBP* Parameters

In this section we discuss the effect of various parameters available in the state-space *MBP* for tuning. In the last section we saw that if we are given the exact model parameters, then it is straightforward to construct the minimum variance processor. In practice, these parameters are rarely given and tests must be performed to estimate them. The usual approach taken is to develop a first principles model from the underlying physics; however, an alternative is to "identify" the model from measured data (e.g., see [18], [19]). Once the model is developed, it is used along with testing of the actual system to estimate the noise statistics. The *MBP* is then constructed and adjusted using the statistical tests to analyze its performance on both simulated and measured data. To observe the effect of various filter parameters on the performance of the processor and develop some intuition, we choose the previous *RC*-circuit example and investigate the *MBP* parameters available. We select the three primary parameters as:

1. Process noise covariance $R_{ww}$
2. Measurement noise covariance $R_{vv}$
3. Initial condition (state) covariance $\tilde{P}(0|0)$ (or equivalently $\hat{x}(0|0)$)

Before we continue with the discussion, let us recall the expression for the gain. Previously we showed that the variation of the gain is related to variations in $\tilde{P}$ and $R_{ww}$, $R_{vv}$, that is,

$$K(t) = \tilde{P}(t|t)C'(t)R_{vv}^{-1}(t)$$

However, if we recall the covariance prediction equation of Table 5.1, we see that the model uncertainty can be represented by the process noise covariance, $R_{ww}$. Theoretically, as $t \to \infty$, then $\tilde{P} \to R_{ww}$ which implies that $R_{ww}$ provides a limit on the error covariance. For large $R_{ww}$, $\tilde{P}$ is large, indicating high uncertainty or a "poor" model. For small $R_{ww}$, $\tilde{P}$ is small, indicating a "good" model. We can go through the same reasoning for the innovation covariance, since $R_{ee}(t) \to R_{vv}(t)$ for $t \to \infty$. For large $R_{vv}$, $R_{ee}$ is large, indicating high uncertainty or a "poor" measurement. For small $R_{vv}$, $R_{ee}$ small, indicating a "good" measurement. Therefore, we can *heuristically* conceive of the gain as the ratio of process-to-measurement noise ($C = I$) or *noise ratio*; that is,

$$K \propto \frac{R_{ww}}{R_{vv}} \tag{5.76}$$

Using the results of Eq. (5.1), we can again see that the variations of $R_{ww}$ are consistent with the variation of $K$. Intuitively we can perceive the *MBP* as a deterministic filter with a time-varying bandwidth ($BW$) specified by the processor

gain. In fact, combining the predicted and corrected state estimate equations of Eqs. (5.13) and (5.21), we have that

$$\begin{aligned}\hat{x}(t|t) &= \hat{x}(t|t-1) + K(t)e(t) = \hat{x}(t|t-1) + K(t)\left(y(t) - C\hat{x}(t|t-1)\right) \\ &= (I - K(t)C)\,\hat{x}(t|t-1) + K(t)y(t) \\ &= (I - K(t)C)\,A\hat{x}(t-1|t-1) + K(t)y(t) \end{aligned} \tag{5.77}$$

We have ignored the deterministic input in this relation. So we see that the new transition matrix of the *MBP* is defined by $(I - K(t)C)\,A$ and it is the poles of this matrix that determine the response ($BW$) of the *MBP*. In essence, adjusting the gain through the process and measurement covariance of Eq. (5.76) increases or decreases the overall bandwidth of the processor.

If we consider $K$ in terms of the noise ratio, then the effect on the processor bandwidth can be explained easily. As $R_{ww}$ increases, K increases (believing measurement) and the processor $BW$ increases (good measurement). Thus the *MBP* transient performance is faster ($1/BW$), but at the cost of more noise passed due to the higher BW creating noisier estimates. The identical effect is achieved by small $R_{vv}$. Now, if $R_{ww}$ decreases ($R_{vv}$ increases), K decreases, thereby decreasing the $BW$. The processor performance is slower, but the noise is filtered resulting in smoother estimates. To illustrate these concepts, we consider again the tuning example of Example 5.1.

***Example 5.2*** Using the $RC$-circuit tuning problem of the previous section, we vary the process noise covariance and analyze its effect on the *MBP* performance. We choose three values of $R_{ww}$ $(1, 0.01, 10^{-12})$. *SSPACK_PC* was applied to this problem and the results are shown in Figure 5.5 [17]. The state estimates for the large, medium, and small values of $R_{ww}$ are shown in Figure 5.5*a* through *c* along with the corresponding estimation errors. For large $R_{ww}$ the processor bandwidth is increased, and the state estimates are noisier, becoming smoother as $R_{ww}$ decreases. This occurs because the gain is large for large process noise covariance (believe measurement), and the state estimate just becomes a scaled measurement, not believing the model at all. Also note that the estimation error is large (0.61) for large $R_{ww}$. Because of the relationship of process noise covariance to the steady-state $\tilde{P}$ and $K_{ss}$, we note that in all cases the bounds predicted by the *MBP* have decreased with decreasing $R_{ww}$. For example, the error variance for small $R_{ww}$ is $1.4 \times 10^{-11} \ll 0.61$. For comparison purposes, we show the state estimates for the medium and small $R_{ww}$ cases superimposed in Figure 5.5*d* along with the corresponding error covariances for each case. We conclude that $\tilde{P}$ is proportional to $R_{ww}$ in steady state. This completes the example.

Next we investigate a similar effect by varying $R_{vv}$. Here the measurement noise covariance will not have as much direct effect on $\tilde{P}$, but it will effect the gain through the innovations covariance.

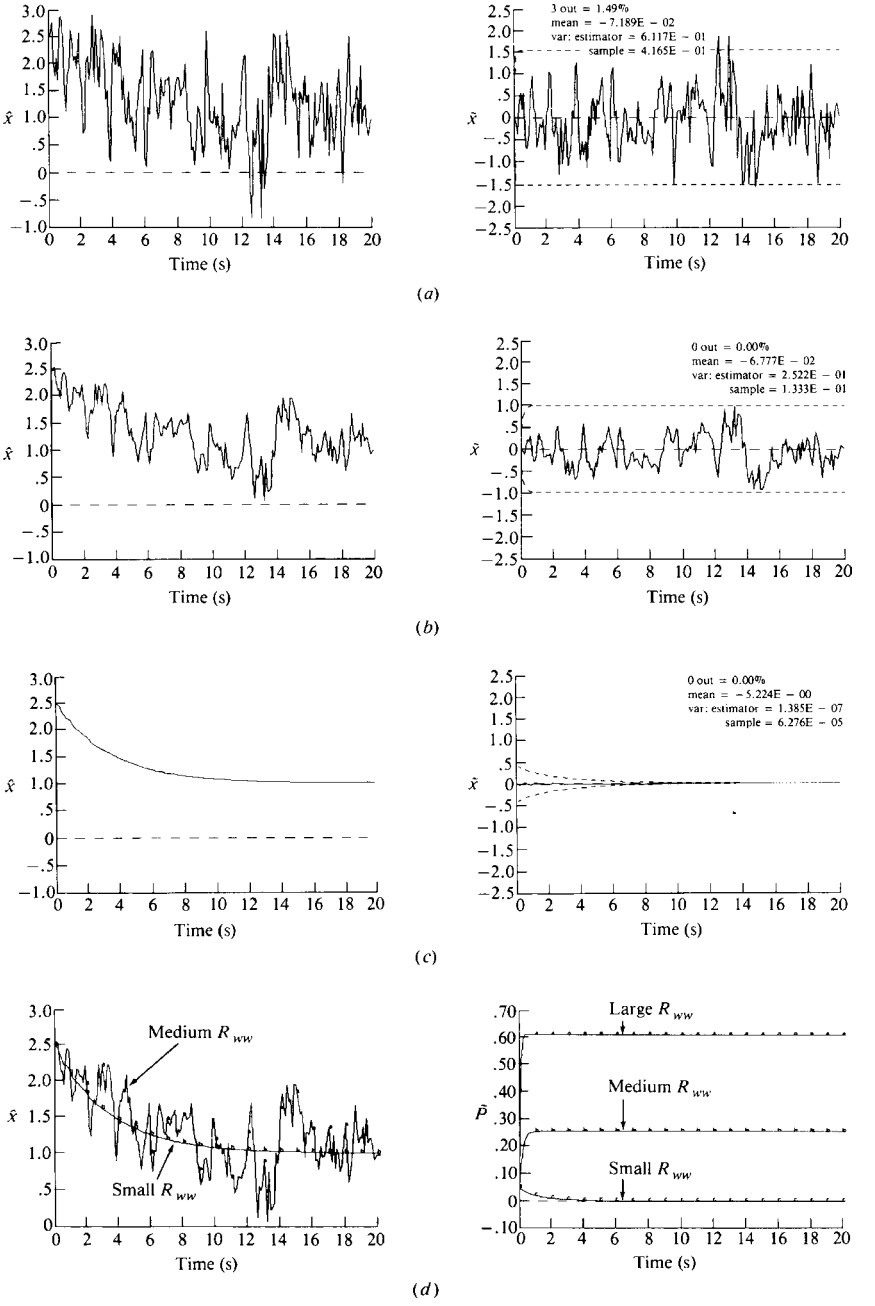



**Figure 5.5.** *RC*-circuit problem *MBP* tuning with process noise covariance ($R_{ww}$). (*a*) Estimated state and error covariances for *large* $R_{ww}$. (*b*) Estimated state and error covariances for *medium* $R_{ww}$. (*c*) Estimated state and error covariances for *small* $R_{ww}$. (*d*) State estimates for *medium* and *small WSSR* for *medium* and *small* $R_{ww}$ with corresponding error covariances.

***Example 5.3*** Consider the *RC*-circuit tuning problem and vary $R_{vv}$ for the values $(20, 0.1, 10^{-4})$ analyzing its effect on the *MBP* performance using *SSPACK_PC* [17]. The effect of large, medium, and small measurement noise covariances on the state estimates are similar to those in Figure 5.5 with small $R_{vv}$ replacing large $R_{ww}$. These results follow the bandwidth interpretation of the gain. The innovations sequences are shown in Figure 5.6*a* through *c* for the various values of $R_{vv}$. It is clear from the figure that the confidence bounds are more sensitive to changes in $R_{vv}$ because of its direct relation to $R_{ee}$. Also the corresponding error covariance in Figure 5.6*a* indicates that the transient performance is affected as well, but all the

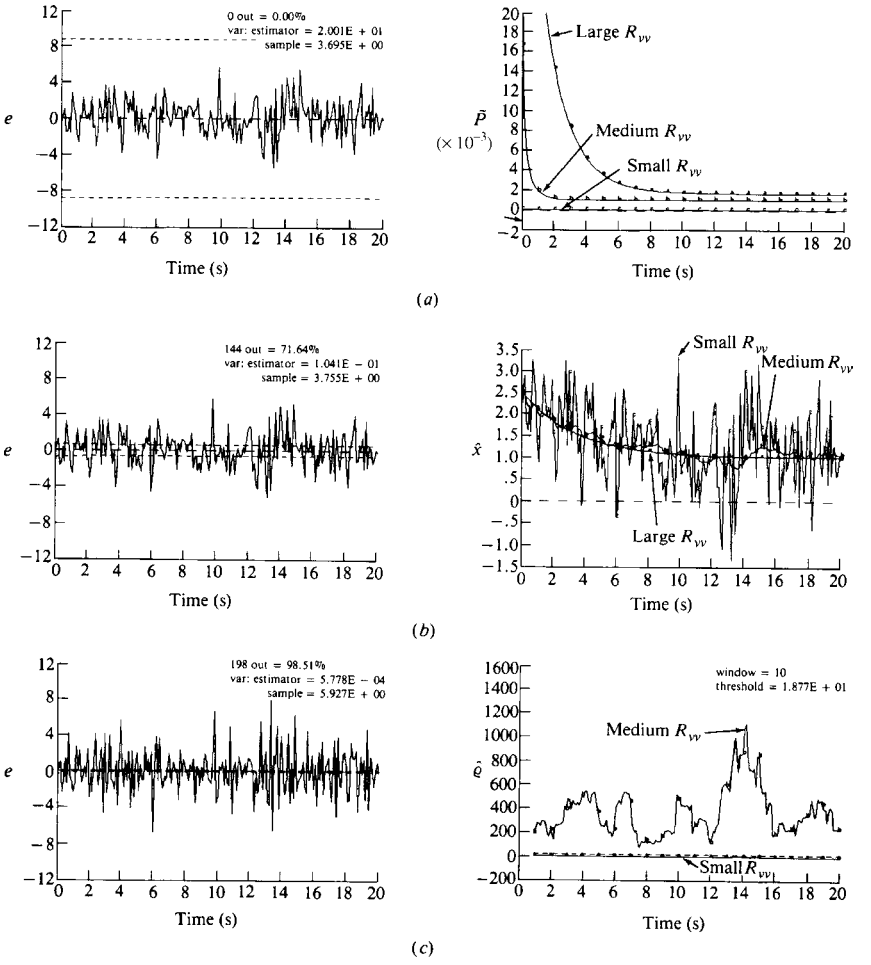



**Figure 5.6.** *RC*-circuit problem *MBP* tuning with measurement noise covariance ($R_{vv}$). (*a*) Innovations sequence for *large* $R_{vv}$ and error covariances. (*b*) Innovations sequence for *medium* $R_{vv}$ and state estimates. (*c*) Innovations sequence for *small* $R_{vv}$ and *WSSR* for medium and small $R_{vv}$.

steady-state $\tilde{P}$ appear relatively insensitive to these changes. The state estimates of each case are shown in Figure 5.6*b* where we see that for small $R_{vv}$ (believe measurement), noisy state estimates dominate, while for large $R_{vv}$ (believe model) smooth estimates result. This completes the example.

Finally, we study the effect of the initial conditions $\left(\hat{x}(0|0), \tilde{P}(0|0)\right)$ on the processor performance. Jazwinski [5] has shown that the initial statistic $\tilde{P}(0|0)$ is *forgotten* as more and more data are processed. This is important because $\tilde{P}(0|0)$ is rarely known. It also implies that the numerical errors in calculating $\tilde{P}$ appear relatively insensitive to these changes. These results follow from the fact that as long as the processor is stable and the system (state-space model) is completely controllable and observable (see Chapter 2), then

$$\lim \ \tilde{P}(t|t) \longrightarrow \tilde{P} \ \text{ small} \qquad \text{as } t \to \infty \tag{5.78}$$

Thus the estimation error approaching zero implies that the state estimate converges to the true value given that there exists enough data. The initial estimates will affect the transient performance of the processor, since a large initial error covariance $\tilde{P}(0|0)$ gives a large initial gain $K(0)$ heavily weighting the initial measurements and ignoring the model. Consider the following example, which indicates the effect of the initial conditions.

***Example 5.4*** Again using the $RC$-circuit problem, we vary $\tilde{P}(0|0)$ for $(10, 10^{-12})$ and analyze the effect on the processor performance. The results are shown in Figure 5.7*a* where we observe that the initial error covariance is quickly "forgotten" by the *MBP* as more data are processed, since both estimates depicted become identical after 6 seconds. The corrected error covariances in Figure 5.7*b* indicate that $\tilde{P}$ converges to the same steady-state value as expected. This completes the example. We summarize all of these results in Table 5.4.

### 5.6.2 Model Mismatch Performance in the State-Space *MBP*

In this subsection we investigate the effects on the *MBP* of using models that do not "exactly" match the underlying process dynamics. We first analyze the performance of the processor under these conditions and then continue with our tuning example to demonstrate the effects.

**Table 5.4. Performance Summary of State-Space *MBP* Tuning Parameters**

| Parameter | Transient | Steady State | Indicators |
|---|---|---|---|
| $R_{ww}$ | Yes | Yes | Noisy estimates, $\pm 2\sqrt{R_{ee}} \pm 2\sqrt{\tilde{P}}$ |
| $R_{vv}$ | Yes | Yes | $\pm\sqrt{R_{ee}} \pm 2\sqrt{\tilde{P}}$ |
| $\tilde{P}$ | Yes | No | Initial innovations and tracking errors |

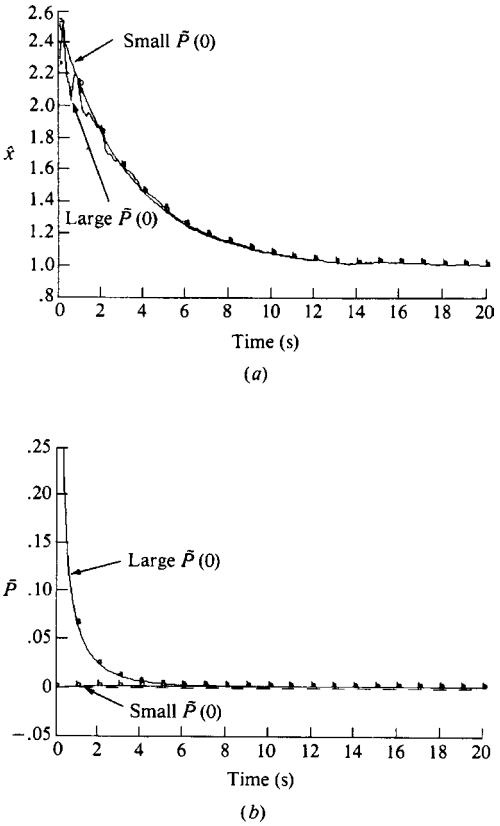



**Figure 5.7.** *RC*-circuit problem *MBP* tuning with initial error covariance ($\tilde{P}(0|0)$). (*a*) Estimated states. (*b*) Error covariances.

Recall that in applying a *MBP* to a specific system, the state-space model parameters, noise statistics and initial conditions must be specified in order to achieve the minimum error variance estimate. The process model is usually an approximation to the underlying physics, and its parameters and noise statistics are rarely exact; that is, the process model used for the processor differs from the real process that generates the measurements. Sometimes the approximation is intentional. For instance, using a reduced-order model in order to decrease computational complexity or linearizing a nonlinear model. It is clear that an imprecise processor model degrades *MBP* performance as observed in the last subsection, when we chose various values for process, measurement, and initial error covariances. In fact these modeling errors or "model mismatches" may even cause the processor to diverge. In designing a *MBP* it is therefore important to evaluate the effect of the approximations made in the selection of the models. In this subsection we do not present the precise mathematical relationship caused by modeling errors. These relations can be found in Jazwinski [5] and Martin [15].

Suppose the actual process is described by

$$\begin{aligned} \text{Actual:} \quad x(t) &= Ax(t-1) + Bu(t-1) + w(t-1) \\ y(t) &= Cx(t) + v(t) \end{aligned} \tag{5.79}$$

$$\begin{aligned} \text{Model:} \quad x(t) &= \overline{A}x(t-1) + \overline{B}u(t-1) + w(t-1) \\ y(t) &= \overline{C}x(t) + v(t) \end{aligned} \tag{5.80}$$

If we design a *MBP* using the model set defined by $\overline{\Sigma} := \{\overline{A}, \overline{B}, \overline{C}, \overline{R}_{ww}, \overline{R}_{vv}, \overline{\tilde{P}}(0|0), \hat{\overline{x}}(0|0)\}$, then it processes the actual inputs $\{u(t)\}$ and measurements $\{y(t)\}$ with this imprecise model. The processor is given by

$$\begin{aligned} \hat{x}(t|t-1) &= \overline{A}\hat{x}(t-1|t-1) + \overline{B}u(t-1) \\ \hat{x}(t|t) &= \hat{x}(t|t-1) + \overline{K}e(t) \end{aligned} \tag{5.81}$$

If we assume that the modeling error is $(\triangle A, \triangle B, \triangle C)$ (ignoring the noise covariances and initial conditions), then we have that

$$\begin{aligned} \triangle A &:= A - \overline{A} \\ \triangle B &:= B - \overline{B} \\ \triangle C &:= C - \overline{C} \end{aligned} \tag{5.82}$$

To derive expressions for the errors, we must use the "actual" model set $\Sigma := \{A, B, C, R_{ww}, R_{vv}, \tilde{P}(0|0), \hat{x}(0|0)\}$, the definitions of Eq. (5.82) and substitute into the *MBP* relations of Table 5.1. For details, see [5] or Martin [15] where it is shown that the innovations sequence is *no longer* zero-mean and white when model mismatch occurs. In fact the innovations mean and covariance with mismatch are given by

$$\tilde{m}_e(t-1|t-1) := E\{e(t-i)\} = C\tilde{m}_e(t-i|t-i-1) + \triangle C m(t-i) \tag{5.83}$$

where $\tilde{m}_e := E\{\tilde{x}\}$ and $m := E\{x\}$ which are not zero-mean due to the model mismatch $(\triangle A, \triangle B, \triangle C)$, or

$$\tilde{m}_e = f\,(\triangle A, \triangle B, \triangle C) \tag{5.84}$$

Similarly the innovations covariance is given by

$$\begin{aligned} \overline{R}_{ee}(t-i) = {} & C\tilde{P}(t,t-i)C' + CR_{\tilde{x}x}(t,t-i)\triangle C' + CR_{\tilde{x}v}(t,t-i) \\ & + \triangle C R_{x\tilde{x}}(t,t-i)C' + \triangle C\tilde{P}(t,t-i)\triangle C' \quad i = 0,\ldots,t \end{aligned} \tag{5.85}$$

where $\tilde{P} = f\,(\triangle A, \triangle C)$.

Since the quantities of prime concern in the processor are the state estimates and the innovations, let us aggregate all of the model mismatches into a $\triangle$ term and investigate the effect on the state estimation error and innovations statistics. We will assume that the error can be decomposed as

$$\tilde{X}^* := \underbrace{\tilde{X}}_{\text{Normal}} \quad - \quad \underbrace{\triangle\tilde{X}}_{\text{Mismatch}} \tag{5.86}$$

along with the corresponding innovations given by

$$E^* := \underbrace{E}_{\text{Normal}} \quad - \quad \underbrace{\triangle E}_{\text{Mismatch}} \tag{5.87}$$

We assume the worst case, that is, the estimator cannot track the state $x(t)$, and $E\{\tilde{x}(t|t-1)\}$ is not zero, or

$$E\{\tilde{X}^*\} = E\{\tilde{X}\} - E\{\triangle\tilde{X}\} = -E\{\triangle\tilde{X}\} \tag{5.88}$$

Similarly the innovations are

$$E\{E^*\} = E\{E\} - E\{\triangle E\} = -E\{\triangle E\} \tag{5.89}$$

So we see that in the worst case that the innovations and state estimation error are *biased* when a model mismatch occurs. To see the effects of mismatch, consider the following example.

***Example 5.5*** In this example we investigate the *RC*-circuit tuning problem for the following three cases:

1. Process dynamics modeling error: $\triangle A$
2. Input or bias modeling error: $\triangle B$[5]
3. Measurement or trend modeling error: $\triangle C$

We apply *SSPACK_PC* [17] to analyze the effects of these modeling errors on *MBP* performance.

*Case 1: Process dynamics errors*: $A_{true} \longrightarrow A + \triangle A = 0.97 - 0.27 = 0.7$. The results are shown in Figure 5.8. In Figure 5.8*a* we see the estimated state and estimation error with confidence limits. The time constant of the state estimate and therefore the processor; that is $(I - KC)\,A$ is much faster that the actual process due to the model mismatch. This is demonstrated by comparing Figure 5.8*a* with the true state in Figure 5.8*b*. The estimation error gives a clear

[5]The errors $\triangle B$, $\triangle C$ can be considered bias or trend errors, since the terms $u$, $Cx$ can be used to model biases and trends [20].

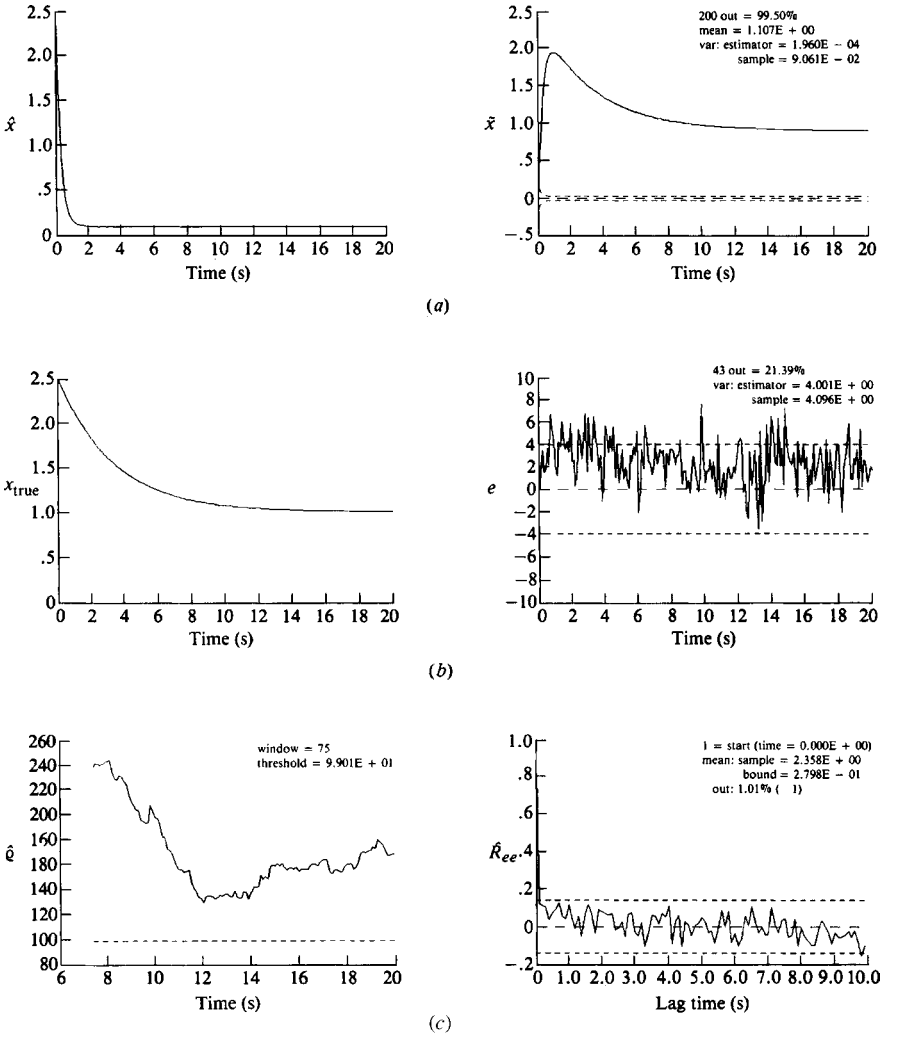



**Figure 5.8.** *RC*-circuit problem *MBP* tuning with process dynamics model mismatch ($\triangle A$). (*a*) Estimated state and error. (*b*) True state and innovations. (*c*) *WSSR* and zero-mean/whiteness tests.

indication of the mismatch. The innovations are biased ($2.4 > 0.28$). The *WSSR* is no longer below the threshold for a window length of 75 samples as before, but the innovations still appear white.

*Case 2: Input (bias) error*: $B_{true} \longrightarrow B + \triangle B = 100 + 900 = 1000$. The results of this investigation are shown in Figure 5.9. The state estimate is severely deteriorated as indicated by the estimation errors shown in Figure 5.9*a*. The innovations are biased ($14 > 0.6$) and nonwhite (56% out with *WSSR* > threshold) due to the mismatched input or bias term.

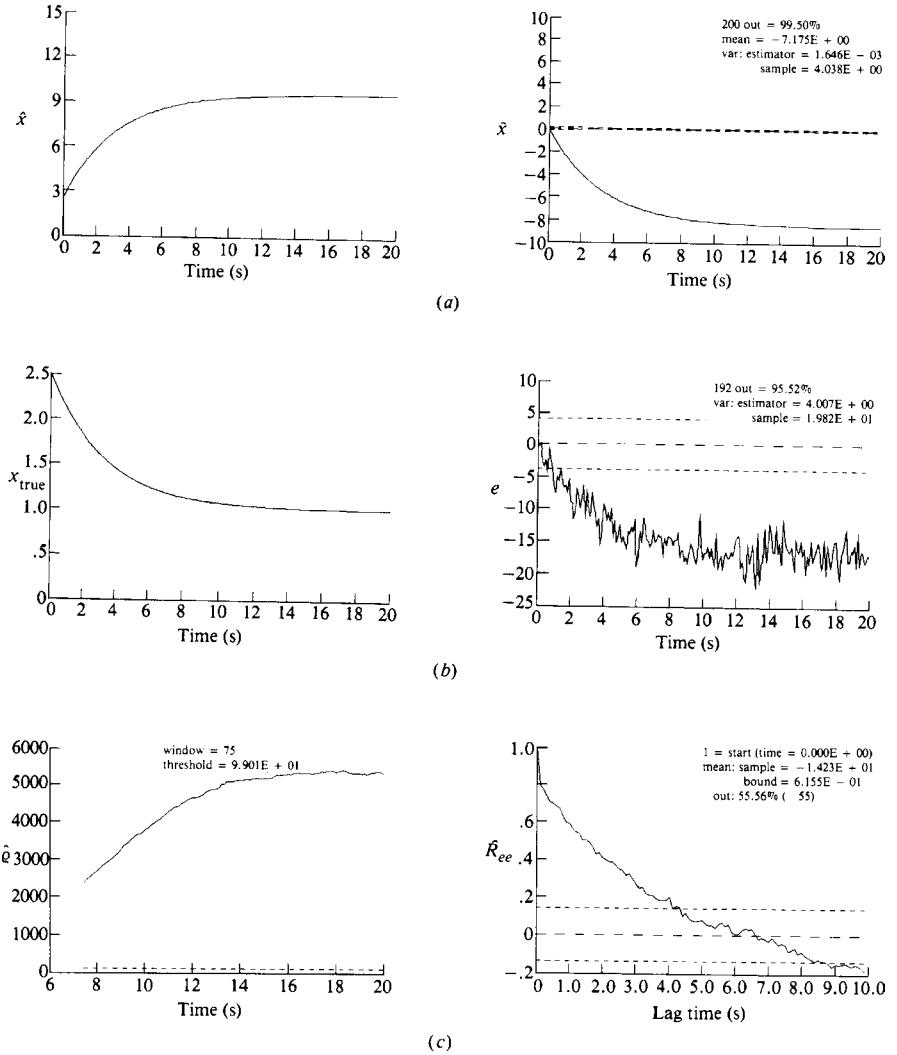



**Figure 5.9.** *RC*-circuit problem *MBP* tuning with input model mismatch ($\triangle B$). (*a*) Estimated state and error. (*b*) True state and innovations. (*c*) *WSSR* and zero-mean/whiteness tests.

*Case 3: Measurement (trend) error*: $C_{true} \longrightarrow C + \triangle C = 2 + 8 = 10$. The results are shown in Figure 5.10. The estimates are unacceptable as indicated by the estimation errors shown in Figure 5.10*b*. We see that the innovations are biased ($4.9 > 0.4$) and nonwhite (1% out with *WSSR* exceeding the threshold).

This completes the section on modeling errors and their effect on *MBP* performance, next we consider the general approach to processor design.

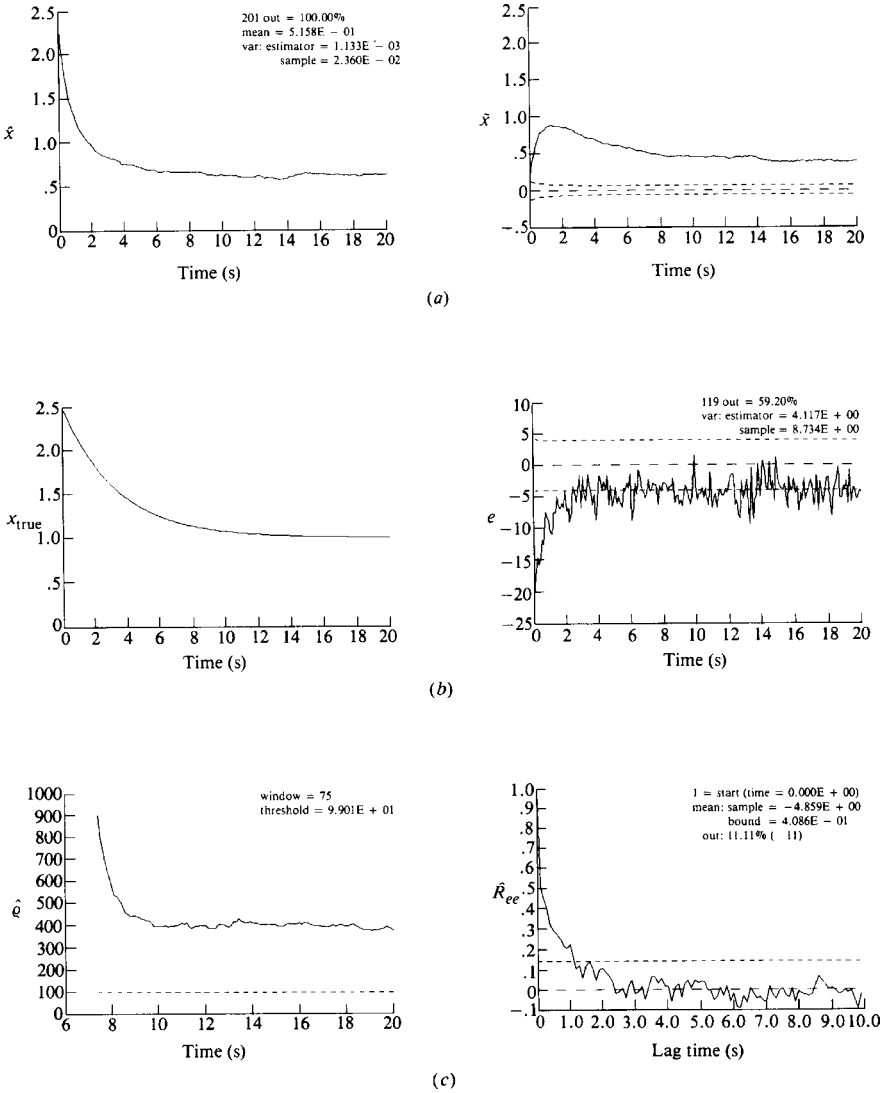



**Figure 5.10.** *RC*-circuit problem *MBP* tuning with output model mismatch ($\triangle C$). (*a*) Estimated state and error. (*b*) True state and innovation. (*c*) *WSSR* and zero-mean/whiteness tests.

## 5.7 *MBP* DESIGN METHODOLOGY

In this section we develop a general methodology to design the state-space *MBP* or Kalman filter. This approach has evolved primarily from the solution of navigation and tracking problems, where the models involved are quite good and are continually updated and improved [20]. Designing a processor is a straightforward

procedure as long as all of the information about the underlying process or system under investigation is available or can be gathered in a reasonable period of time. After deciding that a processor is necessary, the development proceeds through various phases of *MBP Design Methodology*:

1. Model development
2. Simulation
3. Processor (minimum variance) design and error analysis
4. Application (tuning) to data sets
5. Performance analysis

The model development phase consists of developing models characterizing the underlying process physics or phenomenology. Here we develop a "process model" in the mathematical form of a linear or nonlinear dynamic equations. Typically this requires that the signal processor has the required knowledge of the physics or that an expert in the area is available to perform this task. Simultaneously the measurement instrumentation is investigated in terms of bandwidth, response time, physical relations, and so forth, to develop a corresponding "measurement system model." Finally models of the inherent uncertainties must be developed. Here random and systematic errors should be considered as well as noise models. Thus in this modeling step we develop the mathematical models of the propagation dynamics, measurement system and noise sources. Once the state-space models have been specified, we search for a set of parameters/data to initialize the processor. The parameters/data can come from a historical database, or from work performed by others in the same or equivalent environment, or from previous detailed "truth" model computer simulations or merely from educated guesses based on any a priori information and experience of the modeler/designer.

Note that part of the modeling and simulation phases consists of gathering all of the parameters/data information and employing them in our state-space models to perform the Gauss-Markov simulation, the second step. Here the use of *additive noise sources* in this formulation enables us to "lump" the uncertainties evolving from: initial conditions, propagation dynamics, into $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{ww})$; and measurement noise, $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{vv})$. Note that we are *not* attempting to "precisely" characterize these uncertainties, but we simply admit to our "ignorance" and lump the uncertainty into additive gaussian noise processes controlled by their inherent covariances. Once we have specified the parameters/data, we are now ready to perform Gauss-Markov simulations. This procedure is sometimes referred to as "sanity" testing. We typically use the following definitions of signal-to-noise ratio (*SNR*) in deciding what signal–noise levels to perform the simulations at

$$SNR_{\text{in}} \equiv \frac{\mathbf{P}_{i,i}}{\mathbf{R}_{ww}(i,i)}, \qquad i = 1, \ldots, N_x; \tag{5.90}$$

and

$$SNR_{\text{out}} \equiv \frac{\mathbf{c}_i'(t)\mathbf{P}_{i,i}\mathbf{c}_i(t)}{\mathbf{R}_{vv}(i,i)} \tag{5.91}$$

where $\mathbf{P}_{i,i} := \text{diag}[\mathbf{P}]$ is the state covariance ($\text{cov}(\mathbf{x}(t))$) and $\mathbf{c_i}(t)$ is the $i$th column of the measurement model; $\mathbf{R}_{ww}$, $\mathbf{R}_{vv}$ are the noise covariances.

Once these models are completed, then a simulator (simulation phase) should be constructed to ensure that reasonable measurements are being synthesized. The simulation, typically called a *truth model*, can also be used to study simplifications of the models, if they are complex (see Figure 5.11). Given that the models appear adequate, then various *MBP* can be designed from the truth model simulated measurements. However, if error analysis is to be performed, then the optimal *MBP* must be developed (if possible) based on the truth model in order to produce the (minimum) error covariance matrix, $\tilde{P}_{\text{true}}$, for performance comparisons. The square root of the diagonals of $\tilde{P}_{\text{true}}$ represent the *RMS* errors that can be expected from each state estimate—this is the *best* that can be achieved for the problem providing a lower performance bound. Sometimes the truth model is of very high dimensionality or nonlinear, but it is still possible to at least obtain an approximation to the error covariance. By selecting a more detailed (complex) model than is anticipated for the actual application, it is then possible to estimate the bounding error covariance matrix. This phase is also shown in Figure 5.11. This methodology employing the truth model and simulated data produces results similar to the tuned estimator discussed previously.

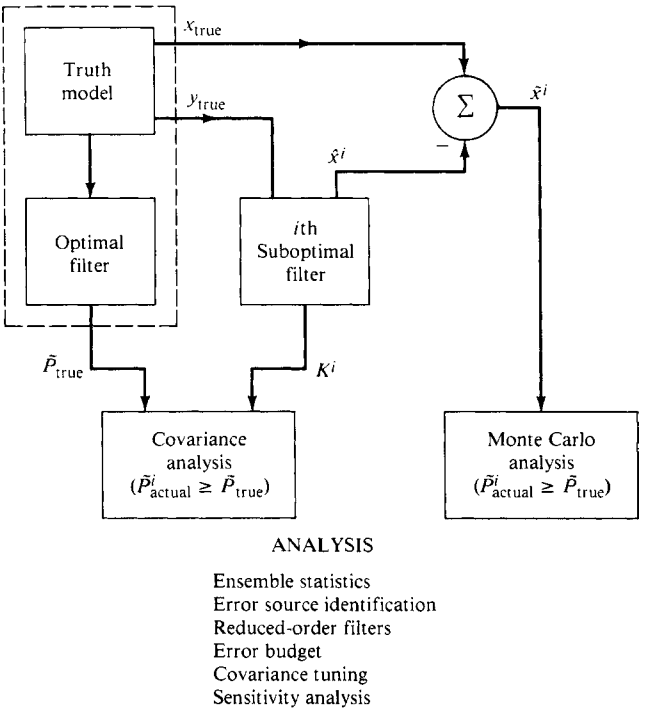



**Figure 5.11.** *MBP* Performance Analysis.

Once the Gauss-Markov simulation is complete, the *MBP* is designed to achieve a minimum variance estimate in the linear model case or a best mean-squared error estimate in the nonlinear model case—this is called *minimum variance design*. Since the models used in the processor are "exactly" those used to perform the simulation and synthesize the design data, we will have as output the minimum variance or best mean-squared error estimates. For the nonlinear case the estimates are deemed approximate minimum variance because the processor uses techniques which approximate the nonlinear functions in order to obtain the required estimates. We expect the processor to perform well using the results of the various Gauss-Markov simulations to bound its overall expected performance on real data. We essentially use simulations at various *SNR*s (above) to obtain a "feel" for learning how to tune (adjusting noise covariances, etc.) the processor using our particular model sets and parameters.

Once we have completed the simulations and studied the sensitivity of the processor to various parameter values, we are ready to attack the actual data set. With real data, the processor can only perform as well as the dynamical models used represent the underlying phenomenology generating the data. Poor models used in the processor can actually cause the performance to degrade substantially, and enhancement may not be possible at all in contrast to the simulation step where we have assured that the model "faithfully" represents the data. In practice, it is never possible to accurately model everything. Thus the goal of minimum variance design is to achieve as close to the optimal design as possible by investigating the consistency of the processor relative to the measured data. This is accomplished by utilizing the theoretical properties of the processor [1], [2]. That is, the processor is deemed *optimal*, if and only if, the residual/innovations sequence is zero-mean and statistically white (uncorrelated). This approach to performance analysis is much like the results in time series/regression analysis which implies that when the model "explains" or fits the data, nothing remains and the residuals are uncorrelated [3], [12]. Therefore, when applying the processor to real data, it is necessary to adjust or "tune" the model parameters until the innovations are zero-mean/white. If it is not possible to achieve this property, then the models are deemed inadequate and must be improved by incorporating more of the phenomenology. The important point here is that the model-based schemes enable the modeler to assess how well the model is performing on real data and decide where it may be improved. For instance, for the experimental data, we can statistically test the innovations and show that they are white. However, when we visually observe the sample correlation function estimate used in the test, it is clear that there still remains some correlation in the innovations. This leads us, as modelers, to believe that we have not captured all of the phenomenology that has generated the data. Therefore we must improve the model or explain why the model is inadequate. Investigating properties of the resulting innovations sequence can indicate what has *not* been modeled. For instance, taking the Fourier transform of the innovations sequence can indicate unmodeled resonances that should be included (see Section 9.3).

Care must be taken when using these statistical tests as noted in [2]. If the models are nonlinear or nonstationary, then the usual zero-mean/whiteness tests, that is,

testing that 95% of the sample (normalized) innovation correlations lie within the bounds rely on quasi-stationary assumptions and sample statistics to estimate the required correlations. However, it can be argued heuristically that when the estimator is tuned, the nonstationarities are being tracked by the MBP even in the nonlinear case. Therefore, the innovations should be covariance stationary.

When data are nonstationary, then a more reliable statistic to use is the *weighted sum-squared residual* (*WSSR*), which is a measure of the overall global estimation performance for the MBP processor determining the "whiteness" of the innovations sequence. It essentially aggregates all of the information available in the innovation vector and tests whiteness by requiring that the decision function lies below the specified threshold to be deemed statistically white. If the *WSSR* statistic does lie beneath the calculated threshold, then theoretically the estimator is tuned and said to converge. That is, for vector measurements, we test that the corresponding innovations sequence is zero-mean/white by performing a statistical hypothesis test against the threshold. Under the zero-mean assumption the *WSSR* statistic is equivalent to testing that the vector innovation sequence is white. Even in the worst case where these estimators may not prove to be completely consistent, the processor (when tuned) predicts the nonstationary innovations covariance, $\mathbf{R}_{ee}(t)$, enabling a simple (varying with $t$) confidence interval to be constructed and used for testing as well, that is,

$$\left[e(t) \pm 1.96\sqrt{\mathbf{R}_{ee}(t)}\right], \qquad t = 1, \ldots, N \tag{5.92}$$

Thus overall performance of the processor can be assessed by analyzing the statistical properties of the innovations which is essentially the approach we take in the design on real data.

More specifically, the design of the processor also consists of certain phases: processor model development, tuning, error analysis—prior to application on the measured data. The development of various processor designs is based primarily on choosing from a variety of simplified models, incorporating them into the processor, tuning and then analyzing its performance compared to the optimal. Analysis of *MBP* performance can be achieved in a variety of ways. The brute force method is to develop and tune each individual design construct, generate an ensemble of estimation errors ($\tilde{X} = X_{\text{true}} - \hat{X}$) by varying initial condition parameters ($\hat{x}(0|0)$, $\tilde{P}(0|0)$) or using random number generators [11] and calculating sample statistics. This is called the *Monte Carlo method*, and it is very costly and time-consuming. However, for some processes there is little other choice (see Figure 5.11).

Another approach to error or performance analysis is called the *covariance analysis method*, It uses the optimal covariance $\tilde{P}_{\text{true}}$ and the various suboptimal gain sequences $\{K^i\}$ produced by each ($i$th) processor design. Here each suboptimal (since it does not employ the truth model) processor is tuned to produce the corresponding gain sequence. This sequence is then used with the truth model to develop the actual error covariance $\tilde{P}_{\text{actual}}$ (see Figure 5.11). The *RMS* errors calculated from the diagonals of this matrix can be used to compare performance

with the optimal design, as well as to calculate error budgets, identify source errors, tune processor covariances, perform sensitivity analysis, etc. (see [20] for details). The advantage of this approach over the brute force or Monte Carlo method is that if both truth and processor models are *linear*, then only *one* run of the designed processor is necessary for analysis.

To illustrate the importance of the covariance analysis method in *MBP* design, let us develop the equations (linear case) to describe the approach and then investigate a simple example to demonstrate its application. Recall that the error covariance equations evolve during the prediction and correction steps of the *MBP* algorithm. For the optimal design we use the truth model to obtain $\tilde{P}_{\text{true}}$. That is, for prediction, we have[6]

$$\tilde{P}_{\text{true}}(t|t-1) = A_{\text{true}}\tilde{P}_{\text{true}}(t-1|t-1)A'_{\text{true}} + R_{ww_{\text{true}}} \tag{5.93}$$

For correction

$$\tilde{P}_{\text{true}}(t|t) = [I - K_{\text{true}}(t)C_{\text{true}}]\,\tilde{P}_{\text{true}}(t|t-1) \tag{5.94}$$

where $\tilde{P}_{\text{true}} \in R^{N_{X_{\text{true}}} \times N_{X_{\text{true}}}}$.

Usually the truth model is of higher dimensionality than the various processor designs, so we define the transformation of the actual state to the true state by the matrix $T \in R^{N_x \times N_{X_{\text{true}}}}$ as

$$x_{\text{actual}} = T x_{\text{true}} \quad \text{with (typically)} \quad T := [I_{N_x} \;\; | \;\; 0_{N_{X_{\text{true}}} - N_x}] \tag{5.95}$$

As shown in Figure 5.11, once the $i$th-processor is designed, the actual error covariance is calculated using the truth model and transformed (for $N_{X_{\text{true}}} > N_x$) suboptimal gain $K^i(t)$ as

$$\tilde{P}_{\text{actual}}(t|t-1) = A_{\text{true}}\tilde{P}_{\text{actual}}(t-1|t-1)A'_{\text{true}} + R_{ww_{\text{true}}} \tag{5.96}$$

and

$$\tilde{P}_{\text{actual}}(t|t) = \left[I - \left(T'K^i(t)\right)C_{\text{true}}\right]\tilde{P}_{\text{actual}}(t|t-1) \tag{5.97}$$

Error analysis is then performed by analyzing and comparing the actual to the true *RMS* errors. We know from Chapter 3 that the Cramer-Rao bound is given by $\tilde{P}_{\text{true}}$. Therefore we have that

$$\tilde{P}_{\text{actual}}(t|t) > \tilde{P}_{\text{true}}(t|t) \tag{5.98}$$

So by comparing square roots of the diagonal entries of these arrays, the degradation in performance using a suboptimal processor can be evaluated. A scalar performance index is also sometimes used [5] with

$$J(t|t) = \text{ trace } \tilde{P}(t|t) \tag{5.99}$$

[6]We choose a time-invariant model to simplify notation, but the results are still valid for time-varying, nonstationary systems.

Therefore

$$J_{\text{actual}}(t|t) > J_{\text{true}}(t|t) \tag{5.100}$$

To demonstrate these concepts, we have the following example:

***Example 5.6*** Suppose that we have the following system (similar to the *RC*-circuit tuning example) with *truth model*

$$x(t) = \begin{bmatrix} 0.97 & 0.1 \\ 0 & 0.87 \end{bmatrix} x(t-1) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t-1) + \begin{bmatrix} 1 \\ 1 \end{bmatrix} w(t-1)$$
$$y(t) = [2 \quad 2]x(t) + v(t)$$

with $R_{ww} = 0.0001$, $R_{vv} = 0.01$, $x(0|0) = [2.5 \quad 2.5]$, and $P(0|0) = \text{diag}[10^{-6} \quad 10^{-6}]$.

We design three suboptimal *MBP* and perform a covariance analysis as demonstrated in Figure 5.11. The designs are as follows:

*Optimal*: with $A_{\text{true}}$
*Suboptimal 1*: with $A_1 = \text{diag}[0.97 \quad 0.87]$
*Suboptimal 2*: with $A_2 = 0.97, c_2 = 2$
*Suboptimal 2*: with $A_3 = 0.87, c_3 = 2$

The optimal and suboptimal processors were run with the "steady-state" results summarized in Table 5.5. The overall performance is as expected. Using the $\sqrt{J}$ as an aggregated performance metric, we see that the optimal processor with embedded truth model gives the best performance, and in fact each suboptimal processor is ordered according to its overall performance. We also note that suboptimal processor 1 performs almost as well as the optimal, as indicated by the *RMS* estimation errors. Note also that this analysis predicts that suboptimal processor 2 will perform better than processor 3.

To confirm these predictions, we used *SSPACK_PC* [17] to simulate and analyze the performance of the processors. The results are shown in Figure 5.12. In

**Table 5.5. Covariance Analysis Results for *MBP* Design Example**

| | Optimal | Suboptimal 1 | Suboptimal 2 | Suboptimal 3 |
|---|---|---|---|---|
| $\sqrt{J}$ | 0.0286 | 0.0313 | 0.0346 | 0.0518 |
| $\sqrt{\tilde{P}_{11}}$ | 0.0239 | 0.0270 | 0.0280 | 0.0488 |
| $\sqrt{\tilde{P}_{22}}$ | 0.0157 | 0.0158 | 0.0203 | 0.0173 |
| $k_1$ | 0.1025 | 0.0788 | 0.0727 | — |
| $k_2$ | 0.0577 | 0.0568 | — | 0.0357 |

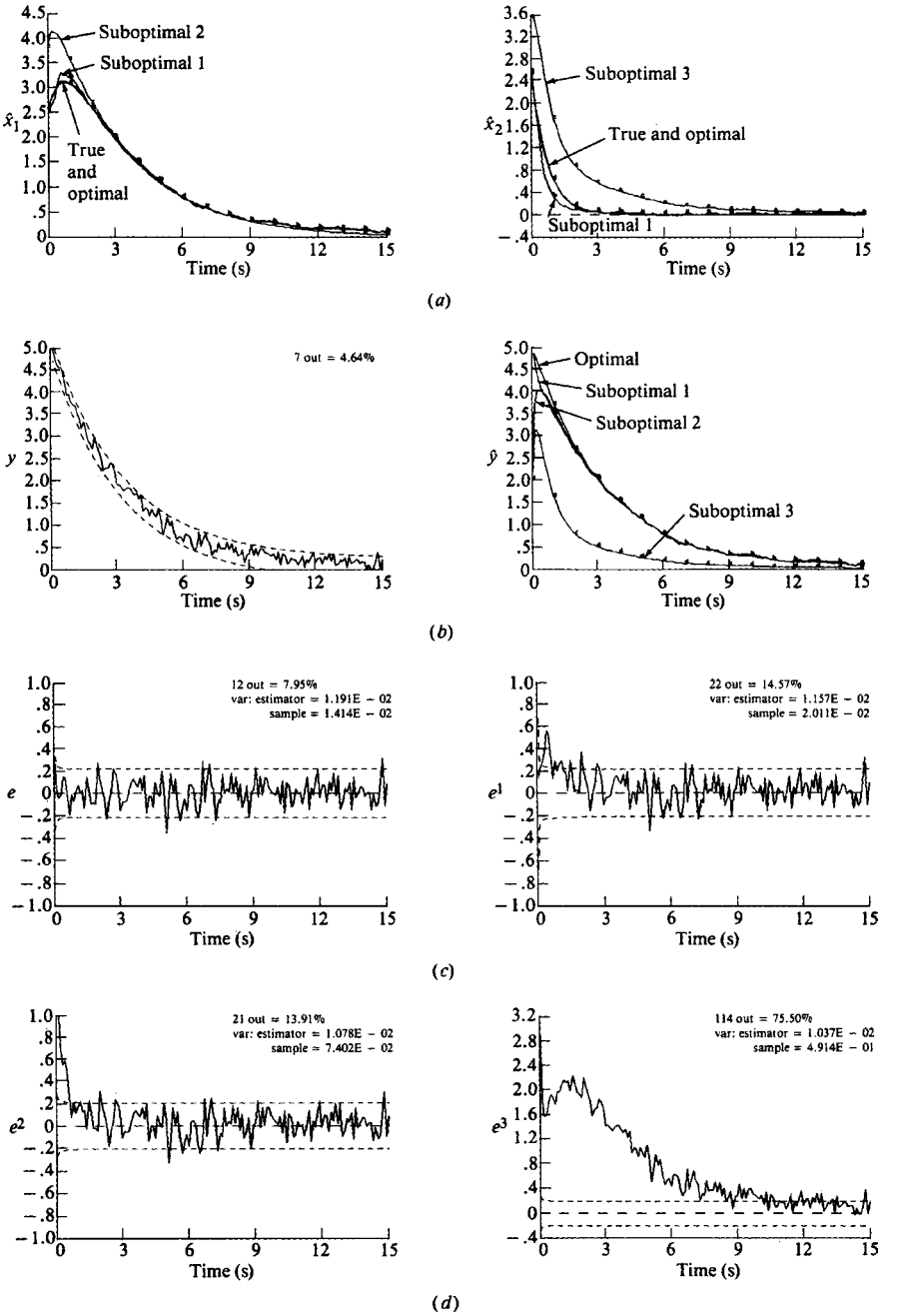



**Figure 5.12.** Suboptimal *MBP* designs for covariance analysis: (*a*) True and estimated states for optimal and suboptimal processors. (*b*) Simulated and processed measurements for optimal and suboptimal processors. (*c*) Optimal and suboptimal processors 1 and innovations. (*d*) Suboptimal processor 2 and 3 innovations.

**Table 5.6. Innovations Analysis Results for *MBP* Design Example**

| | Optimal | Suboptimal 1 | Suboptimal 2 | Suboptimal 3 |
|---|---|---|---|---|
| Mean | 0.013 | 0.053* | 0.074* | 0.749* |
| Whiteness (%) | 1.4 | 10.0* | 6.8* | 66.2* |

*Failed test.

Figure 5.12*a* we observe the true, optimal, and suboptimal state estimates. Note that the optimal estimates overlay the true and can barely be distinguished. We also note that suboptimal processor 1 performs nearly as well as the optimal, which confirms the predictions of the covariance analysis. Suboptimal processor 2 begins tracking the true state after approximately 30 samples, and suboptimal filter 3 does not track the true state very well at all. The simulated and processed measurements are shown in Figure 5.12*b* confirming these predictions as well. Finally, the innovations sequence of each processor is shown in Figure 5.12*c* and 5.12*d*, along with the corresponding zero-mean/whiteness tests. The results of these tests are summarized in Table 5.6.

In each case except the optimal, the zero-mean/whiteness test was failed. Examining the innovations sequences of 5.12*c* and 5.12*d* indicates that during the transient phase of processor operation both suboptimal processors 1 and 2 do not track the true state, thereby creating a bias and nonwhite innovations. It is somewhat surprising how well suboptimal processor 2 tracks, but it does contain the dominant pole and begins tracking the measurement after 30 samples. Suboptimal processor 3 performs poorly. It should be noted that none of these processors was tuned for this analysis. This completes the example. We summarize the results as follows:

Criterion: $J = \text{trace}\ \ \tilde{P}(t|t)$

Models:

Signal:

$$x(t) = \begin{bmatrix} 0.97 & 0.1 \\ 0 & 0.87 \end{bmatrix} x(t-1) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t-1) + \begin{bmatrix} 1 \\ 1 \end{bmatrix} w(t-1)$$

$$x^1(t) = \begin{bmatrix} 0.97 & 0 \\ 0 & 0.87 \end{bmatrix} x^1(t-1) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t-1) + \begin{bmatrix} 1 \\ 1 \end{bmatrix} w(t-1)$$

$$x^2(t) = 0.97x^2(t-1) + 100u(t-1) + w(t-1)$$

$$x^3(t) = 0.87x^3(t-1) + 100u(t-1) + w(t-1)$$

Measurement:

$$y(t) = [2 \ \ 2]x(t) + v(t)$$

$$y(t) = 2x(t) + v(t)$$

Noise: $w \sim \mathcal{N}(0, 0.0001)$ and $v \sim \mathcal{N}(0, 0.01)$
Algorithm: $\hat{x}(t|t) = \hat{x}(t|t-1) + K(t)e(t)$
Quality: $\tilde{P}(t|t) = (I - K(t)C'(t))\tilde{P}(t|t-1)$

From this example it is clear that covariance analysis methods can be used to predict the performance of various processor designs. However, it should also be noted that it may be possible to "tune" a suboptimal processor and still obtain acceptable results. In fact, as in Figure 5.12, if a Monte Carlo approach is to be used, the *MBP* should be tuned first and then analyzed accordingly.

Once the design and analysis are completed, then the estimator should be applied to actual measurement data. Usually failure of the processor to track is caused by model mismatches between reality and processor models. If performance of the processor is not acceptable, then better models must be developed or "fit" to the measurement data.

So we see that the design of a *MBP* is an iterative trial-and-error process in practice. The most difficult, time-consuming, and expensive part of the methodology is developing the models, especially if the process under investigation is complicated. In fact the design of a *MBP* can be a very simple process, if the models are simple. Therefore the entire procedure is dominated by the modeling effort required. Recall from the introduction describing the tradeoffs between modeling and required processor accuracy. This completes the discussion of *MBP* methodology.

## 5.8 *MBP* EXTENSIONS

In this section we discuss extensions of the linear *MBP*. By extensions we mean how the linear discrete processor can be modified to include other problems of high applicability. First, we consider the case of correlated measurement and process noise that leads to the "prediction-form" (only) of the *MBP*. This extension is directly related to innovations model discussed in Chapter 4. Next, we discuss the colored noise cases where process and measurement noise are time-correlated and when the measurement noise can also be characterized by a Gauss-Markov model. Finally, we close the section investigating the effect of unmodeled *bias* and how to incorporate it into the *MBP* for optimal performance.

### 5.8.1 Model-Based Processor: Prediction-Form

The prediction form of the *MBP* is easily obtained by substituting for all of the corrected state and error covariance estimates in the predictor-corrector form. If we advance the arguments one time sample and substitute the state correction equation of Table 5.1 into the prediction relations, then using Eq. (5.20) we obtain

$$\begin{aligned}\hat{x}(t+1|t) &= A(t)\hat{x}(t|t) + B(t)u(t) \\ &= A(t)\left[\hat{x}(t|t-1) + R_{xe}(t)R_{ee}^{-1}(t)e(t)\right] + B(t)u(t) \end{aligned} \quad (5.101)$$

or

$$\hat{x}(t+1|t) = A(t)\hat{x}(t|t-1) + B(t)u(t) + A(t)R_{xe}(t)R_{ee}^{-1}(t)e(t) \tag{5.102}$$

Defining the *predicted gain* as

$$K_p(t) := A(t)R_{xe}(t)R_{ee}^{-1}(t) = A(t)\tilde{P}(t|t-1)C'(t)R_{ee}^{-1}(t) \tag{5.103}$$

and substituting into Eq. (5.101), we obtain the *state prediction equation*

$$\hat{x}(t+1|t) = A(t)\hat{x}(t|t-1) + B(t)u(t) + K_p(t)e(t) \tag{5.104}$$

expressed entirely in terms of the predicted state and error covariance estimates.

Using the same approach for the error covariance relations, we start with the prediction equation and substitute for the error covariance correction of Table 5.1

$$\begin{aligned}\tilde{P}(t+1|t) &= A(t)\tilde{P}(t|t)A'(t) + R_{ww}(t) \\ &= A(t)\left((I - R_{xe}(t)R_{ee}^{-1}(t)C(t))\tilde{P}(t|t-1)\right)A'(t) + R_{ww}(t)\end{aligned} \tag{5.105}$$

Expanding this equation, we have

$$\begin{aligned}\tilde{P}(t+1|t) &= A(t)\tilde{P}(t|t-1)A'(t) \\ &\quad -A(t)R_{xe}(t)R_{ee}^{-1}(t)C(t)\tilde{P}(t|t-1)A'(t) + R_{ww}(t)\end{aligned}$$

Substituting for $R_{xe}(t)$ and inserting the identity $I = R_{ee}(t)R_{ee}^{-1}(t)$, we obtain

$$\begin{aligned}\tilde{P}(t+1|t) &= A(t)\tilde{P}(t|t-1)A'(t) - A(t)\tilde{P}(t|t-1)C'(t)R_{ee}^{-1}(t) \\ &\quad \times R_{ee}(t)R_{ee}^{-1}(t)C(t)\tilde{P}(t|t-1)A'(t) + R_{ww}(t)\end{aligned}$$

Identifying the expression for the predicted gain, we have the final result for the *predicted state error covariance* known as the *discrete Riccati equation*

$$\tilde{P}(t+1|t) = A(t)\tilde{P}(t|t-1)A'(t) - K_p(t)R_{ee}(t)K_p'(t) + R_{ww}(t) \tag{5.106}$$

We summarize the prediction form of the *MBP* in Table 5.7.

The *innovations model* discussed in Chapter 2 is already in the "prediction form" of the *MBP*, since Eq. (5.104) is the state propagation relation of the model when $\hat{x}(t|t-1) \to \hat{x}(t)$. Therefore

$$\hat{x}(t+1) = A(t)\hat{x}(t) + B(t)u(t) + K_p(t)e(t) \tag{5.107}$$

By definition with $\hat{x}(t|t-1) \to \hat{x}(t)$,

$$y(t) = C(t)\hat{x}(t) + e(t) \tag{5.108}$$

is its corresponding measurement model.

So we see that the innovations model evolves directly from the prediction-form of the *MBP*.

**Table 5.7. State-Space *MBP* (Kalman Filter) Algorithm (Prediction-Form)**

| | |
|---|---|
| Prediction | |
| $\hat{x}(t+1\|t) = A(t)\hat{x}(t\|t-1) + B(t)u(t) + K_p(t)e(t)$ | (State prediction) |
| $\tilde{P}(t+1\|t) = A(t)\tilde{P}(t\|t-1)A'(t) - K_p(t)R_{ee}(t)K_p'(t) + R_{ww}(t)$ | (Covariance prediction) |
| Innovation | |
| $e(t) = y(t) - \hat{y}(t\|t-1) = y(t) - C(t)\hat{x}(t\|t-1)$ | (Innovation) |
| $R_{ee}(t) = C(t)\tilde{P}(t\|t-1)C'(t) + R_{vv}(t)$ | (Innovation covariance) |
| Gain | |
| $K_p(t) = A(t)\tilde{P}(t\|t-1)C'(t)R_{ee}^{-1}(t)$ | (Gain or weight) |
| Initial conditions | |
| $\hat{x}(0\|0), \quad \tilde{P}(0\|0)$ | |

This completes the subsection, next we see how this form can be used in applications when the noise sources are correlated.

### 5.8.2 Model-Based Processor: Colored Noise

In many applications the noise sources in the Gauss-Markov model are correlated or equivalently colored, rather than white. In this subsection we show how the *MBP* can be extended to solve this problem. First, we consider the case where the process and measurement noise sequences are time-correlated, that is,

$$\text{cov}\,(w(t), v(k)) = R_{wv}(t)\delta(t-k) \tag{5.109}$$

In deriving the prediction equation for the *MBP*, we used the fact that $w(t)$ was uncorrelated with the past $\hat{w}(t|T) = E\{w(t)|Y(T)\} = 0$ for $t \geq 0$, but this constraint is no longer valid when Eq. (5.109) is true. Substituting the equivalent innovations for $y(t)$, we can express the conditional mean as

$$\hat{w}(t|t) = E\{w(t)|Y(t)\} = E\{w(t)|Y(t-1)\} + E\{w(t)|e(t)\} \tag{5.110}$$

using the orthogonal decomposition property of the measurement space. Since $w(t)$ is orthogonal to $Y(t-1)$, the first term is null. From minimum variance estimation, we have that

$$\hat{w}(t|t) = E\{w(t)|e(t)\} = R_{we}(t)R_{ee}^{-1}(t)e(t) \tag{5.111}$$

Using the orthogonality properties of $w$ and Eq. (5.30), we can express the cross-covariance as

$$R_{we}(t) = E\{w(t)e'(t)\} = E\{w(t)v'(t)\} + E\{w(t)\tilde{x}'(t|t-1)\}C'(t) = R_{wv}(t) \tag{5.112}$$

since the last term is null. Thus the predicted state estimate, taking into account the process and measurement noise time-correlation, is now given by

$$\begin{aligned}\hat{x}(t+1|t) &= E\{x(t+1)|Y(t)\} = E\{A(t)x(t) + B(t)u(t) + w(t)|Y(t)\} \\ &= A(t)E\{x(t)|Y(t)\} + B(t)u(t) + E\{w(t)|Y(t)\} \\ &= A(t)\hat{x}(t|t) + B(t)u(t) + R_{wv}(t)R_{ee}^{-1}(t)e(t)\end{aligned} \tag{5.113}$$

where we have used the results of Eqs. (5.111) and (5.112). Now substituting for the corrected state estimate as before in Eq. (5.101), we write the prediction-form of this equation as

$$\hat{x}(t+1|t) = A(t)\left(\hat{x}(t|t-1) + R_{xe}(t)R_{ee}^{-1}(t)e(t)\right) + Bu(t) + R_{wv}(t)R_{ee}^{-1}(t)e(t) \tag{5.114}$$

This equation can be combined as

$$\hat{x}(t+1|t) = A(t)\hat{x}(t|t-1) + Bu(t) + \left(A(t)R_{xe}(t) + R_{wv}(t)\right)R_{ee}^{-1}(t)e(t) \tag{5.115}$$

We recognize a new definition for the *predicted gain* in the time-correlated noise case as

$$K_p(t) := \left(A(t)R_{xe}(t) + R_{wv}(t)\right)R_{ee}^{-1}(t) \tag{5.116}$$

which leads to the *MBP* prediction form of Table 5.7 with this definition of $K_p(t)$ of Eq. (5.116) inserted in both the state and error covariance relations.

Another case of interest occurs when the noise sources are not white but colored. Colored noise usually occurs when the dynamics of the noise sources are *not* wideband relative to the system dynamics. That is, they can be represented by a linear system driven by white noise. We limit our discussion to measurement noise dynamics and refer the interested reader to [1], [5], [20] for process noise dynamics.

We recall from the *representation theorem* of Chapter 4 that any correlated process can be modeled by driving a linear system with white noise (see Figure 5.13). This implies that the measurement system model can now be represented by

$$y(t) = C(t)x(t) + z(t) \tag{5.117}$$

where $z(t)$ is the output of a correlated noise (measurement instrument) system defined by the following Gauss-Markov representation

$$\begin{aligned}v(t) &= A_v(t-1)v(t-1) + B_v(t-1)m(t-1) + r(t-1) \\ z(t) &= C_v(t)v(t) + n(t)\end{aligned} \tag{5.118}$$

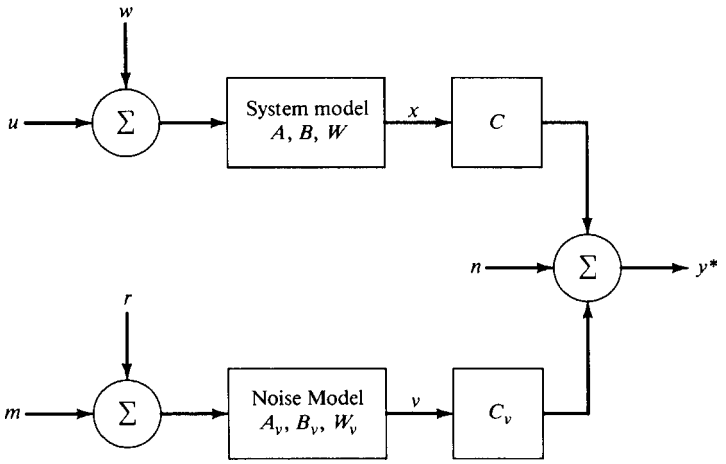



**Figure 5.13.** Colored measurement noise model.

where $r \sim \mathcal{N}(0, R_{rr}(t))$ and $n \sim \mathcal{N}(0, R_{nn}(t))$. This colored noise model of Eq. (5.118) can be *augmented* into the original Gauss-Markov model by defining the following state, measurement, input and noise vectors by

$$x^*(t) := [x'(t) \mid v'(t)]', \quad y^*(t) := [y'(t) \mid z'(t)]', \quad u^*(t) := [u'(t) \mid m'(t)]',$$
$$w^*(t) := [w'(t) \mid r'(t)]'$$

for $x^* \in R^{(N_x+N_v)}$, $u^* \in R^{(N_u+N_m)}$, and $w^* \in R^{(N_w+N_r)}$

Using these definitions and both models we obtain the *augmented colored noise* Gauss-Markov model as

$$\begin{aligned} x^*(t) &= A^*(t-1)x^*(t-1) + B^*(t-1)u^*(t-1) + w^*(t-1) \\ y^*(t) &= C^*(t)x^*(t) + n(t) \end{aligned} \tag{5.119}$$

The matrices defined in this augmented model are given by

$$A^*(t-1) = \begin{bmatrix} A(t-1) & 0 \\ 0 & A_v(t-1) \end{bmatrix}, \quad B^*(t-1) = \begin{bmatrix} B(t-1) & 0 \\ 0 & B_v(t-1) \end{bmatrix}$$
$$C^*(t) = [C(t) \mid C_v(t)], \quad R^*_{ww}(t) = \text{diag}\,[R_{ww}(t)\ ,\ R_{rr}(t)],$$
$$n \sim \mathcal{N}(0, R_{nn}(t))$$

The optimal *MBP* in this case is identical to that presented in Table 5.1, with the exception that the augmented model is used in the processor. Consider the following example:

***Example 5.7*** We again use our *RC*-circuit tuning example of the previous chapter with Gauss-Markov model

$$x(t) = 0.97x(t-1) + 100u(t-1) + w(t-1)$$
$$y(t) = 2x(t) + z(t)$$

with $u(t) = 0.03$ and $R_{ww} = 0.0001$. Suppose that our measurement system can be modeled with the following dynamics:

$$v(t) = 0.90v(t-1) + r(t-1)$$
$$z(t) = 1.5v(t) + n(t)$$

with $R_{rr} = 0.004$ and $R_{nn} = 1.0$. Thus the "augmented" Gauss-Markov model for this problem is given by

$$x^*(t) = \begin{bmatrix} 0.97 & 0 \\ 0 & 0.90 \end{bmatrix} x^*(t-1) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u^*(t-1) + w^*(t-1)$$

$$y^*(t) = [2 \quad 1.5]x^*(t) + n(t)$$

We simulate this system with correlated measurements (state 2) and initial state $x(0) = [2.5 \quad 2.0]$ using the *SSPACK_PC* software [17]. The simulation is shown in Figure 5.14 with the noisy and true (mean) measurements shown in *a* through *c*. Notice that the dynamics of the instrument are faster (sharper transient decay) than the dynamics of the *RC*-circuit, but there is still some filtering by the instrument as well as gain. Compare the faster decay time of the true measurement with the original *RC*-circuit example of discussed previously. Note also the effects of the process noise on both simulated states.

We see that the measurement and second state lie within the prescribed 95% confidence limits, but the first state does not (6% instead of 5%). We can run the simulator again with different random seed values until the limits are satisfied. The augmented *MBP* for this problem is given by the preceding Gauss-Markov model matrices with the following "guesses" for the processor parameters:

$$\hat{x}(0|0) = [2.5 \quad 2.0] \quad \tilde{P}(0|0) = \text{diag}\,[0.01 \quad 0.01] \quad R^*_{ww} = \text{diag}\,[0.0001 \quad 0.004]$$

The *MBP* simulation using *SSPACK_PC* is shown in Figure 5.15. The state and measurement estimates and corresponding errors are shown in Figure 5.15*a* through *c*. Here we see that the *MBP* tracks very well. The estimation errors and innovations indicate that initially the errors are larger because of transients, but eventually converge. Note that the sample statistics in each case match those predicted by the processor. Also note that the innovations variance predicted by the processor

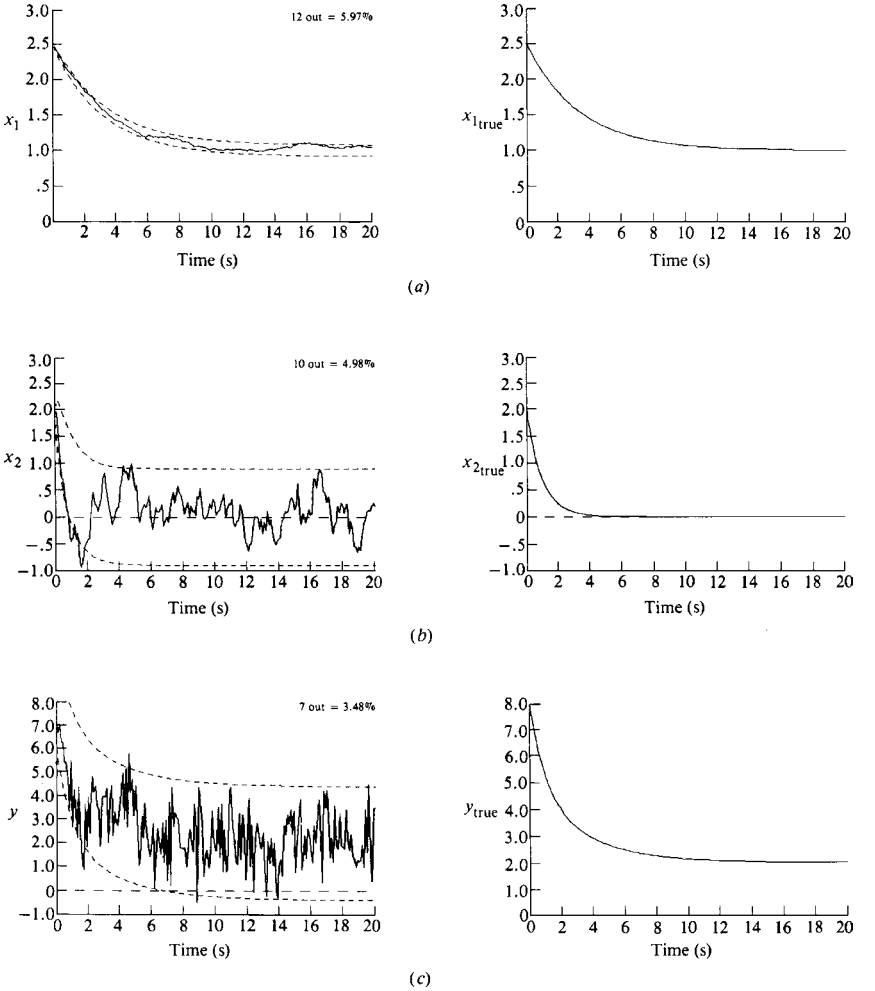



**Figure 5.14.** Gauss-Markov simulation for colored measurement noise. (*a*) Simulated and true state 1. (*b*) Simulated and true state 2. (*c*) Simulated and true measurement.

underestimates the true variance (10% out) primarily because of the initial errors in tracking. The innovations are "reasonably" zero-mean ($0.013 < 0.16$) and white (4% out) in this case, indicating a properly tuned processor. The *WSSR* indicates a nonwhite process, since the threshold is exceeded. This is due to initial tracking errors of the processor. This can be corrected by starting the *WSSR* test after the transient has occurred. So we see that the colored measurement noise can easily be handled using the standard state-space *MBP* algorithm by merely augmenting the coloring filter states. This completes the example. We summarize the results as follows:

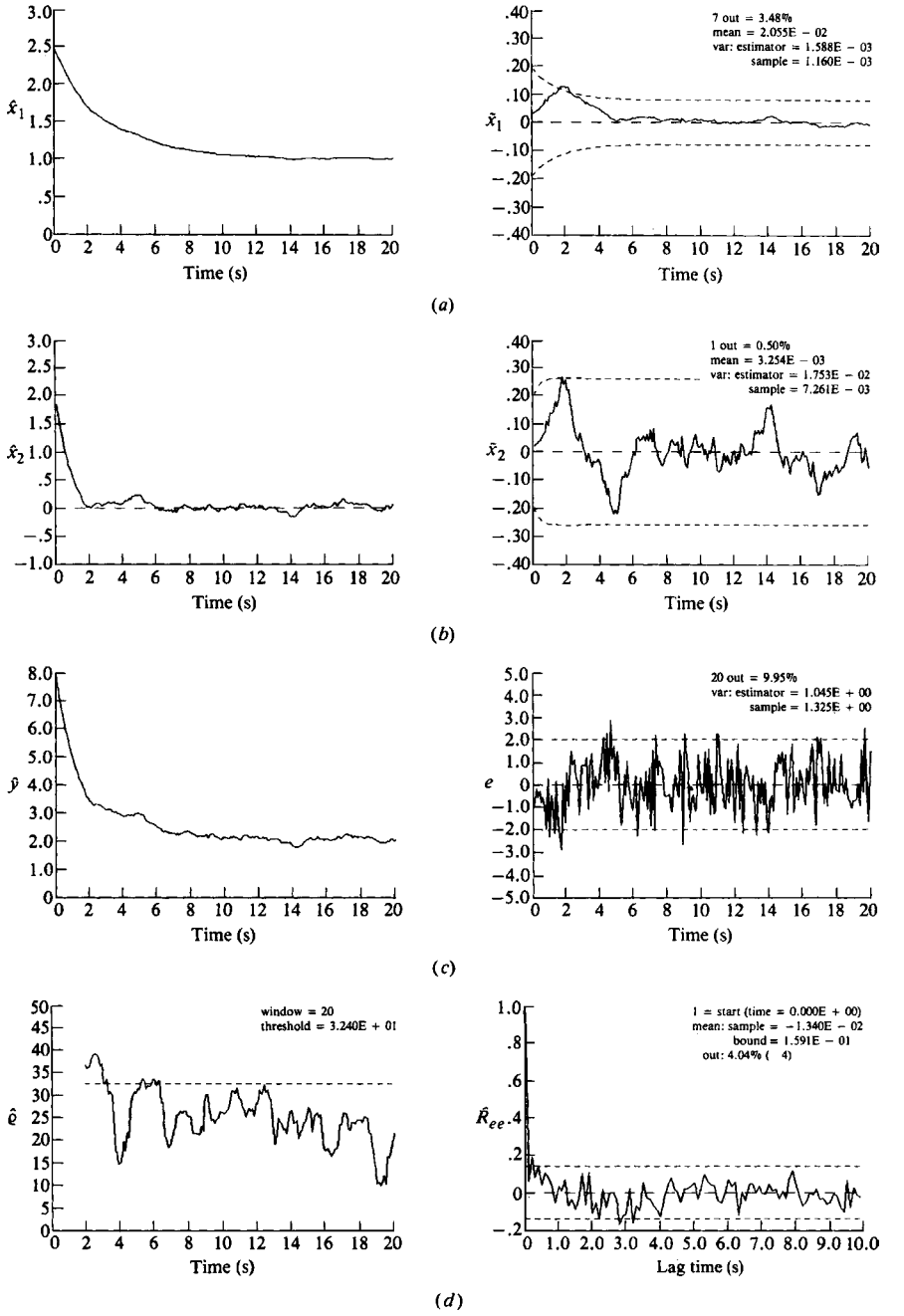



**Figure 5.15.** Augmented *MBP* for colored measurement noise problem. (*a*) Estimated state 1 and error. (*b*) Estimated state 2 and error. (*c*) Filtered measurement and error (innovations). (*d*) WSSR and zero-mean/whiteness tests.

Criterion: $J = \text{trace } \tilde{P}(t|t)$

Models:

Signal: $x(t) = 0.97x(t-1) + 100u(t-1) + w(t-1)$

Measurement: $y(t) = 2.0x(t-1) + z(t)$

Noise:

$$v(t) = 0.90v(t-1) + r(t-1)$$

$$z(t) = 1.5v(t) + n(t)$$

$$w \sim \mathcal{N}(0, 0.0001), \quad r \sim \mathcal{N}(0, 0.004), \quad n \sim \mathcal{N}(0, 1.0)$$

Initial: $\hat{x}(0|0) = [2.5 \;\; 2.0], \quad \tilde{P}(0|0) = \text{diag } [0.01 \;\; 0.01]$

Algorithm: $\hat{x}^*(t|t) = \hat{x}^*(t|t-1) + K(t)e(t)$

Quality: $\tilde{P}(t|t) = [I - K(t)C(t)]\tilde{P}(t|t-1)$

Note that if augmenting the state vector imposes intolerable constraints on the algorithm, than a clever scheme of measurement differencing has been developed to solve this problem [1]. This completes the subsection on colored noise. Next we consider how the linear *MBP* can be used to deal with the problem of *bias* in the data.

### 5.8.3 Model-Based Processor: Bias Correction

In this subsection we discuss the extension of the *MBP* to the case of bias in the data. Bias correction is an important property that the *MBP* can easily incorporate. In this situation, when the bias is known a priori, the process can be generalized by incorporating a known "bias model" in the original Gauss-Markov representation so that

$$\begin{aligned} x(t) &= A(t-1)x(t-1) + B(t-1)u(t-1) + w(t-1) \\ y(t) &= C(t)x(t) + D(t)u(t) + v(t) \end{aligned} \tag{5.120}$$

Known biases can then easily be modeled if they enter into the states with the "$Bu$-term" or if they enter in the measurement through the "$Du$-term" in the Gauss-Markov model. The model changes in the measurement mean to incorporate the additional term, that is,

$$m_y(t) = C(t)m_x(t) + D(t)u(t) \tag{5.121}$$

The covariance expressions remain the same, since by definition the mean is removed.

Next let us investigate the effect of the known bias in terms of the *MBP* directly. As seen in Table 5.1, the prediction equations already incorporate the known input,

but the measurement equation through the innovation sequence does not. The innovations must account for the *bias correction* "$Du$-term." Therefore we have that

$$e(t) = y(t) - \hat{y}(t|t-1) - D(t)u(t) \tag{5.122}$$

and the bias has been removed. So we see that the known bias can easily be incorporated into the Gauss-Markov representation. With this incorporation, the *MBP* is then capable of removing its effects. Consider the following example that demonstrates the performance of the processor.

***Example 5.8*** We investigate the *RC*-circuit problem and perform a simulation with a "known bias" by simply incorporating a feedthrough $Du$-term with $D = 100$, therefore creating a constant bias of 3 volts. We show the results of designing the *MBP* with and without the term in Figures 5.16 and 5.17 for comparison. The processor with bias term included is clearly tuned as observed by the estimated

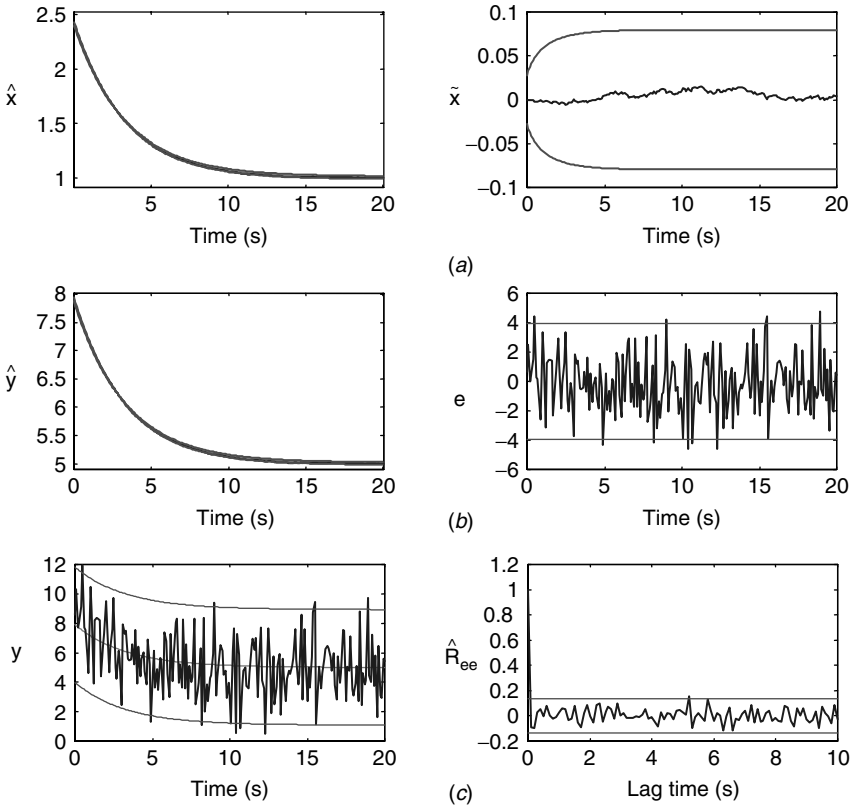



**Figure 5.16.** Bias-*corrected MBP* for *RC*-circuit problem. (*a*) Estimated state and error. (*b*) Filtered measurement and error (innovations). (*c*) Measurement and zero-mean (0.16 < 0.37) and whiteness tests (0.99% out).

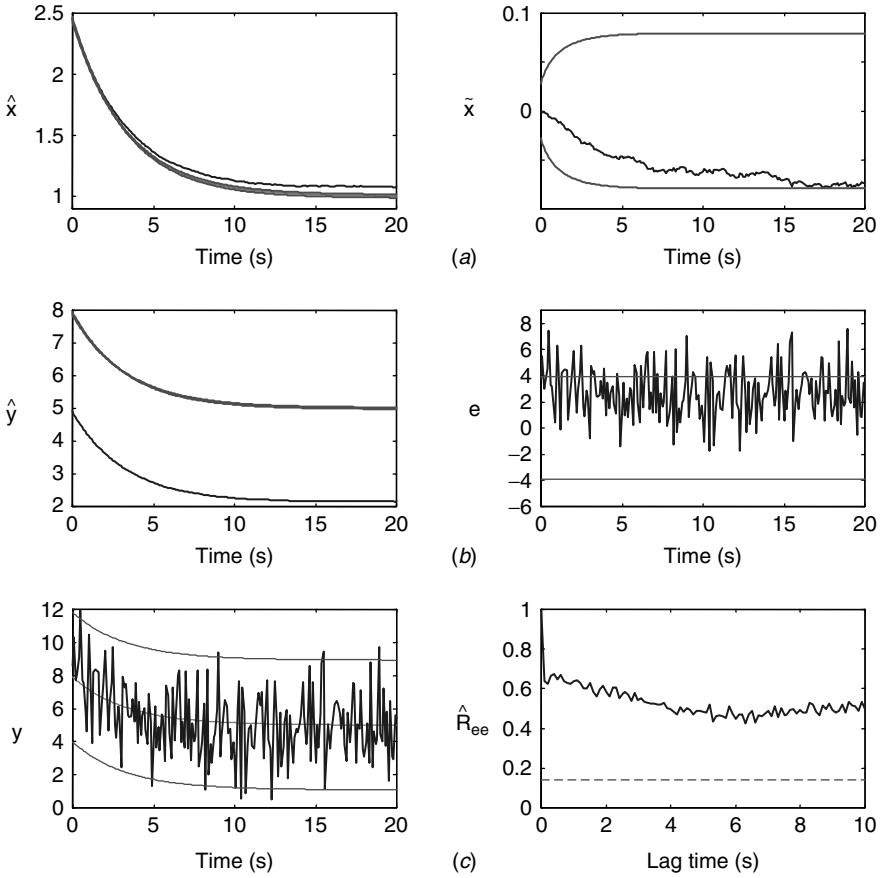



**Figure 5.17.** Biased *MBP* for *RC*-circuit problem. (*a*) Estimated state and error. (*b*) Filtered measurement and error (innovations). (*c*) Measurement and zero-mean ($2 > 0.37$) and whiteness tests (26% out).

state and associated error in Figure 5.16*a*, along with the estimated measurement and innovation in Figure 5.16*b*. The innovations are zero-mean ($0.16 < 0.37$) and white (0.99% out) as shown in Figure 5.16*c*. On the other hand, if the bias is not included, the results are clearly inferior. Both the estimated state and error are biased as shown in Figure 5.17*a*. Note that they are even more pronounced in the estimated measurement and accompanying innovations in Figure 5.17*b*. The innovations are clearly biased ($2 > 0.37$) and nonwhite (26% out) as depicted in Figure 5.17*c*.

Criterion: $J(t|t) = \text{trace } \tilde{P}(t|t)$

Models:

Measurement: $y(t) = 2x(t) + 100u(t) + v(t)$

Signal: $x(t) = 0.97x(t-1) + 100u(t-1) + w^*(t-1)$
Noise: $w^* \sim \mathcal{N}(0, 10^{-6}), \quad v \sim \mathcal{N}(0, 4)$
Algorithm: $\hat{x}(t|t) = \hat{x}(t|t-1) + K(t)e(t)$
Quality: $P(t|t) = [1 - 2K(t)]\tilde{P}(t|t-1)$

This completes the example. Next we consider two important problems that can be solved using the *MBP*.

## 5.9 *MBP* IDENTIFIER

The *MBP* can easily be formulated to solve the well-known (see [18]) *system identification problem*:

> **GIVEN** a set of noisy measurements $\{y(t)\}$ and inputs $\{u(t)\}$. **FIND** the minimum (error) variance estimate $\hat{\Theta}(t|t)$ of $\Theta(t)$, a vector of unknown parameters of a linear system.

The identification problem is depicted in Figure 5.18. The scalar linear system is given by the *autoregressive model with exogenous inputs* (*ARX*) of Chapter 2 as

$$A(q^{-1})y(t) = B(q^{-1})u(t) + e(t) \tag{5.123}$$

where $A$ and $B$ are polynomials in the backward shift operator $q^{-i}$ and $\{e(t)\}$ is white, that is,

$$\begin{aligned} A(q^{-1}) &= 1 + a_1 q^{-1} + a_2 q^{-2} + \cdots + a_{N_a} q^{-N_a} \\ B(q^{-1}) &= b_o + b_1 q^{-1} + b_2 q^{-2} + \cdots + b_{N_b} q^{-N_b} \end{aligned} \tag{5.124}$$

Thus Eq. (5.123) can be expressed as

$$\begin{aligned} &y(t) - a_1 y(t-1) - a_2 y(t-2) - \cdots - a_{N_a} y(t-N_a) \\ &\quad = b_o u(t) + b_1 u(t-1) + b_2 u(t-2) + \cdots + b_{N_b} u(t-N_b) + e(t) \end{aligned} \tag{5.125}$$

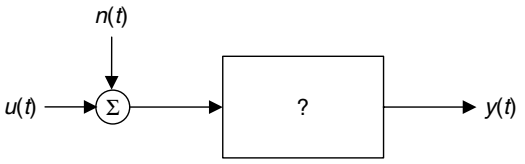



**Figure 5.18.** The identification problem.

When this system is driven by random inputs, the models are given by the *ARX* model. We must convert this model to the state-space/measurement-system framework required by the *MBP*. The measurement model can be expressed as

$$y(t)=[-y(t-1)\cdots-y(t-N_a)|u(t)\cdots u(t-N_b)]\begin{bmatrix} a_1 \\ \vdots \\ a_{N_a} \\ -- \\ b_o \\ \vdots \\ b_{N_b} \end{bmatrix}=c'(t)\Theta(t)+v(t) \tag{5.126}$$

where

$$c'(t)=[-y(t-1)\cdots-y(t-N_a)|u(t)\cdots u(t-N_b)] \qquad \text{for } v\sim\mathcal{N}(0,R_{vv})$$

and

$$\Theta(t)=[a_1\cdots a_{N_a}|b_0\cdots b_{N_b}]'$$

which represents the measurement equation in the *MBP* formulation. The parameters can be modeled as constants with uncertainty $w(t)$; that is, the state-space model used in this formulation takes the form

$$\Theta(t)=\Theta(t-1)+w(t) \tag{5.127}$$

where $w\sim\mathcal{N}(0,R_{ww})$. We summarize the *MBP* used as an identifier in Table 5.8. Thus the *MBP* provides the minimum variance estimates of the parameters of the *ARX* model. Consider the following example.

***Example 5.9*** Suppose that we have the *RC*-circuit tuning problem

$$x(t)=0.97x(t-1)+100u(t-1)+w(t-1)$$

and

$$y(t)=2.0x(t)+v(t)$$

where $x(0)=2.5$, $R_{ww}=10^{-6}$, and $R_{vv}=10^{-12}$. The "identification" model is given by

$$\begin{aligned}\Theta(t)&=\Theta(t-1)+w^*(t)\\ y(t)&=c'(t)\Theta(t)+v(t)\end{aligned}$$

where $c'(t)=[-y(t-1)|u(t)u(t-1)]$, $\Theta'(t)=[a_1|b_0\,b_1]$, $v\sim\mathcal{N}(0,10^{-12})$, and $w^*\sim\mathcal{N}(0,10^{-6})$. Here $w^*(t)$ is different from the process noise of the Gauss-Markov model.

**Table 5.8. Model-Based Identifier**

Prediction

$$\hat{\Theta}(t|t-1) = \hat{\Theta}(t-1|t-1) \quad \text{(Parameter prediction)}$$

$$\tilde{P}(t|t-1) = \tilde{P}(t-1|t-1) + R_{ww}(t-1) \quad \text{(Covariance prediction)}$$

Innovation

$$e(t) = y(t) - \hat{y}(t|t-1) = y(t) - c'(t)\hat{\Theta}(t|t-1) \quad \text{(Innovation)}$$

$$R_{ee}(t) = c'(t)\tilde{P}(t|t-1)c(t) + R_{vv}(t) \quad \text{(Innovation covariance)}$$

Gain

$$K(t) = \tilde{P}(t|t-1)c(t)R_{ee}^{-1}(t) \quad \text{(Kalman gain or weight)}$$

Correction

$$\hat{\Theta}(t|t) = \hat{\Theta}(t|t-1) + K(t)e(t) \quad \text{(Parameter correction)}$$

$$\tilde{P}(t|t) = [I - K(t)c'(t)]\tilde{P}(t|t-1) \quad \text{(Covariance correction)}$$

Initial conditions

$$\hat{\Theta}(0|0) \quad \text{and} \quad \tilde{P}(0|0)$$

where

$$c'(t) = [-y(t-1)\cdots - y(t-N_a)|u(t)\cdots u(t-N_b)]$$

$$\Theta(t) = [a_1 \cdots a_{N_a}|b_0 \cdots b_{N_b}]'$$

Using the Gauss-Markov model and *SSPACK_PC* [17], we simulated the data with the specified variances. The performance of the model-based identifier is shown in Figure 5.19. The parameter estimates in Figure 5.19*a* and Figure 5.19*b* show that the estimator identifies the $a$ parameter in about 10 samples, but the $b_0$ does not converge at all in this time interval. The innovations are clearly zero-mean ($2.3 \times 10^{-4} < 2.8 \times 10^{-4}$) and white (0% lie outside), as depicted in Figure 5.19*c*. The measurement filtering property of the processor is shown in Figure 5.19*b*. This completes the example.

We summarize the results as follows:

Criterion: $J(t|t) = \text{trace } \tilde{P}(t|t)$

Models:

Signal: $\Theta(t) = \Theta(t-1) + w^*(t-1)$

Measurement: $y(t) = c'(t)\Theta(t) + v(t)$

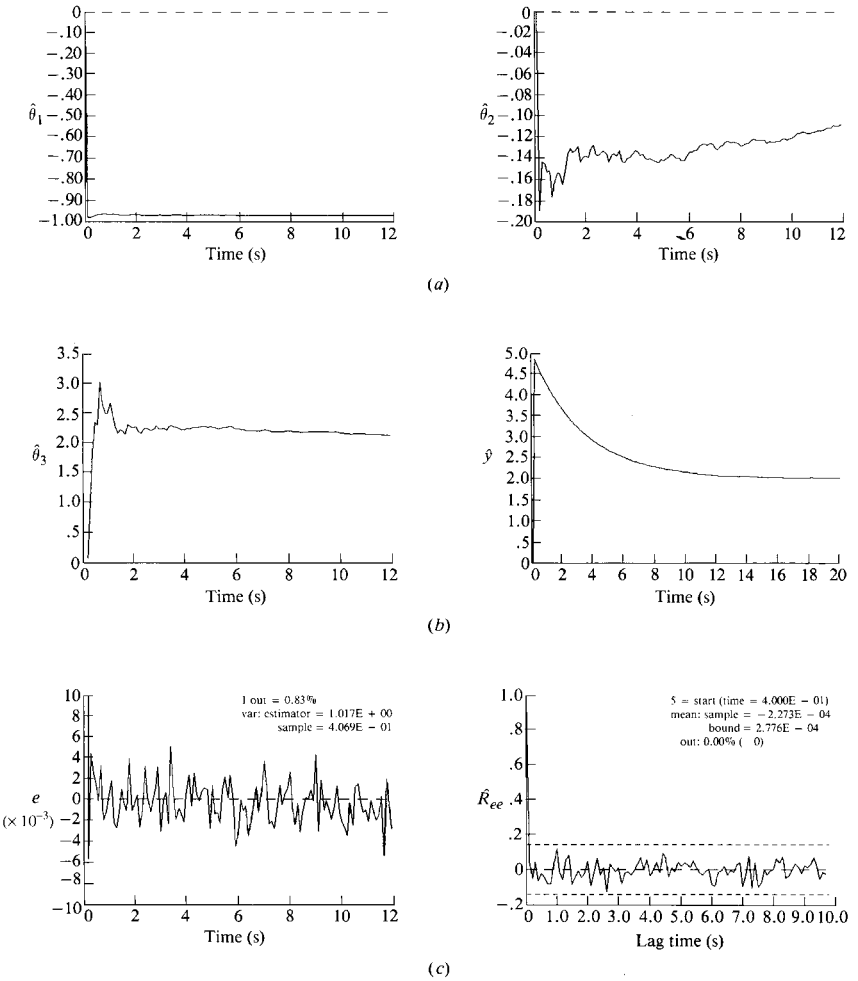



**Figure 5.19.** Model-based identifier output for tuning example. *(a)* Parameter estimates for $a_1$ and $b_0$. *(b)* Parameter estimates $b_1$ and filtered measurement. *(c)* Prediction error (innovation) and whiteness test.

| | |
|---|---|
| Noise: | $w^* \sim \mathcal{N}(0, 10^{-6}) \quad v \sim \mathcal{N}(0, 10^{-12})$ |
| Initial state: | $\hat{\Theta}(0\|0) = [0.0\ 0.0\ 0.0], \quad \tilde{P}(0\|0)$ |
| Algorithm: | $\hat{\Theta}(t\|t) = \hat{\Theta}(t\|t-1) + K(t)e(t)$ |
| Quality: | $P(t\|t) = [I - K(t)c'(t)]\tilde{P}(t\|t-1)$ |

This completes the section on the extension of the *MBP* as a recursive identification algorithm. Note that the model-based identifier is identical to the recursive least-squares algorithm of Chapter 3, with the exception that the parameters can

include process noise $w^*(t)$. Although this difference appears minimal, it actually becomes very significant and necessary to avoid estimator divergence (see [18], [19], or [10] for details). In the next section we see how the filter can be extended to estimate an unknown input as well as state.

## 5.10 *MBP* DECONVOLVER

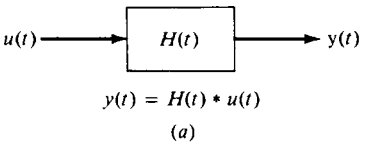
In this section we consider extending the *MBP* algorithm to solve the problem of estimating an unknown input from data that have been "filtered." This problem is called *deconvolution* in signal processing literature and occurs commonly in seismic and speech processing [21], [22] as well as transient problems [23] (see also Section 4.3.2).

In many measurement systems it is necessary to deconvolve or estimate the input to an instrument given that the data are noisy. The basic deconvolution problem is depicted in Figure 5.20*a* for deterministic inputs $\{u(t)\}$ and outputs $\{y(t)\}$. The problem can be simply stated as follows:

> **GIVEN** the impulse response $H(t)$ of a linear system and outputs $\{y(t)\}$. **FIND** the unknown input $\{u(t)\}$ over some time interval.

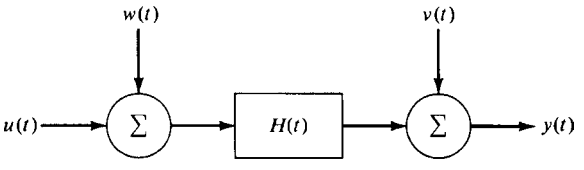
In practice, this problem is complicated by the fact that the data are noisy and impulse response models are uncertain. Therefore, a more pragmatic view of the problem would account for these uncertainties. The uncertainties lead us to define the *stochastic deconvolution problem* shown in Figure 5.20 *b*. This problem can be stated as follows:

> **GIVEN** a model of the linear system $H(t)$ and discrete noisy measurements $\{y(t)\}$. **FIND** the minimum (error) variance estimate of the input sequence $\{u(t)\}$ over some time interval.




$y(t) = H(t) * u(t)$

(*a*)




$y(t) = [H(t) * (u(t) + w(t))] + v(t)$

(*b*)

**Figure 5.20.** Deconvolution problem.

The solution to this problem using the *MBP* algorithm again involves developing a model for the input and augmenting the state vector.

Suppose that we utilize the standard discrete Gauss-Markov model of Section 2.11.4 and define the following Gauss-Markov model for the input signal:

$$u(t) = F(t-1)u(t-1) + n(t-1) \tag{5.128}$$

where $n \sim \mathcal{N}(0, R_{nn}(t))$. The augmented Gauss-Markov model is given by $X_u := [x|u]\prime$ and $w_u' := [w \mid n]$:

$$X_u(t) = A_u(t-1)X_u(t-1) + w_u(t-1)$$

and

$$y(t) = C_u(t)X_u(t) + v(t)$$

The matrices in the augmented model are given by

$$A_u(t-1) = \begin{bmatrix} A(t-1) & B(t-1) \\ 0 & F(t-1) \end{bmatrix},$$

$$R_{w_u} = \begin{bmatrix} R_{ww}(t-1) & R_{wn}(t-1) R_{nw}(t-1) & R_{nn}(t-1) \end{bmatrix}$$

and

$$C_u(t) = [C(t) \mid 0]$$

This model can be simplified by choosing $F = I$; that is, $u$ is a piecewise constant. This model becomes valid if the system is oversampled (see [14] for details). The *MBP* for this problem is the standard algorithm of Sec. 5.1 with the augmented matrices. The augmented *MBP* is sometimes called the *Schmidt-Kalman filter*. Schmidt (see [4]) developed a suboptimal method to estimate parameters and decrease the computational burden of the augmented system. Consider the following example to illustrate this extension.

***Example 5.10*** For the simulated measurements we again use the *RC*-circuit tuning example with a higher *SNR* as shown in Figure 5.21*(c)*. Here the model is with $Bu \rightarrow u$, since $B$ is assumed unknown:

$$x(t) = 0.97x(t-1) + u(t-1) + w(t-1)$$
$$y(t) = 2.0x(t) + v(t)$$

where $x(0) = 2.5$, $R_{ww} = 10^{-4}$, and $R_{vv} = 4 \times 10^{-3}$. The Schmidt-Kalman filter *MBP* for this problem using the "augmented" model and a piecewise constant approximation for the unknown input is given by

$$X_u(t) = \begin{bmatrix} 0.97 & 1.0 \\ 0 & 1.0 \end{bmatrix} X_u(t-1) + \begin{bmatrix} w(t-1) \\ n(t-1) \end{bmatrix}$$

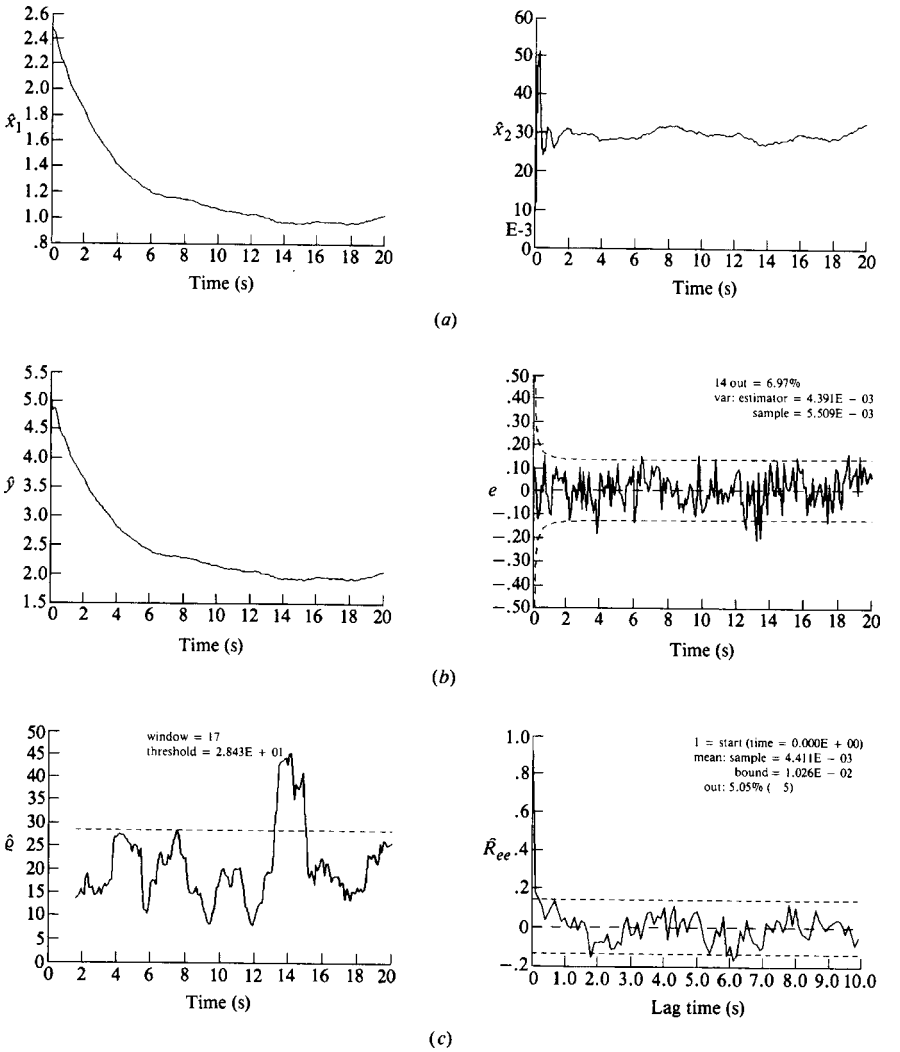



**Figure 5.21.** The Schmidt-Kalman filter *MBP* deconvolver. *(a)* Estimated state and input. *(b)* Filtered measurement and innovation. *(c)* WSSR and whiteness test.

and

$$y(t) = [2\ 0]X_u(t) + v(t)$$

The results of the *MBP* simulation using *SSPACK_PC* [17] are shown in Figure 5.21. Here the state estimates are shown in Figure 5.21*a*, along with the filtered measurement and innovations in Figure 5.21*b*. The filter tracks the input quite reasonably after an initial transient as indicated. The optimality of the estimator is confirmed by the corresponding zero-mean ($0.0045 < 0.01$) and white (5%

lie outside) innovations sequence. This completes the example. We summarize the results as follows:

| | |
|---|---|
| Criterion: | $J(t\|t) = \text{trace}\ \tilde{P}(t\|t)$ |
| Models: | |
| Signal: | $x(t) = 0.97x(t-1) + u(t-1) + w(t-1)$ |
| Measurement: | $y(t) = 2.0x(t) + v(t)$ |
| Noise: | $w^* \sim \mathcal{N}(0, 10^{-4}),\ n \sim \mathcal{N}(0.5 \times 10^{-7}),\ v \sim \mathcal{N}(0, 4 \times 10^{-3})$ |
| Initial state: | $\hat{x}(0\|0) = [2.55\ 0.0]', \quad \tilde{P}(0\|0) = \text{diag}[5 \times 10^{-2}, 5 \times 10^{-2}]$ |
| Algorithm: | $\hat{X}_u(t\|t) = \hat{X}_u(t\|t-1) + K(t)e(t)$ |
| Quality: | $\tilde{P}(t\|t) = [I - K(t)C(t)]\tilde{P}(t\|t-1)$ |

## 5.11 STEADY-STATE *MBP* DESIGN

In this section we discuss a special case of the state-space *MBP*—the steady-state design. Here the data are assumed to be stationary, leading to a time-invariant state-space model and under certain conditions a constant error covariance and corresponding gain. We first develop the processor and then show how it is precisely equivalent to the classical Wiener filter design. In filtering jargon this processor is called the *steady-state MBP* (Kalman filter) ([20], [25]).

### 5.11.1 Steady-State *MBP*

We briefly develop the steady-state *MBP* technique in contrast to the usual recursive time-step algorithms, we have discussed throughout this chapter. By *steady state* we mean that the processor has embedded time invariant state-space model parameters; that is, the underlying model is defined by the set of parameters, $\Sigma = \{A, B, C, R_{ww}, R_{vv}, \hat{x}(0|0), \tilde{P}(0|0)\}$. We state without proof the fundamental theorem (see [1], [5], [25] for details). If we have a stationary process implying the time-invariant system, $\Sigma$, and additionally the system is completely controllable and observable and stable (poles of A lie within the unit circle) with $\tilde{P}(0|0) > 0$, then the *MBP* is *asymptotically stable*. What this theorem means from a pragmatic point of view is that as time increases (to infinity in the limit), the initial error covariance is forgotten as more and more data are processed and so the computation of $\tilde{P}(t|t)$ is computationally stable. Furthermore these conditions imply that

$$\lim_{t\to\infty} \tilde{P}(t|t) = \lim_{t\to\infty} \tilde{P}(t|t-1) \to \tilde{P}_{\text{SS}}\ \text{(a constant)} \tag{5.129}$$

Therefore, with this relation true, it implies that we can define a *steady-state gain* associated with this covariance as

$$\lim_{t\to\infty} K(t) \to K_{\text{SS}}\ \text{(a constant)} \tag{5.130}$$

Let us construct the corresponding *steady-state MBP* using the correction equation of the state estimator with the steady-state gain, that is,

$$\hat{x}(t|t) = \hat{x}(t|t-1) + K_{SS}e(t) = (I - K_{SS}C)\hat{x}(t|t-1) + K_{SS}y(t) \tag{5.131}$$

but since

$$\hat{x}(t|t-1) = A\hat{x}(t-1|t-1) + Bu(t-1) \tag{5.132}$$

we have by substituting into Eq. (5.131) that

$$\hat{x}(t|t) = (I - K_{SS}C)A\hat{x}(t-1|t-1) + (I - K_{SS}C)Bu(t-1) + K_{SS}y(t) \tag{5.133}$$

with the corresponding *steady-state gain* given by

$$K_{SS} = \tilde{P}_{SS}CR_{ee}^{-1} = \tilde{P}_{SS}C\left(C\tilde{P}_{SS}C' + R_{vv}\right)^{-1} \tag{5.134}$$

Examining Eq. (5.133) more closely, we see that using the state-space model, $\Sigma$ and known input sequence, $\{u(t)\}$, we can process the data and extract the corresponding state estimates. The key to the steady-state *MBP* is calculating $K_{SS}$ which in turn implies that we must calculate the corresponding steady-state error covariance. This calculation can be accomplished efficiently by combining the prediction and correction relations of Table 5.1. We have that

$$\tilde{P}(t|t-1) = A\left(I - \tilde{P}(t|t-1)C(C\tilde{P}(t|t-1)C' + R_{vv})^{-1}\right)A' + R_{ww} \tag{5.135}$$

which in steady state becomes

$$\tilde{P}_{SS} = A\left(I - \tilde{P}_{SS}C(C\tilde{P}_{SS}C' + R_{vv})^{-1}\right)A' + R_{ww} \tag{5.136}$$

There are a variety of efficient methods to calculate the steady-state error covariance and gain [1], [25]. However, a brute-force technique is simply to run the standard predictor-corrector algorithm (implemented in *UD* [26] sequential form of Appendix B) until the $\tilde{P}_{SS}$ and $K_{SS}$ converge to constant matrices. Once they converge it is only necessary to run the algorithm again to process the data using $\tilde{P}(0|0) = \tilde{P}_{SS}$, and the corresponding steady-state gain will be calculated directly. This is not the most efficient method to solve the problem, but it clearly does not require the development of a new algorithm.

We summarize the steady-state *MBP* in Table 5.9. We note from the table that the steady-state covariance/gain calculations depend on the model parameters, $\Sigma$, not on the data, which implies that they can be *calculated* prior to processing the actual data. In fact the steady-state processor can be thought of as a simple (multi-input/multi-output) digital filter which is clear if we abuse the notation slightly to write

$$\hat{x}(t+1) = (I - K_{SS}C)A\hat{x}(t) + (I - K_{SS}C)Bu(t) + K_{SS}y(t+1) \tag{5.137}$$

**Table 5.9. Steady-State-Space *MBP* (Kalman Filter) Algorithm**

| | |
|---|---|
| Covariance | |
| $\tilde{P}_{\rm SS} = A\left(I - \tilde{P}_{\rm SS}C(C\tilde{P}_{\rm SS}C' + R_{vv})^{-1}\right)A' + R_{ww}$ | (Steady-state covariance) |
| Gain | |
| $K_{\rm SS} = \tilde{P}_{\rm SS}C\left(\tilde{C}P_{\rm SS}C' + R_{vv}\right)^{-1}$ | (Steady-state gain) |
| Correction | |
| $\hat{x}(t\|t) = (I - K_{\rm SS}C)A\hat{x}(t-1\|t-1) + (I - K_{\rm SS}C)Bu(t-1) + K_{\rm SS}y(t)$ | (State estimate) |
| Initial conditions | |
| $\hat{x}(0\|0), \quad \tilde{P}(0\|0)$ | |

For its inherent simplicity compared to the full time-varying processor, the steady-state *MBP* is desirable and adequate in many applications, but realize that it will be *suboptimal* in some cases during the initial transients of the data. However, if we developed the model from first principles, then the underlying physics has been incorporated in the *MBP* yielding a big advantage over non physics-based designs. Note also that once the steady-state MBP is calculated, the multivariable spectral factorization (Wiener filter) problem is solved automatically as developed in the next section and Section 8.3. This completes the discussion of the steady-state *MBP*. Next let us re-examine our *RC*-circuit problem.

***Example 5.11*** Once more we use the tuned *RC*-circuit problem of the Section 5.5 and replace the time-varying gain shown in Figure 5.4*d* with their steady-state values obtained by running the algorithm of Table 5.1 (error covariance and gain) until they converge to constants. The steady-state values estimated are

$$\tilde{P}_{\rm SS} = 1.646 \times 10^{-3} \quad \text{and} \quad K_{\rm SS} = 8.23 \times 10^{-4}$$

Next the algorithm is executed with the initial error covariance set to its steady-state value ($\tilde{P}(0|0) = \tilde{P}_{\rm SS}$) giving the steady-state gain as shown in Figure 5.22*c*. The performance of the processor is not as good as the optimal (see Figure 5.5) during the transient stages of the processor. The errors are larger, for instance, the number of state estimate samples exceeding the predicted confidence limits are greater ($10\% > 7\%$) and the state estimation errors are larger as shown in 5.22*a*. The zero-mean/whiteness test of the innovations in 5.22*b* indicates that both *MBP* perform roughly the same with the optimal performing slightly better with smaller mean ($0.11 < 0.16$) and whiteness ($0\% < 1\%$) of the samples exceeding the bound.

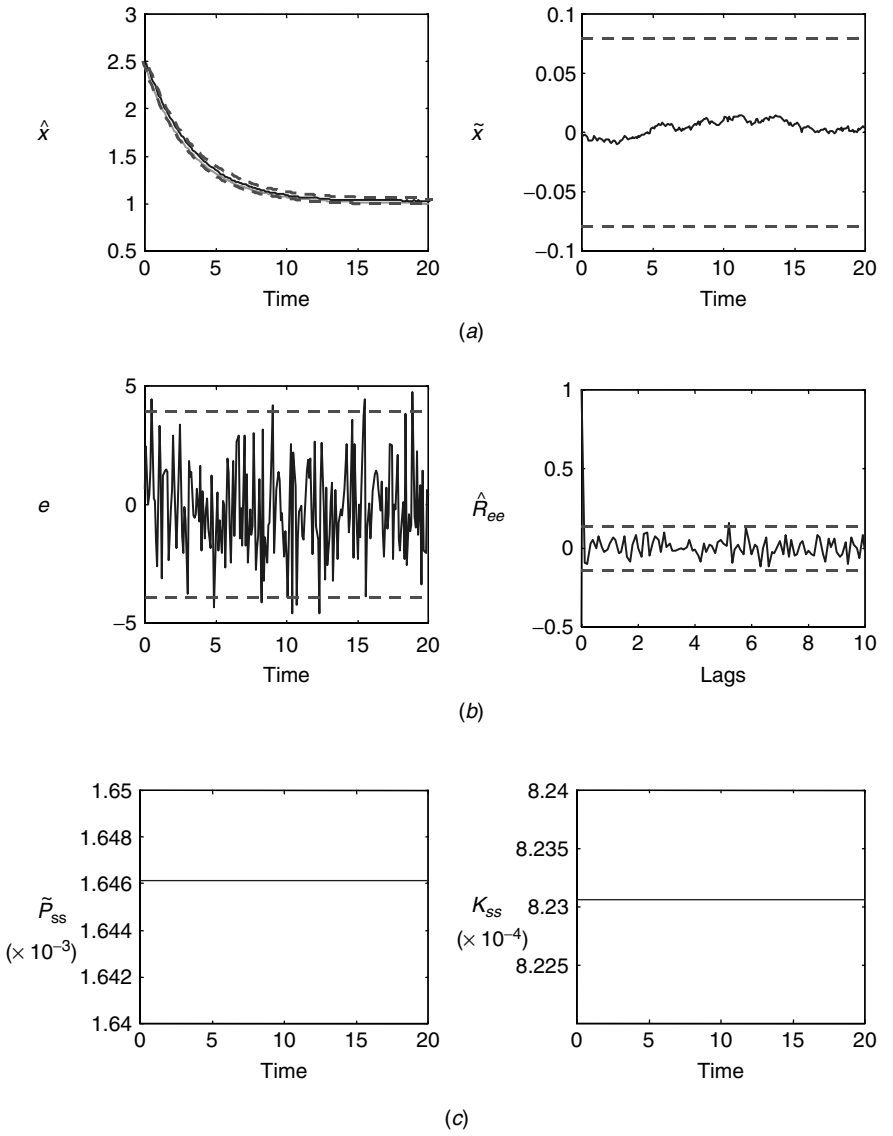



**Figure 5.22.** Steady-state MBP design for *RC*-circuit problem. (*a*) Estimated state (10% out) and estimation error. (*b*) Innovations and zero-mean/whiteness tests ($0.002 < 0.004$) and (1% out). (*c*) Steady-state error covariance (0.00165) and gain (0.000823).

Examining the innovations sequence itself, it is clear there is a problem for the steady-state processor during the initial stage of the algorithm. So we see that there is a performance penalty for using a steady-state processor in lieu of the optimal time-varying *MBP*; however, the computational advantages can be significant for large numbers of states.

This completes the example, next we investigate the steady-state *MBP* and its relation to the Wiener filter.

### 5.11.2 Steady-State *MBP* and the Wiener Filter

In this subsection we show the relationship between the Wiener filter and its state-space counterpart, the Kalman filter. Detailed proofs of these relations are available for both the continuous and discrete cases [20]. Our approach is to state the Wiener solution and then show that the steady-state Kalman filter provides a solution with all the necessary properties. We use frequency-domain techniques to show the equivalence. The time-domain approach would be to use the batch innovations solution discussed earlier. We choose the frequency domain for historical reasons, since the classical Wiener solution has more intuitive appeal.

Recall that the Wiener filter solution in the frequency domain can be solved by spectral factorization, since

$$K(z) = [S_{sy}(z)S_{yy}^{-1}(z^{-})]_{cp}S_{yy}^{-1}(z^{+}) \tag{5.138}$$

where $K(z)$ has all its poles and zeros within the unit circle. The classical approach to Wiener filtering can be accomplished in the frequency domain by factoring the power spectral density (*PSD*) of the measurement sequence; that is,

$$S_{yy}(z) = K(z)K'(z^{-1}) \tag{5.139}$$

The factorization is unique, stable, and minimum-phase (see [1] for proof).

Next we must show that the *steady-state Kalman filter* or the *innovations model* (ignoring the deterministic input) given by

$$\begin{aligned} \hat{x}(t) &= A\hat{x}(t-1) + Ke(t-1) \\ y(t) &= C\hat{x}(t) + e(t) = \hat{y}(t) + e(t) \end{aligned} \tag{5.140}$$

where $e$ is the zero-mean, white innovations with covariance $R_{ee}$, is stable and minimum-phase and therefore, in fact, the *Wiener solution*. The "transfer function" of the innovations model is defined as

$$T(z) := \frac{Y(z)}{E(Z)} = C(zI - A)^{-1}K \tag{5.141}$$

Let us calculate the measurement covariance of Eq. (5.140):

$$R_{yy}(k) = \mathrm{Cov}[y(t+k)y(t)] = R_{\hat{y}\hat{y}}(k) + R_{\hat{y}e}(k) + R_{e\hat{y}}(k) + R_{ee}(k) \tag{5.142}$$

where $\hat{y}(t) := C\hat{x}(t)$. Taking $z$ transforms, we obtain the measurement *PSD* as

$$S_{yy}(z) = S_{\hat{y}\hat{y}}(z) + S_{\hat{y}e}(z) + S_{e\hat{y}}(z) + S_{ee}(z) \tag{5.143}$$

Using the linear system relations of Chapter 2, we see that

$$\begin{aligned} S_{yy}(z) &= CS_{\hat{x}\hat{x}}(z)C' = T(z)S_{ee}(z)T'(z^{-1}); \quad S_{ee}(z) = R_{ee} \\ S_{\hat{y}e}(z) &= CS_{\hat{x}e}(z) = T(z)S_{ee}(z); \quad \text{and} \quad S_{e\hat{y}}(z) = S_{ee}(z)T'(z^{-1}) \end{aligned} \tag{5.144}$$

Thus the measurement *PSD* is given by

$$S_{yy}(z) = T(z)S_{ee}(z)T'(z^{-1}) + T(z)S_{ee}(z) + S_{ee}(z)T'(z^{-1}) + S_{ee}(z) \tag{5.145}$$

Since $S_{ee}(z) = R_{ee}$ and $R_{ee} \geq 0$, the following factorization always exists as

$$R_{ee} = R_{ee}^{1/2}(R_{ee}')^{1/2} \tag{5.146}$$

Thus from Eq. (5.146), $S_{yy}(z)$ of Eq. (5.145) can be written as

$$S_{yy}(z) = [T(z)R_{ee}^{1/2} \;\; R_{ee}^{1/2}] \begin{bmatrix} (R_{ee}')^{1/2}T'(z^{-1}) \\ (R_{ee}')^{1/2} \end{bmatrix} := T_e(z)T_e'(z^{-1}) \tag{5.147}$$

which shows that the innovations model indeed admits a spectral factorization of the type desired. To show that $T_e(z)$ is the unique, stable, minimum-phase spectral factor, it is necessary to show that $|T_e(z)|$ has all its poles within the unit circle (stable). It has been shown (e.g., [1], [4], [8]), that $T_e(z)$ does satisfy these constraints. Therefore

$$T_e(z) = K(z)$$

is the Wiener solution.

It is interesting to note that there is an entire class of solutions [27] that evolves from the spectral factorization theory characterized by

$$S_{yy}(z) = \left[C(zI - A)^{-1}LN \;\; | \;\; N\right] \begin{bmatrix} N'L'(z^{-1}I - A')^{-1}C'] \\ --- \\ N' \end{bmatrix} \tag{5.148}$$

The unique Wiener solution is given by the relations

$$L = PC'(NN')^{-1} \quad \text{and} \quad (NN') = CPC' + R_{vv} \tag{5.149}$$

only when $P$ is either the minimum $P_*$ or maximum $P^*$ solution of the steady-state Riccati equation (see [27] for details); that is, $P_* < P < P^*$ and $L = K$ (gain) and $NN' = R_{ee}$ (innovations covariance) for $P = P^*$.

This completes the discussion on the equivalence of the steady-state Kalman filter and the Wiener filter. (See Sec. 8.3 for more details.)

## 5.12 CASE STUDY: *MBP* DESIGN FOR A STORAGE TANK

In this section we consider the design of a *MBP* for a storage tank containing a radioactive material—plutonium nitrate. In this study we design an optimal *MBP* capable of predicting the amount (mass) of plutonium present in the tank at any instant—this process is called material accounting ([24], [28]). It is required to estimate the plutonium mass at the end of each day. The tank is shown in Figure 5.23 with its various dynamics depicted. The level of solution in the tank at any time is attributed to the dynamics of evaporation of the water and nitric acid as well as radiolysis effects.

The process model is developed relating the solution *mass* $m(t)$ to the *plutonium concentration* $P(t)$, that is,

$$\frac{d}{dt}m(t) = -K_H - K_r m(t) P(t) \tag{5.150}$$

where

$m$ = the solution mass, kg
$K_H$ = the evaporation rate of the water and acid solution, kg/day
$K_r$ = the radiolysis constant, kg (water)/day per kg (Pu)
$P$ = the plutonium concentration kg (Pu)/kg (solution)

Since the change in plutonium concentration is very slow due to the radiolysis effects, we assume that it is constant, $P(t) = P$ for all $t$.

Next we develop the measurement model. A typical measurement system for a tank solution is the pneumatic bubbler shown in Figure 5.24. Neglecting second

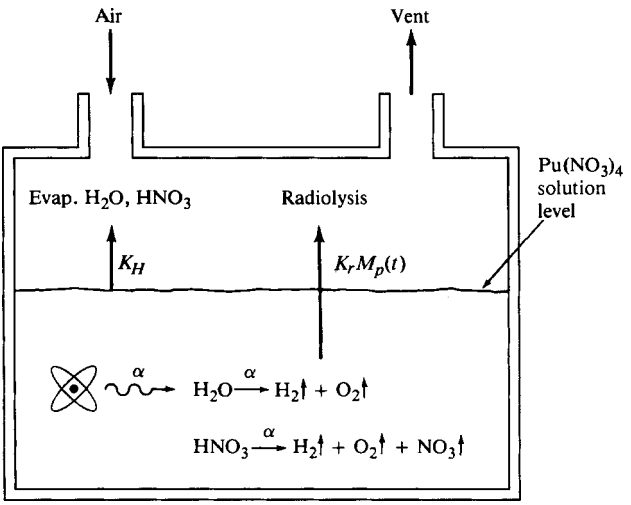



**Figure 5.23.** Plutonium nitrate storage tank problem.

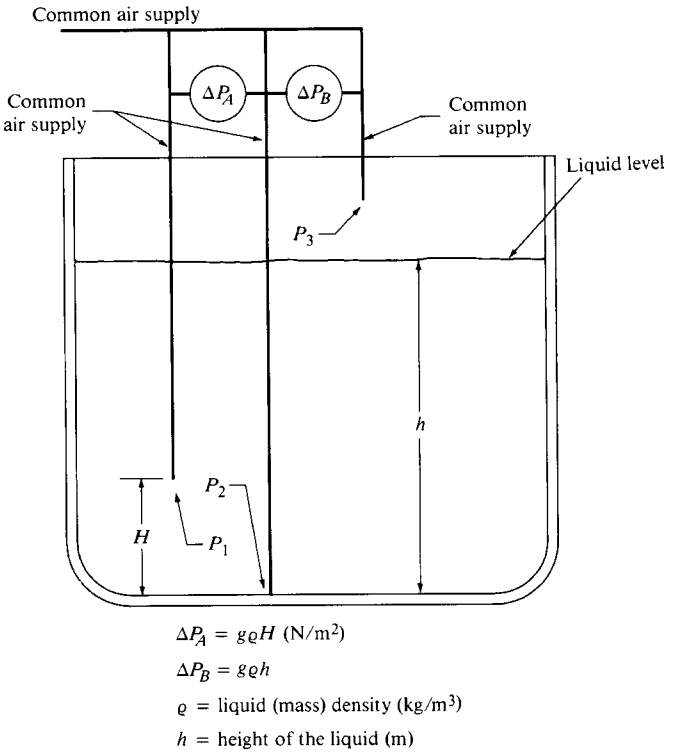



**Figure 5.24.** Pneumatic bubbler measurement system for storage tank.

order effects, the measurement model can be developed from pressure head density relations. The bubbler measures density $\rho$ and the level $h$ of liquid based on differential pressures. The pressure at the bubbler tubes is given by (see Figure 5.24)

$$p_1 = g\rho(t)\,(h(t) - H) + p_3 \quad \text{and} \quad p_2 = g\rho(t)h(t) + p_3 \tag{5.151}$$

where $p_1$, $p_2$, $p_3$, $H$, $h(t)$ are the respective pressures and heights shown in the figure and $\rho(t)$ and $g$ are the corresponding density and gravitational constant. It follows that the *pressure differentials* are

$$\triangle P_A = p_2 - p_1 = g\rho(t)H \quad \text{and} \quad \triangle P_B = p_2 - p_3 = g\rho(t)h(t) \tag{5.152}$$

Thus the bubbler measures (indirectly) the density and height from these pressure measurements. The gauges measure $\triangle P_A$ and $\triangle P_B$ to give

$$\rho(t) = \frac{\triangle P_A}{gH} \quad \text{and} \quad h(t) = \frac{\triangle P_B}{\rho(t)g} \tag{5.153}$$

We are interested in the solution mass at a given time, so the model is still not complete. This mass is related to the density and level by

$$m(t) = a\rho(t) + b\frac{\triangle P_B}{g} \quad \text{or} \quad \triangle P_B = \frac{g}{b}m(t) - \frac{a}{b}g\rho(t) \tag{5.154}$$

We now have our measurement model expressed in terms of the differential pressure readings $\triangle P_A$ and $\triangle P_B$.

We define the state vector as $x' := [m \quad \rho]$. Since we do not have a meaningful model to use for the relationship of density to the solution mass, we model it simply as a random walk

$$\frac{d}{dt}\rho(t) = w(t) \tag{5.155}$$

where $w$ is white with variance $R_{w_c w_c}$.

Summarizing the process and measurement models in state-space form, we have

$$\frac{d}{dt}\begin{bmatrix} m(t) \\ \\ \rho(t) \end{bmatrix} = \begin{bmatrix} -K_r P & 0 \\ \\ 0 & 0 \end{bmatrix}\begin{bmatrix} m(t) \\ \\ \rho(t) \end{bmatrix} + \begin{bmatrix} 1 \\ \\ 0 \end{bmatrix} u(t) + \begin{bmatrix} w_1(t) \\ \\ w_2(t) \end{bmatrix} \tag{5.156}$$

where $u$ is a step function of amplitude $-K_H$. The corresponding measurement model is

$$\begin{bmatrix} \triangle P_A \\ \\ \triangle P_B \end{bmatrix} = \begin{bmatrix} g/b & -(a/b)g \\ \\ 0 & gH \end{bmatrix}\begin{bmatrix} m(t) \\ \\ \rho(t) \end{bmatrix} + \begin{bmatrix} v_1(t) \\ \\ v_2(t) \end{bmatrix} \tag{5.157}$$

Finally, we must develop models for the process and measurement uncertainties. The process uncertainty can be adjusted using the simulation, but as a starting value we select $R_{w_c w_c} = I$, based on judgments of the density and accuracy of the model parameters. The measurements can be estimated from manufacturer's specifications of the measurement instrumentation. From the set of model parameters (see Table 5.10), the simulation model is given by

$$\frac{d}{dt}x(t) = \begin{bmatrix} -1.43 \times 10^{-5} & 0 \\ \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 1 \\ \\ 0 \end{bmatrix} u(t) + w(t)$$

and the measurement model by

$$y(t) = \begin{bmatrix} 29.8 & -0.623 \\ \\ 0 & 24.9 \end{bmatrix} x(t) + v(t)$$

**Table 5.10. Storage Tank Model Parameters**

| Parameter | Value | Units |
|---|---|---|
| $K_i$ | $-8.122 \times 10^{-5}$ | Kg (water)/day per kg ($Pu$) |
| $H_H$ | 0.5 | kg (solution)/day |
| $H$ | 2.54 | m |
| $a$ | 0.0209 | $m^3$ |
| $b$ | 0.328 | $m^2$ |
| $g$ | 9.8 | $m/s^2$ |
| $P$ | 0.173 | kg ($Pu$)/kg (solution) |
| $m_p$ | 1448 | $kg/m^3$ |
| $m_m$ | 983 | kg (solution) |
| $R_{vv}(1)$ | $5.06 \times 10^4$ | $N^2/m^4$ |
| $R_{vv}(2)$ | $1.4 \times 10^5$ | $N^2/m^4$ |

The uncertainty in the measurements is approximated from typical specifications and is corrected by tuning the simulation. Using the model parameter values from Table 5.10, we estimate the nominal $\triangle P_A$ and $\triangle P_B$ to be $3.6 \times 10^4$ and $3.0 \times 10^4$ $N/m^2$. Let us assume that pressure gauges of $5.0 \times 10^4$ and $3.0 \times 10^4$ are used to measure the differentials. We will assume a "worst-case" variance (precision) of a gauge is within $\pm 0.75\%$ of its full scale reading, that is,

$$\sigma(\text{ gauge }) = 0.75\% \text{ F.S.} \tag{5.158}$$

For our gauges we have that

$$\sqrt{R_{vv}(1,1)} = 0.0075(3.0 \times 10^4) = 225 \quad N/m^2$$
$$\sqrt{R_{vv}(2,2)} = 0.0075(5.0 \times 10^4) = 375 \quad N/m^2$$

Thus we have that the measurement covariance matrix is $R_{vv} = \text{diag}\,[5.06 \times 10^4 \;\; 1.4 \times 10^5]$ $(N^2/m^4)$.

We now have enough information to design the optimal *MBP*. We choose to implement a discrete rather than continuous-time system and use the matrix exponential series approximation available in *SSPACK_PC* [17] (an error tolerance of $10^{-6}$ gives six terms in series) with the resulting discrete-time matrices

$$A = \begin{bmatrix} 0.99 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 0.99 \\ 0 \end{bmatrix}$$

Thus the discrete *MBP* model with sampling interval of 0.1 *day* is

$$x(t) = \begin{bmatrix} 0.999 & 0 \\ 0 & 1 \end{bmatrix} x(t-1) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t-1) + w(t-1)$$
$$y(t) = \begin{bmatrix} 29.8 & -0.623 \\ 0 & 24.9 \end{bmatrix} x(t) + v(t)$$

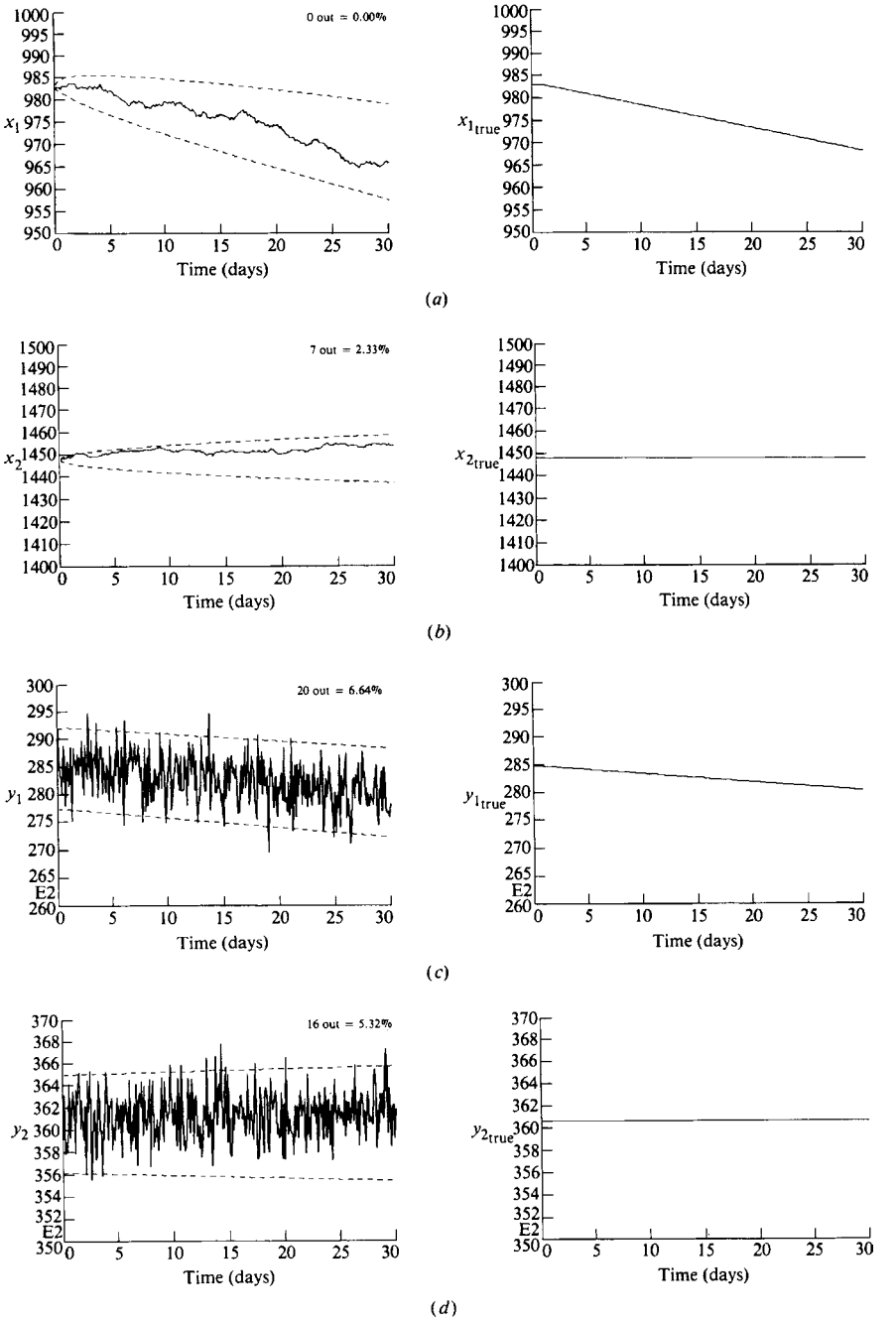



**Figure 5.25.** Plutonium nitrate storage tank Gauss-Markov simulation. (*a*) Simulated and true solution mass. (*b*) Simulated and true density. (*c, d*) Simulated and true differential pressure measurements.

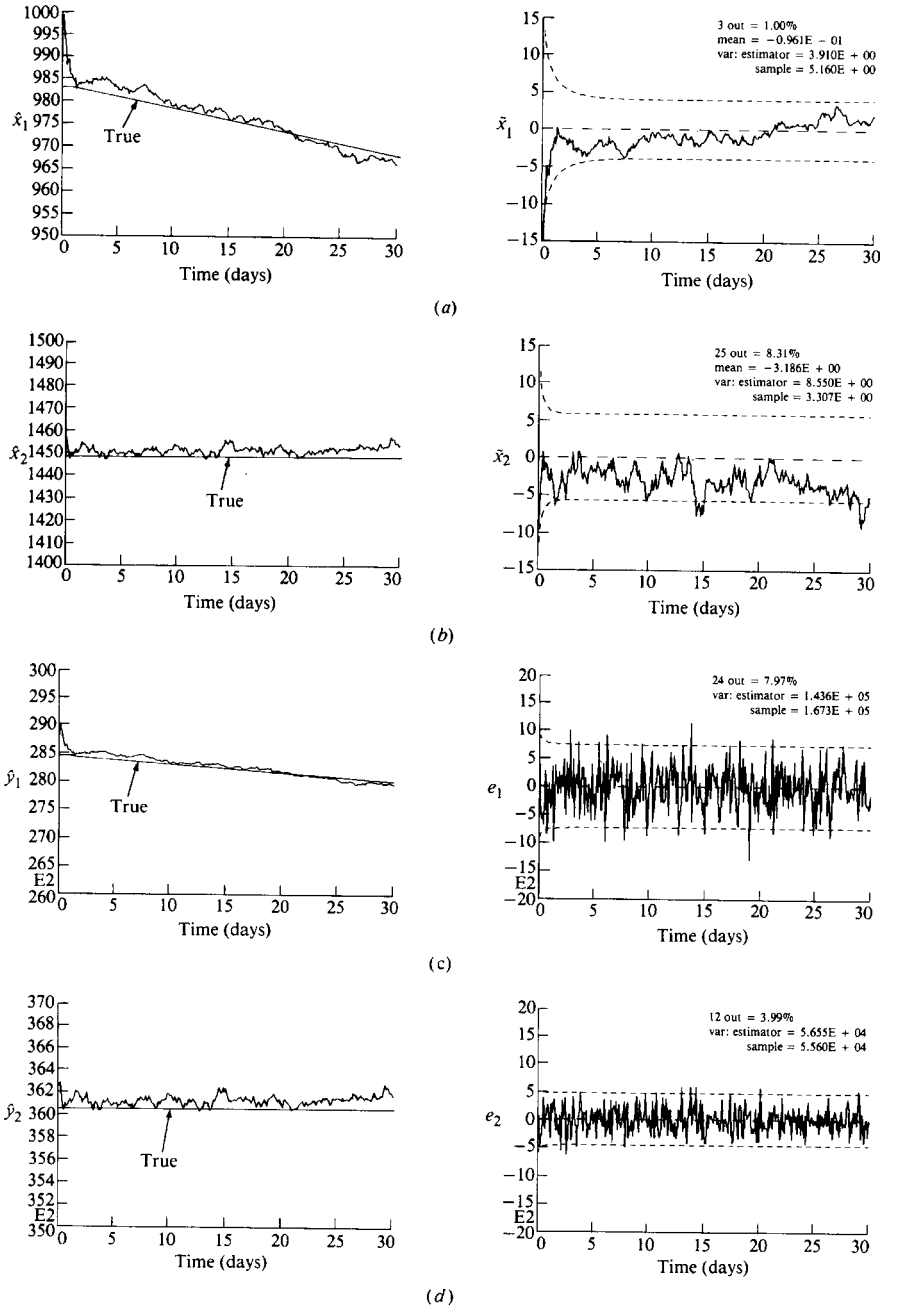



**Figure 5.26.** Plutonium nitrate storage tank MBP minimum variance design. (*a*) True and estimated solution mass and estimation error. (*b*) True and estimated density and estimation error. (*c, d*) True and filtered differential pressure measurements and innovations (error).

$R_{ww} = \text{diag}\ [10\ \ 10]$, $R_{vv} = \text{diag}\ [5.06 \times 10^4\ \ \ 1.4 \times 10^5]$, with initial conditions $\hat{x}(0|0) = [988\ \ \ 1455\ \ ]'$, and $\tilde{P}(0|0) = \text{diag}\ [0.01\ \ 0.01]$.

The *SSPACK_PC* [17] Gauss-Markov model was executed for a 30 day period. The results are shown in Figure 5.25. We see the simulated and true solution masses ($kg$ of solution) in Figure 5.25*a* and the density in Figure 5.25*b*. The noisy and true differential pressure measurements are shown in Figure 5.25*c* and *d*. The results are as expected with the mass decreasing slowly due to evaporation and radiolysis and the density staying constant. The differential gauge measurements are noisy.

The *MBP* was designed using the minimum (error) variance approach and the results are shown in Figure 5.26. The true and estimated solution mass is shown in 5.26*a* of the figure as well as the corresponding estimation error. After a brief transient phase the estimator begins tracking the true value as shown by the error curves (within $\pm 2\sigma$ confidence limits). Similar results hold for the density estimate shown in Figure 5.26*b*. In Figure 5.26*c* and Figure 5.26*d* we see the filtered and true differential pressure and innovations sequences. The innovations are zero-mean ($26 < 60$ and $19 < 33$) and white (2% and 2% outside the limits) as shown in Figure 5.27, indicating a successful minimum variance design with expected solution mass RMS error (square root of $\tilde{P}_{11}$) of approximately 2 kg. The *WSSR*

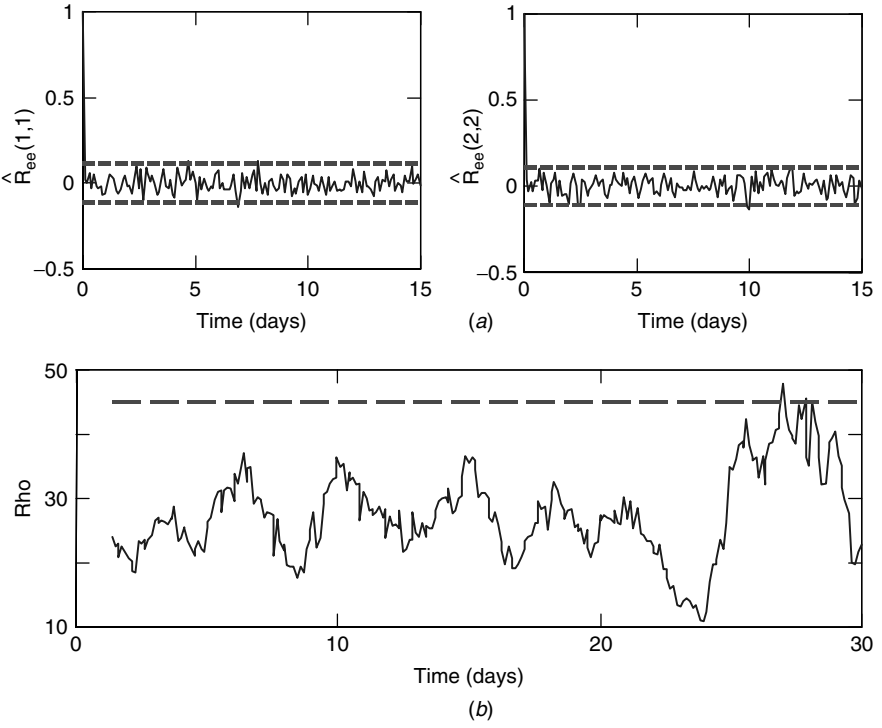



**Figure 5.27.** Plutonium nitrate storage tank MBP performance. (*a*) Zero-mean ($26 < 60$, $19 < 33$) and whiteness (2% out and 2% out) tests. (*b*) WSSR test (threshold = 45).

test is also shown in the figure. Here it actually exceeds the threshold indicating a slight bias in the estimates. The *MBP* design is complete and ready for application.

We summarize the results of this case study as follows:

Criterion: $J = \text{trace } \tilde{P}(t|t)$

Models:

Signal: $\dfrac{d}{dt}m(t) = -K_H - K_r m(t)P(t) + w_1(t)$

Measurements: $\dfrac{d}{dt}\rho(t) = w_2(t)$

$\triangle P_A = g\rho(t)H + v_1(t)$ and $\triangle P_B = g\rho(t)h(t) + v_2(t)$

Noise: $w \sim \mathcal{N}(0, R_{ww})$ and $v \sim \mathcal{N}(0, R_{vv})$

Algorithm: $\hat{x}(t|t) = \hat{x}(t|t-1)) + K(t)e(t)$

Quality: $\tilde{P}(t|t) = (I - K(t)C(t))\tilde{P}(t|t-1)$

This completes the case study on the estimation of plutonium in a storage tank.

## 5.13 SUMMARY

We developed here the concept of the state-space model-based processor or equivalently the Kalman filter both theoretically and heuristically. We derived it theoretically using the innovations as well as Bayesian approaches giving insight to the processor structure as well as the properties of the innovations that were exploited in "tuning." We showed how the processor can be tuned according to the procedure developed in [2] and applied it to an *RC*-circuit problem. Next we demonstrated the sources of model mismatch and how they impact the *MBP* by observing their effect on the innovations sequence. We discussed the overall philosophy of processor design and described the underlying methodology and how it can be used. Then we discussed extensions of the *MBP* for prediction (only) leading to the innovations model of Chapter 2. We showed how the processor can be used to solve the identification and deconvolution problems. We showed how bias could be easily compensated for in MBP design. We developed the steady-state *MBP* and showed its equivalence to the classical Wiener filter of Chapter 4. Finally we discussed the development of a *MBP* for estimation of mass in a storage tank case study.

## *MATLAB* NOTES

*SSPACK_PC* [17] is a third-party toolbox in *MATLAB* that can be used to design model-based signal processors. The development environment (see Appendix C for

details) consists of a GUI-driven Supervisor that enables the choice of model set (linear or nonlinear) as well as techniques (Kalman filter, extended Kalman (nonlinear) filter, Kalman identifier, etc.) and processors to implement the various models and perform statistical analysis of the resulting designs. This package incorporates the *linear MBP* algorithms discussed in this chapter—all implemented in the *UD*-factorized form [26] for stable and efficient calculations. The package enables the signal processor to simulate data using the Gauss-Markov model and its equivalents (see Section 2.12) and approximate Gauss-Markov (nonlinear) models while designing model-based signal (state-space) and/or parameter estimators. Steady-state (Wiener filter) designs can also be easily be achieved from the package. The versatility of the state-space model-based signal processing approach allows for many equivalent applications after performing the required transformation to state-space form (see Chapter 2).

In terms of the linear state-space models of this chapter, *SSPACK_PC* has embedded the following algorithms. Linear discrete-time systems using the Gauss-Markov model are provided for the *time-invariant* case using the simulator (**SSDSIM**) and *MBP* (**SSDEST**) while the *time-varying* case is developed in the simulator (**SSTSIM**) and corresponding processor (**SSTEST**). Continuous-time systems are converted seamlessly for both simulation and estimation using the (**SSCTOD**) transformations. The linear *time-invariant* innovations models are also available for simulation and processing using the (**SSISIM**) simulator and (**SSIEST**) *MBP*. Gauss-Markov models can be converted to the innovations form using the (**GMTOINV**), and visa versa, using (**INVTOGM**) routines again embedded seamlessly in the GUI-driven environment. Finally the system identification problem can be solved using the equivalent *ARX* model-based (**SSDID**) algorithm that automatically converts the solution to the Gauss-Markov canonical form discussed in Chapter 2. The heart of the package is the command or GUI-driven postprocessor (**SSPOST**) that is used to analyze and display the results of simulation and processing (see **http://www.techni-soft.net** for more details).

## REFERENCES

1. B. D. Anderson and J. B. Moore, *Optimal Filtering*, Englewood Cliffs, NJ: Prentice-Hall, 1979.
2. J. V. Candy, *Signal Processing: The Model-Based Approach*, New York: McGraw-Hill, 1986.
3. G. E. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*, San Francisco: Holden-Day, 1976.
4. R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *J. Basic Eng.*, Vol. 82, 35–43 1960.
5. A. Jazwinski, *Stochastic Processes and Filtering Theory*, New York: Academic Press, 1970.

6. T. Kailath, "The innovations approach to detection and estimation theory," *Proc. IEEE*, **58**, 680–695 1970.
7. T. Kailath, *Lectures on Kalman and Wiener Filtering Theory*, New York: Springer-Verlag, 1981.
8. A. P. Sage and J. L. Melsa, *Estimation Theory with Applications to Communications and Control*, New York: McGraw-Hill, 1971.
9. A. P. Sage and J. L. Melsa, *System Identification*, New York: Academic Press, 1971.
10. G. C. Goodwin and R. L. Payne, *Dynamic System Identification*, New York: Academic Press, 1976.
11. P. Meyer, *Introductory Probability and Statistical Applications*, Reading, MA: Addison-Wesley, 1965.
12. G. Jenkins and J. Watts, *Spectral Analysis*, San Francisco: Holden-Day, 1969.
13. R. K. Mehra and J. Peschon, "An innovations approach to fault detection and diagnosis in dynamic systems," *Automatica*, **7**, 637–640 1971.
14. A. S. Wilsky, "A survey of design methods for failure detection in dynamic systems,"*Automatica*, **12**, 601–611 1976.
15. W. C. Martin and A. R. Stubberud, "The innovations process with application to identification," in C. T. Leondes (ed.), *Control and Dynamic Systems*, New York: Academic Press, 1976.
16. F. Schweppe, *Uncertain Dynamic Systems*, Engelwood Cliffs, NJ: Prentice-Hall, 1973.
17. J. V. Candy and P. M. Candy, "SSPACK_PC: A model-based signal processing package on personal computers," *DSP Applications*, **2** (3), 33–42 1993 (see **http://www.techni-soft.net** for more details).
18. L. J. Ljung, *System Identification: Theory for the User*, Englewood Cliffs, NJ: Prentice-Hall, 1987.
19. T. Soderstrom and P. Stoica, *System Identification*, New York: Academic Press, 1989.
20. P. Maybeck, *Stochastic Models, Estimation, and Control*, New York: Academic Press, 1979.
21. M. T. Silvia and E. A. Robinson, *Deconvolution of Geophysical Time Series in the Exploration of Oil and Natural Gas*, New York: Elsevier, 1979.
22. J. Mendel, J. Kormylo, J. Lee, and F. Ashirafi, "A novel approach to seismic signal processing and modeling," *Geophysics*, **46**, 1398–1414 1981.
23. J. V. Candy and J. E. Zicker, "Deconvolution of noisy transient signals: A Kalman filtering application," *LLNL Rep.*, UCID-87432; and *Proc. CDC Conf.*, Orlando, 1982.
24. J. V. Candy and R. B. Rozsa, "Safeguards for a plutonium concentrator—An applied estimation approach," *Automatica*, **16**, 615–627 1980.
25. J. M. Mendel, *Lessons in Estimation Theory for Signal Processing, Communications and Control*, Englewood Cliffs, NJ: Prentice-Hall, 1995.
26. G. J. Bierman, *Factorization Methods of Discrete Sequential Estimation*, New York: Academic Press, 1977.
27. P. Faurre, "Stochastic realization algorithms," in *System Identification: Advances and Case Studies*, R. Mehra and C. Laineotis (eds.), New York: Academic Press, 1976.
28. D. R. Dunn, "Dynamic models, estimation and detection concepts for safeguarding $Pu(NO_3)_4$ storage tanks," *LLNL Rep.*, UCRL-79216, 1977.

## PROBLEMS

**5.1** The variance of a zero-mean sequence of $N$ data values $\{y(t)\}$ can be estimated using

$$R_{yy}(N) = \frac{1}{N}\sum_{i=1}^{N} y^2(i)$$

**(a)** Develop a recursive form for this estimator.

**(b)** Repeat part *(a)* if $\{y(t)\}$ has mean $m_y$.

**5.2** The data set $\{y(t)\} = \{2.5, -0.5, 2.2, 1.7\}$ was generated by the following Gauss-Markov model:

$$x(t) = -0.5x(t-1) + w(t-1)$$
$$y(t) = x(t) + v(t)$$

where $R_{ww} = 1$, $R_{vv} = 4$, and $x(0) \sim N(0, 1.33)$.

**(a)** Using the *MBP* algorithm calculate $\hat{x}(2|2)$.

**(b)** Assume that $\tilde{P}(t) = \cdots = \tilde{P}$ is a constant (steady-state value). Calculate $\tilde{P}$ using the prediction and correction covariance equations.

**(c)** Calculate the corresponding (steady-state) gain $\overline{K}$.

**(d)** Under the assumptions of steady-state operation of the algorithm ($P = \tilde{P}$, $K = \overline{K}$), recalculate $\hat{x}(2|2)$.

**(e)** What calculations have been eliminated using $\overline{K}(\tilde{P})$ instead of $K(t)(\tilde{P}(t|t))$?

**(f)** What are the trade-offs between the steady-state processor and a transient or dynamic filter?

**5.3** A birdwatcher is counting the number of birds migrating to and from a particular nesting area. Suppose that the number of migratory birds, $m(t)$, is modeled by a first-order *ARMA* model:

$$m(t) = -0.5m(t-1) + w(t) \quad \text{for} \quad w \sim \mathcal{N}(10, 75)$$

while the number of resident birds is static, that is,

$$r(t) = r(t-1)$$

The number of resident birds is averaged leading to the expression

$$y(t) = 0.5r(t) + m(t) + v(t) \qquad \text{for} \quad w \sim \mathcal{N}(0, 0.1)$$

**(a)** Develop the two state Gauss-Markov model with initial values $r(0) = 20$ birds, $m(0) = 100$ birds, cov $r(0) = 25$.

**(b)** Use the *MBP* algorithm to estimate the number of resident and migrating birds in the nesting area for the data set $y(t) = \{70, 80\}$; that is, what is $\hat{x}(2|2)$?

**5.4** Show that for a batch of $N$ measurements the $N_x N \times N_y N$ matrix, $R_{xe}(\underline{N})$ is block lower triangular and $R_{ee}(\underline{N})$ is block diagonal.

**5.5** Using the predictor-corrector *MBP* equations, develop the *prediction form* of the filter by rewriting the *MBP* equations only in terms of predicted estimates and covariances.

**(a)** What is the equation for the *prediction gain* $K_p(t)$ in this algorithm?

**(b)** How is $K_p$ related to $K$ in the predictor-corrector form?

**(c)** Develop the predicted form of the error covariance equation (Riccati equation).

**5.6** Suppose that $\{y(t)\}$ is a zero-mean scalar process with covariance

$$R_y(i, j) = E\{y(i)y(j)\} = \alpha^{|i-j|}, \qquad 0 < \alpha < 1$$

Find the corresponding innovations sequence $\{e(t)\}$.

**5.7** Suppose that we have a *scalar* process given by

$$x(t) = \tfrac{1}{2}x(t-1) + w(t-1), \quad w \sim N(0, R_{ww})$$
$$y(t) = cx(t) + v(t), \qquad\qquad v \sim N(0, R_{vv})$$

**(a)** What is the steady-state gain for this systems?

**(b)** What is the effect of increasing $R_{ww}$ on $K_{ss}$ for fixed $R_{vv}$?

**(c)** What is the effect of decreasing $R_{vv}$ on $K_{ss}$ for fixed $R_{ww}$?

**(d)** Suppose $R_{vv} = 0$; what is the effect of increasing $R_{ww}$ on $K_{ss}$?

**(e)** Suppose $R_{ww} = 0$; what is the effect of decreasing $R_{vv}$ on $K_{ss}$?

**5.8** In a simple radar system, a large-amplitude, narrow-width pulsed sinusoid is transmitted by an antenna. The pulse propagates through space at light speed ($c = 3 \times 10^8$ m/s) until it hits the object being tracked. The object reflects a portion of the energy. The radar antenna receives a portion of the reflected energy. By locating the time difference between the leading edge of the transmitted and received pulse $\triangle t$, the distance or range from the radar $r$ can be determined, that is,

$$r = \frac{c\triangle t}{2}$$

The pulse is transmitted periodically, that is,

$$r(t) = \frac{c\triangle(t)}{2}$$

Assume that the object is traveling at a constant velocity $\rho$ with random disturbance $w \sim \mathcal{N}(0, 100 \text{ m/s})$. Therefore the range of the object at $t + 1$ is

$$r(t+1) = r(t) + T\rho(t) \quad T = 1 \text{ m/s}$$

where $T$ is the sampling interval between received range values. The measured range is contaminated by noise

$$y(t) = r(t) + n(t) \qquad \text{for } n \sim \mathcal{N}(0, 10)$$

**(a)** Develop a model to simulate this radar system.

**(b)** Design a range estimator using the *MBP* and assuming that the initial target range is $r(0) \approx 20 \pm 2$ km.

**5.9** Suppose that we are given the *ARMAX* model

$$y(t) + a_1 y(t-1) = b_1 u(t-1) + c_1 e(t-1)$$

**(a)** Assuming $c_1$ is known, develop the *MBP* or identifier for this problem.

**(b)** Assuming $c_1$ is unknown, "extend" *MBP* or identifier to solve this problem.

**5.10** Suppose that we are given the *ARMAX* model

$$y(t) + 1.5y(t-1) - 0.7y(t-2) = u(t-1) + 0.5u(t-2) + e(t)$$

where $u$ is a pseudorandom binary sequence and $e$ is white gaussian noise with variance 0.01.

**(a)** Simulate this process for 256 sample points.

**(b)** Apply the *MBP* identifier algorithm to the data. What are the final parameter estimates? Show plots of results.

**5.11** Develop the *MBP* for deconvolution for the following Gauss-Markov model

$$x(t) = -ax(t-1) + bu(t-1) + w(t-1)$$
$$y(t) = cx(t) + v(t)$$

with $w \sim \mathcal{N}(0, R_{ww})$ if

**(a)** $u$ is assumed piecewise constant.

**(b)** $u$ is assumed piecewise linear.

**(c)** $u$ is random and exponentially correlated.

**(d)** $u$ is modeled by the Taylor series

$$u(t+T) \approx \sum_{i=0}^{N} \frac{\partial^i u(t)}{\partial t^i} \frac{T^i}{i!}$$

**5.12** Suppose that we are given the following innovations model (in steady state):

$$\hat{x}(t) = a\hat{x}(t-1) + ke(t-1)$$
$$y(t) = c\hat{x}(t) + e(t)$$

where $e$ is the zero-mean, white innovations sequence with covariance $R_{ee}$. Derive the equivalent Wiener filter.

**5.13** Construct an innovations sequence $\{e(1), e(2)\}$ from the measurement sequence

$$\begin{bmatrix} 1 \\ 4 \end{bmatrix}, \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

if

$$R_{ee} = \text{diag}\,[1, 2] \quad \text{and} \quad R_{ye}(2, 1) = \begin{bmatrix} 2 & 1 \\ 1 & 6 \end{bmatrix}$$

**5.14** Suppose that we are given a measurement device that not only acquires the current state but also the state to be delayed by one time step (multipath) such that

$$y(t) = Cx(t) + Ex(t-1) + v(t)$$

Derive the recursive form for this associated *MBP*. (*Hint*: Recall from the properties of the state transition matrix that $x(t) = \Phi(t, \tau)x(\tau)$ and $\Phi^{-1}(t, \tau) = \Phi(\tau, t)$.)

**(a)** Using the state transition matrix for discrete-systems, find the relationship between $x(t)$ and $x(t-1)$ in the Gauss-Markov model.

**(b)** Substitute this result into the measurement equation to obtain the usual form

$$y(t) = \tilde{C}x(t) + \tilde{v}(t)$$

What are the relations for $\tilde{C}$ and $\tilde{v}(t)$ in the new system?

**(c)** Derive the new statistics for $\tilde{v}(t)$ $(\mu_{\tilde{v}}, \mathbf{R}_{\tilde{v}\tilde{v}})$.

**(d)** Are $w$ and $v$ correlated? If so, use the prediction form to develop the *MBP* algorithm for this system.

**5.15** We are given the following second-order differential equation:

$$\ddot{z}(t) + \alpha^2 z(t) = 0 \qquad \text{for} \quad \alpha \quad \text{constant}$$
$$y(t) = \beta z(t)$$

**(a)** Discretize the system using *first* differences $(\dot{x}(t) \approx \frac{x(t)-x(t-1)}{dt})$ and obtain the state-space representation.

**(b)** Discretize the system using *central* differences $(\ddot{x}(t) \approx \frac{x(t)-2x(t-1)+x(t-2)}{dt})$ and obtain the state-space representation.

**(c)** Create the Gauss-Markov models for both.

**(d)** Develop the *MBP* algorithms for both.

**5.16** The covariance correction equation of the *MBP* algorithm is seldom used directly. Numerically, the covariance matrix $\tilde{P}(t|t)$ must be positive semidefinite, but in the correction equation we are subtracting a matrix from a positive semidefinite matrix and cannot guarantee that the result will remain positive semidefinite (as it should be) because of roundoff and truncation errors. A solution to this problem is to replace the standard correction equation with the stabilized *Joseph form*, that is,

$$\tilde{P}(t|t) = [I - K(t)C(t)]\tilde{P}(t|t-1)[I - K(t)C(t)]' + K(t)R_{vv}(t)K'(t)$$

**(a)** Derive the *Joseph* stabilized form.

**(b)** Demonstrate that it is equivalent to the standard correction equation.

**5.17** Derive the *continuous-time MBP* of Chapter 2 by starting with the discrete equations of Table 5.1 and using the following sampled-data approximations:

$$\begin{aligned} A &= e^{A_c \triangle t} \approx I + A_c \triangle t \\ B &= B_c \triangle t \\ W &= W_c \triangle t \\ R_{ww} &= R_{w_c w_c} \triangle t \end{aligned}$$

# 6

# NONLINEAR STATE-SPACE MODEL-BASED PROCESSORS

## 6.1 LINEARIZED *MBP* (KALMAN FILTER)

In this section we develop an approximate solution to the nonlinear processing problem involving the linearization of the nonlinear process about a "known" reference trajectory followed by the development of a *MBP* based on the underlying linearized model. In practice, many processes are nonlinear rather than linear. Coupling the nonlinearities with noisy data makes the signal processing problem a challenging one. In this section we limit our discussion to *discrete* nonlinear systems. Continuous solutions to this problem are developed in [1]–[7].

Suppose that we model a *process* by a set of nonlinear stochastic vector difference equations in state-space form as

$$x(t) = a[x(t-1)] + b[u(t-1)] + w(t-1) \tag{6.1}$$

with the corresponding *measurement* model

$$y(t) = c[x(t)] + v(t) \tag{6.2}$$

where $a[\cdot]$, $b[\cdot]$, $c[\cdot]$ are nonlinear vector functions of $x$, $u$, with $x, a, b, w \in R^{N_x \times 1}$, $y, c, v \in R^{N_y \times 1}$, and $w \sim \mathcal{N}(0, R_{ww}(t))$, $v \sim \mathcal{N}(0, R_{vv}(t))$.

*Model-Based Signal Processing*, by James V. Candy

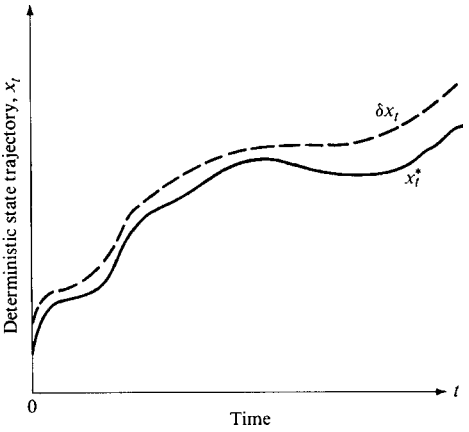



**Figure 6.1.** Linearization of a deterministic system using the reference trajectory defined by $(x^*(t), u^*(t))$.

Ignoring the additive noise sources, we "linearize" the process and measurement models about a known deterministic *reference trajectory* defined by $\left[x^*(t), u^*(t)\right]$[1] (see Figure 6.1), that is,

$$x^*(t) = a[x^*(t-1)] + b[u^*(t-1)] \tag{6.3}$$

Deviations or perturbations from this trajectory are defined by

$$\delta x(t) := x(t) - x^*(t)$$
$$\delta u(t) := u(t) - u^*(t)$$

Substituting the previous equations into these expressions, we obtain the perturbation trajectory as

$$\delta x(t) = a[x(t-1)] - a[x^*(t-1)] + b[u(t-1)] - b[u^*(t-1)] + w(t-1) \tag{6.4}$$

The nonlinear vector functions $a[\cdot]$ and $b[\cdot]$ can be expanded into a first order Taylor series about the reference trajectory $\left[x^*(t), u^*(t)\right]$ as[2]

$$a[x(t-1)] = a[x^*(t-1)] + \frac{da[x^*(t-1)]}{dx^*(t-1)}\delta x(t-1) + \text{H.O.T.}$$

$$b[u(t-1)] = b[u^*(t-1)] + \frac{db[u^*(t-1)]}{du^*(t-1)}\delta u(t-1) + \text{H.O.T.} \tag{6.5}$$

[1]In practice, the reference trajectory is obtained either by developing a mathematical model of the process or by simulating about some reasonable operating conditions to generate the trajectory using the state-space model.

[2]We use the shorthand notation $\frac{d(\cdot)}{d\theta^*}$ to mean $\frac{d(\cdot)}{d\theta}|_{\theta=\theta^*}$.

We define the first order jacobian matrices as

$$A[x^*(t-1)] := \frac{da[x^*(t-1)]}{dx^*(t-1)}, \quad \text{and} \quad B[u^*(t-1)] := \frac{db[u^*(t-1)]}{du^*(t-1)} \tag{6.6}$$

Neglecting the *higher order terms* (H.O.T.) in Eq. (6.5) and substituting into Eq. (6.4), we obtain the linearized process model as

$$\delta x(t) = A[x^*(t-1)]\delta x(t-1) + B[u^*(t-1)]\delta u(t-1) + w(t-1) \tag{6.7}$$

Similarly the measurement system can be linearized by using the reference measurement

$$y^*(t) = c[x^*(t)] \tag{6.8}$$

Now, applying the Taylor series expansion to the nonlinear measurement model yields

$$c[x(t)] = c[x^*(t)] + \frac{dc[x^*(t)]}{dx^*(t)}\delta x(t) + \text{H.O.T.} \tag{6.9}$$

The corresponding measurement perturbation is defined by

$$\delta y(t) := y(t) - y^*(t) = c[x(t)] - c[x^*(t)] + v(t) \tag{6.10}$$

Substituting the first-order approximation for $c[x(t)]$ leads to the linearized measurement perturbation as

$$\delta y(t) = C[x^*(t)]\delta x(t) + v(t) \tag{6.11}$$

where $C[x^*(t)]$ is defined as the measurement jacobian as before.

Summarizing, we have linearized a deterministic nonlinear model using a first-order Taylor series expansion for the functions, $a$, $b$, and $c$ and have developed a *linearized Gauss-Markov perturbation model* valid for small deviations given by

$$\begin{aligned} \delta x(t) &= A[x^*(t-1)]\delta x(t-1) + B[u^*(t-1)]\delta u(t-1) + w(t-1) \\ \delta y(t) &= C[x^*(t)]\delta x(t) + v(t) \end{aligned} \tag{6.12}$$

with $A$, $B$, and $C$ the corresponding jacobian matrices and $w$, $v$ zero-mean, and gaussian.

We can also use linearization techniques to approximate the statistics of Eqs. (6.1) and (6.2). If we use the first-order Taylor series expansion and expand about the mean, $m_x(t)$, rather than $x^*(t)$, then taking expected values

$$m_x(t) = E\{a[x(t-1)]\} + E\{b[u(t-1)]\} + E\{w(t-1)\} \tag{6.13}$$

gives

$$m_x(t) = a[m_x(t-1)] + b[u(t-1)] \tag{6.14}$$

which follows by linearizing $a[\cdot]$ about $m_x$ and taking the expected value to obtain

$$\begin{aligned} E\{a[x(t-1)]\} &= E\{a[m_x(t-1)] + \frac{da[m_x(t-1)]}{dm_x(t-1)}[x(t-1) - m_x(t-1)]\} \\ &= a[m_x(t-1)] + \frac{da[m_x(t-1)]}{dm_x(t-1)}[E\{x(t-1)\} - m_x(t-1)] \\ &= a[m_x(t-1)] \end{aligned}$$

The variance equations $P(t) := \text{cov}(x(t))$ can also be developed in similar manner (see [2] for details) to give

$$P(t) = A[m_x(t-1)]P(t-1)A'[m_x(t-1)] + R_{ww}(t-1) \tag{6.15}$$

Using the same approach, we arrive at the accompanying measurement statistics

$$m_y(t) = c[m_x(t)] \quad \text{and} \quad R_{yy}(t) = C[m_x(t)]P(t)C'[m_x(t)] + R_{vv}(t) \tag{6.16}$$

We summarize these results in an "approximate" Gauss-Markov model of Table 6.1.

Suppose that we utilize the perturbation model of Eq. (6.12) to construct a "linearized" *MBP* using the $(A, B, C)$ jacobians linearized about the reference trajectory $[x^*, u^*]$. The processor follows directly, since each of the jacobians are *deterministic*, but updated at each time step to become time-varying matrices. Substituting the jacobians and $\delta\hat{x}(t|t-1)$ for the matrices and $\hat{x}(t|t-1)$ in Table 5.1, we obtain a form of linearized filter for $\delta\hat{x}(t|t-1)$, the estimated state perturbation. Therefore the linearized filter gives the estimate $\delta\hat{x}(t|t-1)$ of $\delta x(t)$, that is

$$\hat{x}(t|t-1) = \delta\hat{x}(t|t-1) + x^*(t) \tag{6.17}$$

This algorithm is not implemented in this manner because we are interested in the state estimate $\hat{x}(t|t-1)$ not its deviation $\delta\hat{x}(t|t-1)$. We can rewrite this algorithm in terms of $\hat{x}(t|t-1)$, if we substitute the prediction equation of the linearized filter for $\delta\hat{x}(t|t-1)$ above. That is, substitute

$$\delta\hat{x}(t|t-1) = A[x^*(t-1)]\delta\hat{x}(t-1|t-1) + B[u^*(t-1)]\delta u(t-1)$$

and

$$x^*(t) = a[x^*(t-1)] + b[u^*(t-1)]$$

into Eq. (6.17) to obtain

$$\begin{aligned} \hat{x}(t|t-1) = a[x^*(t-1)] &+ A[x^*(t-1)]\left[\hat{x}(t-1|t-1) - x^*(t-1)\right] \\ &+ b[u^*(t-1)] + B[u^*(t-1)]\left[u(t-1) - u^*(t-1)\right] \end{aligned} \tag{6.18}$$

**Table 6.1. Approximate Nonlinear Gauss-Markov Model**

State propagation

$$x(t) = a[x(t-1)] + b[u(t-1)] + w(t-1)$$

State mean propagation

$$m_x(t) = a[m_x(t-1)] + b[u(t-1)]$$

State covariance propagation

$$P(t) = A[m_x(t-1)]P(t-1)A'[m_x(t-1)] + R_{ww}(t-1)$$

Measurement propagation

$$y(t) = c[x(t)] + v(t)$$

Measurement mean propagation

$$m_y(t) = c[m_x(t)]$$

Measurement covariance propagation

$$R_{yy}(t) = C[m_x(t)]P(t)C'[m_x(t)] + R_{vv}(t)$$

Initial conditions

$$x(0) \quad \text{and} \quad P(0)$$

Jacobians

$$A[x^*(t-1)] := \frac{da[x(t-1)]}{dx(t-1)}\Big|_{x=x^*(t-1)}, \quad C[x^*(t)] := \frac{dc[x(t)]}{dx(t)}\Big|_{x=x^*(t)}$$

The corresponding *perturbed* innovation can also be found directly

$$\begin{aligned}\delta e(t) &= \delta y(t) - \delta\hat{y}(t|t-1) = (y(t) - y^*(t)) - (\hat{y}(t|t-1) - y^*(t)) \\ &= y(t) - \hat{y}(t|t-1) = e(t)\end{aligned} \tag{6.19}$$

Using the linear *MBP* with deterministic jacobian matrices results in

$$\delta\hat{y}(t|t-1) = C[x^*(t)]\delta\hat{x}(t|t-1) \tag{6.20}$$

and therefore using this relation and Eq. (6.8) for the reference measurement, we have

$$\hat{y}(t|t-1) = y^*(t) + C[x^*(t)]\delta\hat{x}(t|t-1) = c[x^*(t)] + C[x^*(t)]\delta\hat{x}(t|t-1) \tag{6.21}$$

It follows that the innovation is

$$e(t) = y(t) - c[x^*(t)] - C[x^*(t)][\hat{x}(t|t-1) - x^*(t)] \tag{6.22}$$

The corrected estimate is easily found by substituting Eq. (6.17) to obtain

$$\begin{aligned} \delta\hat{x}(t|t) &= \delta\hat{x}(t|t-1) + K(t)e(t) \\ \left[\hat{x}(t|t) - x^*(t)\right] &= \left[\hat{x}(t|t-1) - x^*(t)\right] + K(t)e(t) \end{aligned} \tag{6.23}$$

which yields the identical correction equation of Table 5.1. Since the state perturbation estimation error is identical to the state estimation error, the corresponding error covariance is given by $\delta\tilde{P}(t|\cdot) = \tilde{P}(t|\cdot)$. Therefore

$$\begin{aligned} \delta\tilde{x}(t|\cdot) &= \delta x(t) - \delta\hat{x}(t|\cdot) = [x(t) - x^*(t)] - [\hat{x}(t|\cdot) - x^*(t)] \\ &= x(t) - \hat{x}(t|\cdot) = \tilde{x}(t|\cdot) \end{aligned} \tag{6.24}$$

The gain is just a function of the measurement linearization, $C[x^*(t)]$ completing the algorithm. We summarize the discrete linearized *MBP* (Kalman filter) in Table 6.2

In a more formal framework, the *LZ-MBP* can be developed under (approximate) gaussian assumptions using the *Bayesian approach* as before in the linear case. We briefly outline the derivation by carefully following the steps in Section 5.4.

The *a posteriori probability* is given by

$$\Pr(x(t)|Y(t)) = \frac{\Pr(y(t)|x(t)) \times \Pr(x(t)|Y(t-1))}{\Pr(y(t)|Y(t-1))} \tag{6.25}$$

Under the Gauss-Markov model assumptions, we know that each of the conditional expectations can be expressed in terms of the conditional gaussian distributions as follows:

1. $\Pr(y(t)|x(t)) \quad : \mathcal{N}(c[x(t)], R_{vv}(t))$
2. $\Pr(x(t)|Y(t-1)) \; : \mathcal{N}(\hat{x}(t|t-1), \tilde{P}(t|t-1))$
3. $\Pr(y(t)|Y(t-1)) \; : \mathcal{N}(\hat{y}(t|t-1), R_{ee}(t))$

Using the nonlinear models developed earlier, substituting the gaussian probabilities and taking logarithms, we obtain the logarithmic a posteriori probability as

$$\begin{aligned} \ln \Pr(x(t)|Y(t)) = \ln \kappa &- \tfrac{1}{2}v'(t)R_{vv}^{-1}(t)v(t) - \tfrac{1}{2}\tilde{x}'(t|t-1)\tilde{P}^{-1}(t|t-1)\tilde{x}(t|t-1) \\ &+ \tfrac{1}{2}e'(t)R_{ee}^{-1}(t)e(t) \end{aligned} \tag{6.26}$$

The *MAP* estimate is then obtained by differentiating this equation, setting the result to zero, and solving for $x(t)$, that is,

$$\nabla_x \ln \Pr(x(t)|Y(t))|_{x=\hat{X}_{map}} = \mathbf{0} \tag{6.27}$$

**Table 6.2. Discrete Linearized *MBP* (Kalman Filter) Algorithm**

Prediction

$$\hat{x}(t|t-1) = a[x^*(t-1)] + A[x^*(t-1)] \left[\hat{x}(t-1|t-1) - x^*(t-1)\right] + b[u^*(t-1)] + B[u^*(t-1)]\left[u(t-1) - u^*(t-1)\right] \quad \text{(State prediction)}$$

$$\tilde{P}(t|t-1) = A[x^*(t-1)]\tilde{P}(t-1|t-1)A'[x^*(t-1)] + R_{ww}(t-1) \quad \text{(Covariance prediction)}$$

Innovation

$$e(t) = y(t) - c[x^*(t)] - C[x^*(t)]\,(\hat{x}(t|t-1) - x^*(t)) \quad \text{(Innovation)}$$

$$R_{ee}(t) = C[x^*(t)]\tilde{P}(t|t-1)C'[x^*(t)] + R_{vv}(t) \quad \text{(Innovation covariance)}$$

Gain

$$K(t) = \tilde{P}(t|t-1)C'[x^*(t)]R_{ee}^{-1}(t) \quad \text{(Gain or weight)}$$

Correction

$$\hat{x}(t|t) = \hat{x}(t|t-1) + K(t)e(t) \quad \text{(State correction)}$$

$$\tilde{P}(t|t) = (I - K(t)C[x^*(t)])\,\tilde{P}(t|t-1) \quad \text{(Covariance correction)}$$

Initial conditions

$$\hat{x}(0|0) \quad \tilde{P}(0|0)$$

Jacobians

$$A[x^*(t-1)] \equiv \frac{da[x(t-1)]}{dx(t-1)}\Big|_{x=x^*(t-1)}, \quad B[u^*(t-1)] \equiv \frac{db[u(t-1)]}{du(t-1)}\Big|_{u=u^*(t-1)},$$

$$C[x^*(t)] \equiv \frac{dc[x(t)]}{dx(t)}\Big|_{x=x^*(t)}$$

Before we derive the *MAP* estimate, we first linearize about a reference trajectory, $x^* \longrightarrow x$ with the nonlinear measurement model approximated by a first-order Taylor series as in Eq. (6.9),

$$c[x(t)] \approx c[x^*(t)] + \frac{dc[x^*(t)]}{dx^*(t)}\delta x(t) = c[x^*(t)] + C[x^*(t)]\left(x(t) - x^*(t)\right)$$

Substituting this result into Eq. (6.26), we obtain

$$\begin{aligned}\ln \Pr(x(t)|Y(t)) = \ln \kappa &- \tfrac{1}{2}\left(y(t) - c[x^*(t)] - C[x^*(t)](x(t) - x^*(t))\right)' R_{vv}^{-1}(t) \\ &\times \left(y(t) - c[x^*(t)] - C[x^*(t)](x(t) - x^*(t))\right) - \tfrac{1}{2}\tilde{x}'(t|t-1) \\ &\times \tilde{P}^{-1}(t|t-1)\tilde{x}(t|t-1) + \tfrac{1}{2}e'(t)R_{ee}^{-1}(t)e(t) \end{aligned} \quad (6.28)$$

Applying the *chain rule* and the gradient operator (see Chapter 3), we obtain the following expression. Note that the last term of Eq. (6.28) is not a function of $x(t)$, but just the data, so it is null.

$$\begin{aligned}\nabla_x \ln \Pr(x(t)|Y(t)) = &-C'[x^*(t)]R_{vv}^{-1}(t)(y(t) - c[x^*(t)] \\ &- C[x^*(t)](x(t) - x^*(t))) \\ &- \tilde{P}^{-1}(t|t-1)\,[x(t) - \hat{x}(t|t-1)] = 0 \end{aligned} \tag{6.29}$$

Multiplying through and grouping like terms in $x(t)$ gives

$$\begin{aligned}&C'[x^*(t)]R_{vv}^{-1}(t)\left(y(t) - c[x^*(t)] + C[x^*(t)]x^*(t)\right) \\ &-\left[\tilde{P}^{-1}(t|t-1) + C'[x^*(t)]R_{vv}^{-1}(t)C[x^*(t)]\right]x(t) \\ &+ \tilde{P}^{-1}(t|t-1)\hat{x}(t|t-1) = 0 \end{aligned} \tag{6.30}$$

Solving for $x(t) = \hat{X}_{\text{map}}(t)$ gives

$$\begin{aligned}\hat{X}_{\text{map}}(t) = &\left[\tilde{P}^{-1}(t|t-1) + C'[x^*(t)]R_{vv}^{-1}(t)C[x^*(t)]\right]^{-1} \\ &\times\left[\tilde{P}^{-1}(t|t-1)\hat{x}(t|t-1) + C'[x^*(t)]R_{vv}^{-1}(t)\right. \\ &\left.\times\left(y(t) - c[x^*(t)] + C[x^*(t)]x^*(t)\right)\right] \end{aligned} \tag{6.31}$$

Using the results of Eqs. (5.52) through (5.57) with $C(t) \longrightarrow C[x^*(t)]$, we rewrite the first term in Eq. (6.31) as

$$\begin{aligned}\left[\tilde{P}^{-1}(t|t-1) + C'[x^*(t)]R_{vv}^{-1}(t)C[x^*(t)]\right]^{-1} &= (I - \tilde{P}(t|t-1)C'[x^*(t)] \\ R_{ee}^{-1}(t)C[x^*(t)])\tilde{P}^{-1}(t|t-1) &= \left(I - K(t)C[x^*(t)]\right) \\ &\times\tilde{P}(t|t-1) \equiv \tilde{P}(t|t) \end{aligned} \tag{6.32}$$

where $K$ is the Kalman gain and $R_{ee}$ is the innovations covariance of the *LZ-MBP*, with this expression precisely the *corrected error covariance*, $\tilde{P}(t|t)$, as in Table 5.1. Solving this equation for the inverse of the predicted error covariance gives

$$\tilde{P}^{-1}(t|t-1) = \tilde{P}^{-1}(t|t) - C'[x^*(t)]R_{vv}^{-1}(t)C[x^*(t)] \tag{6.33}$$

Substituting into Eq. (6.31) using the results of Eq. (6.32) yields

$$\begin{aligned}\hat{X}_{\text{map}}(t) = \tilde{P}(t|t)[&(\tilde{P}^{-1}(t|t)\hat{x}(t|t-1) - C'[x^*(t)]R_{vv}^{-1}(t)C[x^*(t)]\hat{x}(t|t-1)) \\ &+ C'[x^*(t)]R_{vv}^{-1}(t)(y(t) - c[x^*(t)] + C[x^*(t)]x^*(t))] \end{aligned} \tag{6.34}$$

Multiplying through by the corrected error covariance and recognizing the expression for the Kalman gain gives

$$\hat{X}_{\text{map}}(t) = \hat{x}(t|t-1) - K(t)C[x^*(t)]\hat{x}(t|t-1) + K(t)C[x^*(t)]x^*(t) + K(t)\left(y(t) - c[x^*(t)]\right) \tag{6.35}$$

which leads to the final expression for the linearized *MAP* estimate as

$$\hat{X}_{\text{map}}(t) = \hat{x}(t|t) = \hat{x}(t|t-1) + K(t)(y(t) - c[x^*(t)] - C[x^*(t)]\left(\hat{x}(t|t-1) - x^*(t)\right) \tag{6.36}$$

Compare this result to Table 5.1. The error covariances and predicted estimates follow directly guided by the linear case. This completes the derivation. Consider the following discrete nonlinear example of the nonlinear system problem given in Jazwinski [1].

***Example 6.1*** Consider the discrete nonlinear process given by

$$x(t) = (1 - 0.05\triangle T)x(t-1) + 0.04x^2(t-1) + w(t-1)$$

with corresponding measurement model

$$y(t) = x^2(t) + x^3(t) + v(t)$$

where $v(t) \sim \mathcal{N}(0, 0.09)$, $x(0) = 2.3$, $P(0) = 0.01$, $\triangle T = 0.01$ s and $R_{ww} = 0$. The simulated measurement using *SSPACK_PC* [8] is shown in Figure 6.2*c*. The *LZ-MBP* is designed from the following jacobians:

$$A[x(t-1)] = 1 - 0.05\triangle T + 0.08\triangle T x(t-1) \quad \text{and} \quad C[x(t)] = 2x(t) + 3x^2(t)$$

Observing the mean state, we develop a reference trajectory by fitting a line to the simulated state, which is given by

$$x^*(t) = 0.067\ t + 2.0, \qquad 0 \leq\ t\ \leq\ 1.5$$
$$u^*(t) = u(t) = 0.0 \qquad \forall t$$

The *LZ-MBP* algorithm is then given by

$$\hat{x}(t|t-1) = (1 - 0.05\triangle T)x^*(t-1) + \left(1 - 0.05\triangle T + 0.08\triangle T x^*(t-1)\right)[\hat{x}(t-1|t-1) - x^*(t-1)]$$
$$\tilde{P}(t|t-1) = [1 - 0.05\triangle T + 0.08\triangle T x^*(t-1)]^2 \tilde{P}(t-1|t-1)$$
$$e(t) = y(t) - (x^{*2}(t) - x^{*3}(t)) - (2x^*(t) + 3x^{*2}(t))[\hat{x}(t|t-1) - x^*(t)]$$

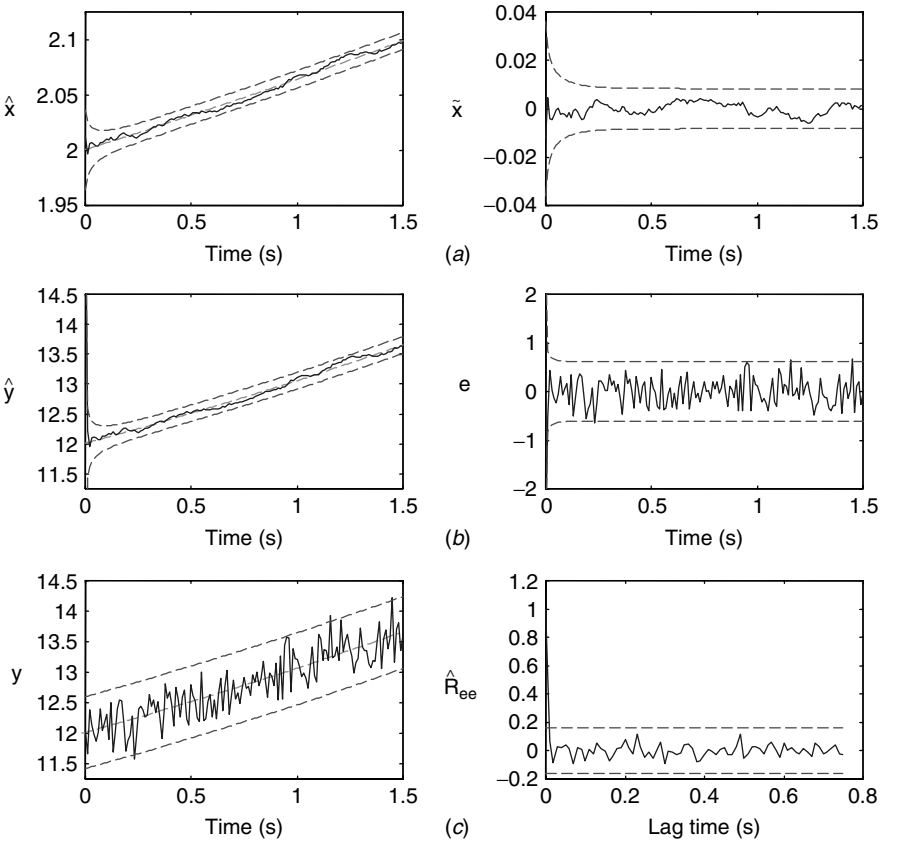



**Figure 6.2.** Linearized MBP simulation. *(a)* Estimated state (0% out) and error (0% out). *(b)* Filtered measurement (1% out) and error (innovation) (2.6% out). *(c)* Simulated measurement and zero-mean/whiteness test ($3.9 \times 10^{-2} < 10.1 \times 10^{-2}$ and 0% out).

$$R_{ee}(t) = [2\hat{x}(t|t-1) + 3\hat{x}^2(t|t-1)]^2 \tilde{P}(t|t-1) + 0.09$$

$$K(t) = \frac{\tilde{P}(t|t-1)[2x^*(t) + 3x^{*2}(t)]}{R_{ee}(t)}$$

$$\hat{x}(t|t) = \hat{x}(t|t-1) + K(t)e(t)$$

$$\tilde{P}(t|t) = (1 - K(t)[2x^*(t) + 3x^{*2}(t)])\tilde{P}(t|t-1)$$

$$\hat{x}(0|0) = 2.3 \quad \text{and} \quad \tilde{P}(0|0) = 0.01$$

A *LZ-MBP* run is depicted in Figure 6.2. Here we see that the state estimate begins tracking the true state after the initial transient. The estimation error is good (0% lie outside confidence limits), indicating that the filter is performing properly for this realization. The filtered measurement and innovations are shown

in Figure 6.2*b* with their corresponding predicted confidence limits. Both estimates lie well within these bounds. The innovations are zero mean ($3.9 \times 10^{-2} < 10.1 \times 10^{-2}$) and white (0% lie outside limits) as shown in Figure 6.2*c* indicating proper tuning. This completes the nonlinear filtering example. We summarize the results as follows:

| | |
|---|---|
| Criterion: | $J = \text{trace } \tilde{P}(t\|t)$ |
| Models: | |
| Signal: | $x(t) = (1 - 0.05\triangle T)x(t-1) + 0.04x^2(t-1) + w(t-1)$ |
| Measurement: | $y(t) = x^2(t) + x^3(t) + v(t)$ |
| Noise: | $w \sim \mathcal{N}(0, 0)$ and $v \sim \mathcal{N}(0, 0.09)$ |
| Initial conditions: | $\hat{x}(0\|0) = 2.3$ and $\tilde{P}(0\|0) = 0.01$ |
| Algorithm: | $\hat{x}(t\|t) = \hat{x}(t\|t-1) + K(t)e(t)$ |
| Quality: | $\tilde{P}(t\|t) = \left(1 - K(t)[2x^*(t) + 3x^{*2}(t)]\right) \tilde{P}(t\|t-1)$ |

## 6.2 EXTENDED *MBP* (EXTENDED KALMAN FILTER)

In this section we develop the extended model-based processor (*XMBP*) or equivalently the extended Kalman filter (*EKF*). The *XMBP* is ad hoc by nature but has become one of the workhorses of (approximate) nonlinear filtering [1], [2], [3], [6], [7], [9]. It has found applicability in a wide variety of applications such as tracking [10], navigation [1], [5], chemical processing [12], ocean acoustics [13], and seismology [14] (see [15] for a detailed list). The *XMBP* evolves directly from the linearized processor of the previous section in which the reference state, $x^*(t)$, used in the linearization process is replaced with the most recently available state estimate, $\hat{x}(t|t)$—this is the step that deems the processor ad hoc. However, we must realize that the jacobians used in the linearization process are deterministic (but time-varying) when a reference or perturbation trajectory is used, whereas the current state estimate is used as an approximation to the conditional mean and so is random, making these jacobians and subsequent equations random. Therefore, although popularly ignored, most *XMBP* designs should be based on ensemble operations to obtain reasonable estimates of the underlying statistics. With this in mind, we develop the processor directly from the *LZ-MBP*.

Thus, if instead of using the reference trajectory, we choose to linearize about each new state estimate as soon as it becomes available, then the *XMBP* algorithm results. The reason for choosing the estimate to linearize about is to use a better reference trajectory as soon as one is available. As a consequence large initial estimation errors do not propagate; therefore linearity assumptions are less likely to be violated.

Thus, if we choose to use the current estimate $\hat{x}(t|\alpha)$, where $\alpha$ is $t-1$ or $t$, to linearize about instead of the reference trajectory $x^*(t)$, then the *XMBP* evolves. That is, let

$$x^*(t) \equiv \hat{x}(t|\alpha) \qquad \text{for } t-1 \leq \alpha \leq t \tag{6.37}$$

Then, for instance, when $\alpha = t-1$, the predicted perturbation is

$$\delta\hat{x}(t|t-1) = \hat{x}(t|t-1) - x^*(t)\,\big|_{x^*=\hat{x}(t|t-1)} = 0 \tag{6.38}$$

It follows immediately that when $x^*(t) = \hat{x}(t|t)$, then $\delta\hat{x}(t|t) = 0$ as well.

Substituting the current estimate either prediction or correction into the *LZ-MBP* algorithm, it is easy to see that each of the difference terms $[\hat{x} - x^*]$ is null, resulting in the *XMBP* algorithm. That is, examining the *prediction* phase of the linearized algorithm, substituting the current available corrected estimate, $\hat{x}(t-1|t-1)$, for the reference, and using the fact that $(u^*(t) = u(t)\ \forall t)$, we have

$$\begin{aligned}\hat{x}(t|t-1) = a[\hat{x}(t-1|t-1)] + A[\hat{x}(t-1|t-1)][\hat{x}(t-1|t-1)\\ - \hat{x}(t-1|t-1)] + b[u(t-1)] + B[u(t-1)]\,[u(t-1) - u(t-1)]\end{aligned}$$

giving the prediction expression

$$\hat{x}(t|t-1) = a[\hat{x}(t-1|t-1)] + b[u(t-1)] \tag{6.39}$$

Now with the predicted estimate available, substituting it for the reference in Eq. (6.22) gives the innovation sequence as

$$\begin{aligned}e(t) &= y(t) - c[\hat{x}(t|t-1)] - C[\hat{x}(t|t-1)]\,[\hat{x}(t|t-1) - \hat{x}(t|t-1)]\\ &= y(t) - c[\hat{x}(t|t-1)]\end{aligned} \tag{6.40}$$

where we have the new predicted or filtered measurement expression

$$\hat{y}(t|t-1) := c[\hat{x}(t|t-1)] \tag{6.41}$$

The corrected state estimate is easily obtained by substituting the predicted estimate for the reference $(\hat{x}(t|t-1) \to x^*(t))$ in Eq. (6.23),

$$\begin{aligned}\delta\hat{x}(t|t) &= \delta\hat{x}(t|t-1) + K(t)e(t)\\ [\hat{x}(t|t) - \hat{x}(t|t-1)] &= [\hat{x}(t|t-1) - \hat{x}(t|t-1)] + K(t)e(t)\\ \hat{x}(t|t) &= \hat{x}(t|t-1) + K(t)e(t)\end{aligned} \tag{6.42}$$

The covariance and gain equations are identical to those in Table 6.2, but with the jacobian matrices $A$, $B$, and $C$ linearized about the predicted state estimate, $\hat{x}(t|t-1)$. Thus we obtain the discrete *XMBP* or equivalent *EKF* algorithm summarized in Table 6.3. Note that the covariance matrices, $\tilde{P}$, and the gain, $K$, are now functions of the current state estimate, which is the *approximate* conditional mean estimate and therefore a single realization of a stochastic process. Thus ensemble (Monte Carlo) techniques should be used to evaluate estimator performance, that is, for new initial conditions selected by a gaussian random number generator (either $\hat{x}(0|0)$ or $\tilde{P}(0|0)$) the algorithm is executed generating a set of estimates

**Table 6.3. Discrete Extended *MBP* (Kalman Filter) Algorithm**

Prediction

$$\hat{x}(t|t-1) = a[\hat{x}(t-1|t-1)] + b[u(t-1] \qquad \text{(State prediction)}$$

$$\tilde{P}(t|t-1) = A[\hat{x}(t|t-1)]\tilde{P}(t-1|t-1)A'[\hat{x}(t|t-1)] + R_{ww}(t-1) \qquad \text{(Covariance prediction)}$$

Innovation

$$e(t) = y(t) - c[\hat{x}(t|t-1)] \qquad \text{(Innovation)}$$

$$R_{ee}(t) = C[\hat{x}(t|t-1)]\tilde{P}(t|t-1)C'[\hat{x}(t|t-1)] + R_{vv}(t) \qquad \text{(Innovation covariance)}$$

Gain

$$K(t) = \tilde{P}(t|t-1)C'[\hat{x}(t|t-1)]R_{ee}^{-1}(t) \qquad \text{(Gain or weight)}$$

Correction

$$\hat{x}(t|t) = \hat{x}(t|t-1) + K(t)e(t) \qquad \text{(State correction)}$$

$$\tilde{P}(t|t) = [I - K(t)C[\hat{x}(t|t-1)]]\,\tilde{P}(t|t-1) \qquad \text{(Covariance correction)}$$

Initial conditions

$$\hat{x}(0|0), \quad \tilde{P}(0|0)$$

Jacobians

$$A[x(t|t-1)] \equiv \frac{da[x(t-1)]}{d\hat{x}(t-1)}\Big|_{x=\hat{x}(t|t-1)}, \quad C[\hat{x}(t|t-1)] \equiv \frac{dc[x(t)]}{dx(t)}\Big|_{x=\hat{x}(t|t-1)}$$

that should be averaged over the entire ensemble to obtain an "expected" state, and so forth. Note also in practice that this algorithm is usually implemented using sequential processing and *UD* factorization techniques (see [16] and Appendix B).

The *XMBP* can be developed under (approximate) gaussian assumptions using the linearized *Bayesian approach* by merely substituting the most current estimate for the reference in the *LZ-MBP* algorithm derivation. We demonstrate this by using the approximate *MAP* estimator for the linearized problem (see the previous section for the derivation) and show how the *XMBP* naturally evolves.

Recall from Eq. (6.36) that the approximate (linearized) *MAP* estimator evolved as

$$\hat{X}_{\text{map}}(t) = \hat{x}(t|t-1) + K(t)(y(t) - c[x^*(t)] - C[x^*(t)]\left(\hat{x}(t|t-1) - x^*(t)\right) \tag{6.43}$$

If we allow the reference trajectory to be the predicted state estimate and substitute into this relation, that is, $x^*(t) \longrightarrow \hat{x}(t|t-1)$, then we obtain

$$\hat{X}_{\text{map}}(t) = \hat{x}(t|t) = \hat{x}(t|t-1) + K(t)(y(t) - c[\hat{x}(t|t-1)] - C[\hat{x}(t|t-1)]\,(\hat{x}(t|t-1) - \hat{x}(t|t-1)) \tag{6.44}$$

Thus the last term is null, and we have the desired result

$$\hat{X}_{\text{map}}(t) = \hat{x}(t|t) = \hat{x}(t|t-1) + K(t)\,(y(t) - c[\hat{x}(t|t-1)])$$
$$= \hat{x}(t|t-1) + K(t)e(t) \tag{6.45}$$

The entire *XMBP* algorithm evolves after making these substitutions as before in the *LZ-MBP* relations. This completes the derivation and demonstrates, at least heuristically, that the *XMBP* has a relationship to the approximate *MAP* estimator as long as the linearization is reasonable—owing to its popular application and success. Consider the following discrete nonlinear example of the previous section.

***Example 6.2*** Consider the discrete nonlinear process and measurement system described in the previous example. The simulated measurement using *SSPACK_PC* [8] is shown in Figure 6.3*c* . The *XMBP* is designed from the following jacobians:

$$A[x(t-1)] = 1 - 0.05\triangle T + 0.08\triangle T x(t-1) \quad \text{and} \quad C[x(t)] = 2x(t) + 3x^2(t)$$

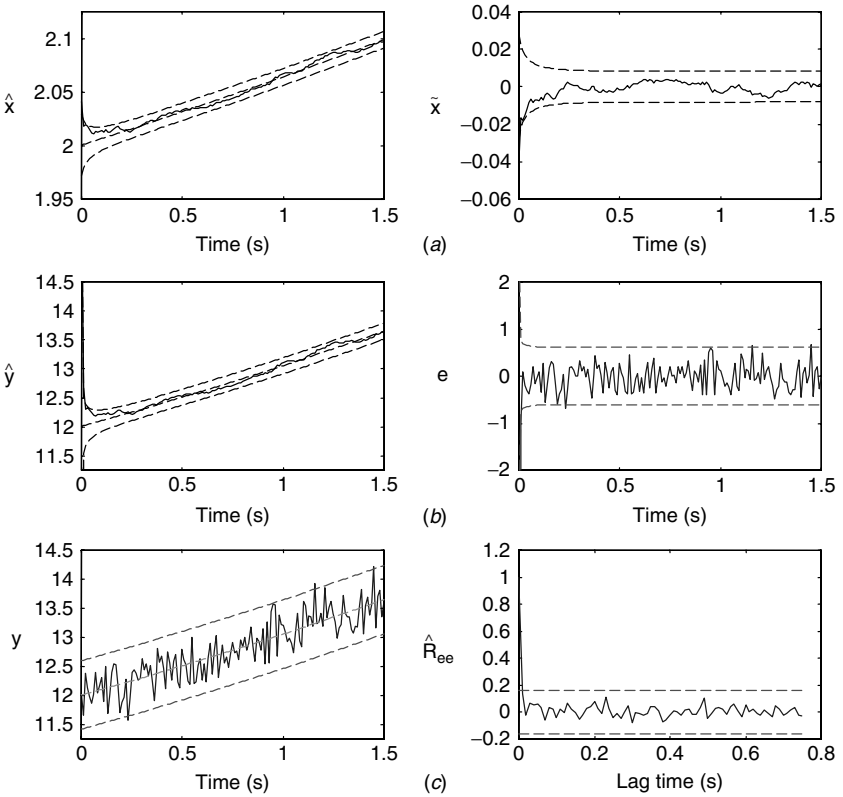



**Figure 6.3.** Extended MBP simulation. *(a)* Estimated state (1% out) and error (1% out). *(b)* Filtered measurement (1.3% out) and error (innovation) (3.3% out). *(c)* Simulated measurement and zero-mean/whiteness test ($6.3 \times 10^{-2} < 11.8 \times 10^{-2}$ and 0% out).

The *XMBP* algorithm is then given by

$$\begin{aligned}
\hat{x}(t|t-1) &= (1-0.05\triangle T)\hat{x}(t-1|t-1) + 0.04\hat{x}^2(t-1|t-1) \\
\tilde{P}(t|t-1) &= [1-0.05\triangle T + 0.08\triangle T x(t-1)]^2 \tilde{P}(t-1|t-1) \\
e(t) &= y(t) - \hat{x}^2(t|t-1) - \hat{x}^3(t|t-1) \\
R_{ee}(t) &= [2\hat{x}(t|t-1) + 3\hat{x}^2(t|t-1)]^2 \tilde{P}(t|t-1) + 0.09 \\
K(t) &= \left(\tilde{P}(t|t-1)[2\hat{x}(t|t-1) + 3\hat{x}^2(t|t-1)]\right) R_{ee}^{-1}(t) \\
\hat{x}(t|t) &= \hat{x}(t|t-1) + K(t)e(t) \\
\tilde{P}(t|t) &= \left(1 - K(t)[2\hat{x}(t|t-1) + 3\hat{x}^2(t|t-1)]\right) \tilde{P}(t|t-1) \\
\hat{x}(0|0) &= 2.3 \quad \text{and} \quad \tilde{P}(0|0) = 0.01
\end{aligned} \tag{6.46}$$

A *XMBP* run is depicted in Figure 6.3. Here we see that the state estimate begins tracking the true state after the initial transient. The estimation error is reasonable ($\sim 1\%$ lie outside limits) indicating the filter is performing properly for this realization. The filtered measurement and innovations are shown in Figure 6.3*b* and lie within the predicted limits. The innovations are zero mean ($6.3 \times 10^{-2} < 11.8 \times 10^{-2}$) and white (0% lie outside limits) as shown in Figure 6.3*c*, indicating proper tuning. Comparing the *XMBP* to the *LZ-MBP* of the previous section shows that it does not perform as well as indicated by the predicted covariance limits for the estimated measurement and innovations. Most of this error is caused by the initial conditions of the processor.

Running an ensemble of 101 realizations of this processor yields similar results for the ensemble estimates: state estimation error increased to ($\sim 2\%$ outside limits), innovations zero mean test increased slightly ($6.7 \times 10^{-2} < 12 \times 10^{-2}$) and whiteness was identical. This completes the nonlinear filtering example. We summarize the results as follows:

| | |
|---|---|
| Criterion: | $J = \text{trace } \tilde{P}(t\|t)$ |
| Models: | |
| Signal: | $x(t) = (1-0.05\triangle T)x(t-1) + 0.04x^2(t-1) + w(t-1)$ |
| Measurement: | $y(t) = x^2(t) + x^3(t) + v(t)$ |
| Noise: | $w \sim \mathcal{N}(0,0)$ and $v \sim \mathcal{N}(0, 0.09)$ |
| Initial conditions: | $\hat{x}(0\|0) = 2.3$ and $\tilde{P}(0\|0) = 0.01$ |
| Algorithm: | $\hat{x}(t\|t) = \hat{x}(t\|t-1) + K(t)e(t)$ |
| Quality: | $\tilde{P}(t\|t) = \left(1-K(t)[2\hat{x}(t\|t-1) + 3\hat{x}^2(t\|t-1)]\right) \tilde{P}(t\|t-1)$ |

Next we consider one of the most popular applications of this approach—the *tracking problem*. The *XMBP* algorithm has bee applied to solve various tracking problems ([1], [5], [10], [15]). The choice of the coordinate system for the tracker determines whether the nonlinearities occur either in the state equations (polar

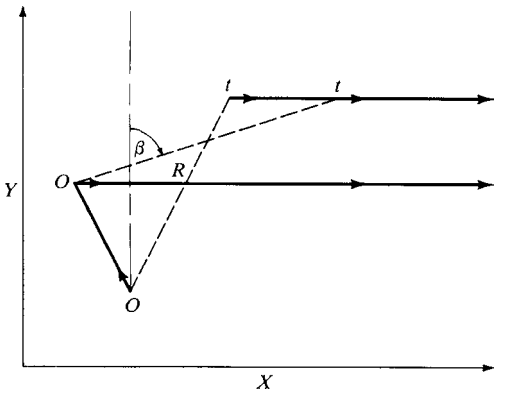



**Figure 6.4.** Observer/target ground track geometry for the XMBP tracking application.

coordinates [17]) or in the measurement equations (cartesian coordinates [18]). The following application depicts typical *XMBP* performance in cartesian coordinates.

***Example 6.3*** Consider the following passive localization and tracking problem that frequently arises in sonar and navigation applications [10]. For this example, consider a maneuvering observer $O$ monitoring noisy "bearings-only" measurements from a target $t$ assumed to be traveling at a constant velocity. These measurements are to be used to estimate target position $r$ and velocity $v$. The problem is geometrically depicted in Fig. 6.4. The velocity and position of the target relative to the observer are defined by

$$v_x(t) := v_{tx}(t) - v_{ox}(t), \quad r_x(t) := r_{tx}(t) - r_{ox}(t)$$
$$v_y(t) := v_{ty}(t) - v_{oy}(t), \quad r_y(t) := r_{ty}(t) - r_{oy}(t)$$

where the subscripts $t$ and $o$ refer to the target and observer respectively.

The velocity is related to the position by

$$v(t) = \frac{d}{dt}r(t) \approx \frac{r(t) - r(t-1)}{\triangle T} \qquad \text{for } \triangle T \text{ the sampling interval}$$

or

$$r(t) = r(t-1) + \triangle T\, v(t-1)$$

and

$$v(t) = v(t-1) + [v(t) - v(t-1)]$$
$$v(t) = [v_t(t-1) - v_o(t-1)] - [v_o(t) - v_o(t-1)] = v(t-1) - \triangle v_o(t-1)$$

since for a constant velocity target $v_t(t) = v_t(t-1) = \cdots = v_t$ and $\triangle v$ is the incremental change in observer velocity. Using these relations, we can easily develop a

Gauss-Markov model of the equations of motion in two dimensions by defining the state vector as $x' := [\ r_x \quad r_y \quad v_x \quad v_y\ ]$ and input $u' := [-\triangle v_{ox} - \triangle v_{oy}]$, but first let us consider the measurement model.

For this problem we have the bearing relation given by

$$\beta(x, t) := \arctan \frac{r_x(t)}{r_y(t)}$$

The entire system can be represented as a Gauss-Markov model with the noise sources representing uncertainties in the states and measurements. Thus, we have the equations of motion given by

$$x(t) = \begin{bmatrix} 1 & 0 & \triangle T & 0 \\ 0 & 1 & 0 & \triangle T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x(t-1) + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -\triangle v_{ox}(t-1) \\ \\ -\triangle v_{oy}(t-1) \end{bmatrix} + w(t-1)$$

with the nonlinear bearing model given by

$$y(t) = \arctan \frac{x_1(t)}{x_2(t)} + v(t)$$

for $w \sim \mathcal{N}(0, R_{ww})$ and $v \sim \mathcal{N}(0, R_{vv})$. The *SSPACK_PC* software [8] was used to simulate this system which is depicted in Figure 6.5 for two legs of a tracking scenario. An impulse-incremental step change ($\triangle v_{ox} = -24$ knots and $\triangle v_{oy} = +10$ knots) was initiated at 0.5 hour, resulting in a change of observer position and velocity depicted in the figure. The simulated bearing measurements are shown in Figure 6.5*d*, The initial conditions for the run were $x'(0) := [\ 0 \quad 15\ \text{nm} \quad 20\ \text{k} \quad -10\ \text{k}\ ]$ and $R_{ww} = \text{diag}\ 10^{-6}$ with the measurement noise covariance given by $R_{vv} = 3.05 \times 10^{-4}\text{rad}^2$ for $\triangle T = 0.33$ h.

The *XMBP* algorithm of Table 6.3 is implemented using this model, and the following jacobian matrices derived from the Gauss-Markov model above:

$$A[x] = A \quad \text{and} \quad C[x] = \begin{bmatrix} \dfrac{x_2(t)}{R^2} & \dfrac{-x_1(t)}{R^2} & 0 & 0 \end{bmatrix}$$

where

$$R = \sqrt{x_1^2(t) + x_2^2(t)}$$

The results of this run are shown in Figure 6.5. In Figure 6.5*a* and 6.5*b* we see the respective $x$ and $y$ position estimates (velocity estimates are not shown)

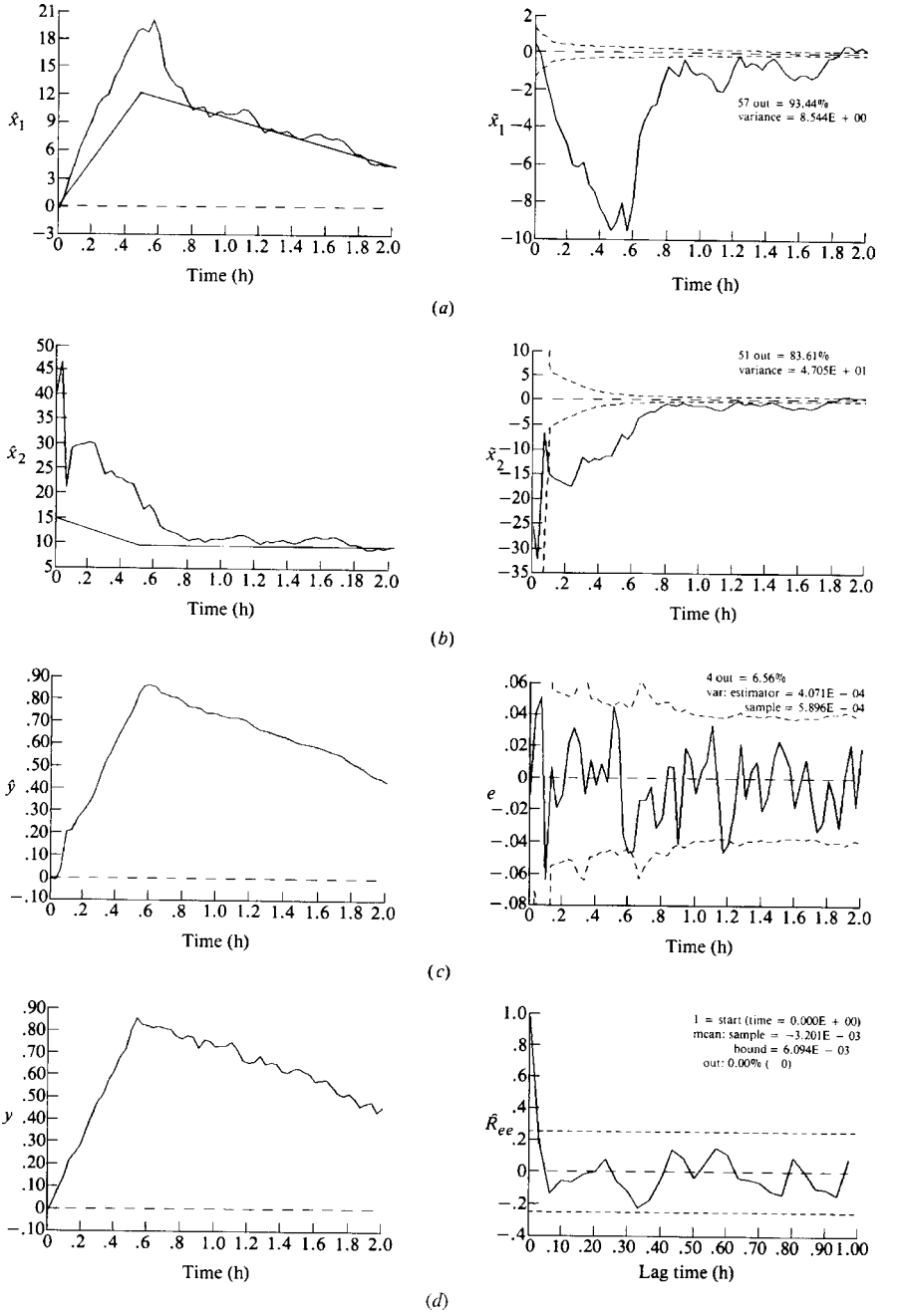



**Figure 6.5.** Extended MBP simulation for bearings-only tracking problem. *(a)* *X*-position estimate and error. *(b)* *Y*-position estimate and error. *(c)* Filtered measurement and error (innovation). *(d)* Simulated measurement and zero-mean/whiteness test ($3.2 \times 10^{-3} < 6.1 \times 10^{-3}$ and 0% out).

and corresponding tracking errors. Here we see that it takes approximately 1.8 hours for the estimator to converge to the true target position (within $\pm 1$ nm). The innovations sequence appears statistically zero-mean and white in Figure 6.5*c* and 6.5*d*, indicating satisfactory performance. The filtered measurement in 6.5*c* is improved considerably over the unprocessed data in 6.5*d*. This completes the example. We summarize the results as follows:

| | |
|---|---|
| Criterion: | $J = \text{trace } \tilde{P}(t\|t)$ |
| Models: | |
| Signal: | $x(t) = Ax(t-1) + bu(t-1) + w(t-1)$ |
| Measurement: | $y(t) = \arctan [x_1(t)/x_2(t)] + v(t)$ |
| Noise: | $w \sim \mathcal{N}(0, 10^{-6})$ and $v \sim \mathcal{N}(0, 3.05 \times 10^{-4})$ |
| Initial conditions: | $\hat{x}'(0\|0) = [0 \text{ 15 nm } 20 \text{ k} - 10 \text{ k}]$ and $\tilde{P}(0\|0) = \text{diag } 10^{-6}$ |
| Algorithm: | $\hat{x}(t\|t) = \hat{x}(t\|t-1) + K(t)e(t)$ |
| Quality: | $\tilde{P}(t\|t) = (I - K(t)C[x])\,\tilde{P}(t\|t-1)$ |

The *XMBP* algorithm can be applied as an online multiple-input, multiple-output system identifier for linear as well as nonlinear systems [1], [19]. We will develop this processor in Chapter 8, since it represents a "parametrically" adaptive, nonlinear state-space processor. This completes the section on the extension of the *MBP* to nonlinear problems. Next we investigate variants of the *XMBP* for improved performance.

## 6.3 ITERATED-EXTENDED *MBP* (ITERATED-EXTENDED KALMAN FILTER)

In this section we discuss an extension of the *XMBP* of the previous section to the iterated-extended (*IX-MBP*). We heuristically motivate the design and then discuss a more detailed approach using the approximate (linearized) Bayesian *MAP* estimation technique as in the *LZ-MBP* development coupled with numerical optimization techniques. This algorithm is based on performing "local" iterations (not global) at a point in time, $t$, to improve the reference trajectory and therefore the underlying estimate in the presence of significant measurement nonlinearities [1]. A local iteration implies that the inherent recursive structure of the processor is retained providing new estimates *prior* to receiving a new measurement.

To develop the iterated-extended processor, we start with the linearized processor correction relation, substituting the "linearized" innovation of Eq. (6.22) of the *LZ-MBP*, that is,

$$\begin{aligned}\hat{x}(t|t) = \hat{x}(t|t-1) &+ K\left(t; x^*(t)\right)[y(t) - c[x^*(t)] \\ &- C[x^*(t)]\left(\hat{x}(t|t-1) - x^*(t)\right)] \end{aligned} \tag{6.47}$$

where we have explicitly shown the dependence of the gain (through the measurement jacobian) on the reference trajectory, $x^*(t)$. The *XMBP* algorithm linearizes about the most currently available estimate, $x^*(t) = \hat{x}(t|t-1)$, in this case. Theoretically the corrected estimate, $\hat{x}(t|t)$, is a better estimate and closer to the true trajectory. Suppose that we continue and re-linearize about $\hat{x}(t|t)$, when it becomes available during the recursion and then recompute the iterated-corrected estimate and so on. That is, define the $(i+1)$th-iterated estimate as $\hat{x}_{i+1}(t|t)$, then the corrected *iterator* equation becomes

$$\begin{aligned}\hat{x}_{i+1}(t|t) = \hat{x}(t|t-1) + K\,(t;\hat{x}_i(t|t))\,[y(t) - c[\hat{x}_i(t|t)] \\ - C[\hat{x}_i(t|t)]\,(\hat{x}(t|t-1) - \hat{x}_i(t|t))]\end{aligned} \tag{6.48}$$

Now, if we start with the 0th iterate as the predicted estimate, that is, $\hat{x}_o \equiv \hat{x}(t|t-1)$, then the *XMBP* results for $i = 0$. Clearly, the corrected estimate in this iteration is given by

$$\begin{aligned}\hat{x}_1(t|t) = \hat{x}(t|t-1) + K_o(t)[y(t) - c[\hat{x}(t|t)] \\ -C[\hat{x}(t|t-1)]\,(\hat{x}(t|t-1) - \hat{x}(t|t-1))]\end{aligned} \tag{6.49}$$

where the last term in this expression is null leaving the usual innovation. We have also simplified the gain notation such that $K\,(t;\hat{x}_i(t|t)) \rightarrow K_i(t)$. Note also that the *gain* and therefore the corresponding error covariance are reevaluated at *each* iteration as are the measurement function and jacobian. The iterations continue until there is little difference in consecutive iterates. The *last* iterate is taken as the corrected estimate. The complete (corrected) iteration loop is given by

$$\begin{aligned} e_i(t) &= y(t) - c[\hat{x}_i(t|t)] \\ R_{e_i e_i}(t) &= C[\hat{x}_i(t|t)]\tilde{P}(t|t-1)C'[\hat{x}_i(t|t)] + R_{vv}(t) \\ K_i(t) &= \tilde{P}(t|t-1)C'[\hat{x}_i(t|t)]R^{-1}_{e_i e_i}(t) \\ \hat{x}_{i+1}(t|t) &= \hat{x}(t|t-1) + K_i(t)\,[e_i(t) - C[\hat{x}_i(t|t)]\,(\hat{x}(t|t-1) - \hat{x}_i(t|t))] \\ \tilde{P}_i(t|t) &= (I - K_i(t)C[\hat{x}_i(t|t)])\,\tilde{P}(t|t-1) \end{aligned} \tag{6.50}$$

A typical stopping rule is

$$\| \hat{x}_{i+1}(t|t) - \hat{x}_i(t|t) \| < \epsilon \quad \text{and} \quad \hat{x}_i(t|t) \rightarrow \hat{x}(t|t) \tag{6.51}$$

The *IX-MBP* algorithm is summarized in Table 6.4.

The *IX-MBP* can be useful in reducing the measurement function nonlinearities improving processor performance. It is designed for measurement nonlinearities and does not improve the previous reference trajectory, but it will improve on the subsequent one.

Next we take a slightly more formal approach to developing the *IX-MBP* from the approximate (linearized) Bayesian perspective as in the *LZ-MBP* of Section 6.1.

**Table 6.4. Discrete Iterated Extended *MBP* (Kalman Filter) Algorithm**

Prediction

$$\hat{x}(t|t-1) = a[\hat{x}(t-1|t-1)] + b[u(t-1)] \quad \text{(State prediction)}$$
$$\tilde{P}(t|t-1) = A[\hat{x}(t|t-1)]\tilde{P}(t-1|t-1)A'[\hat{x}(t|t-1)] + R_{ww}(t-1) \quad \text{(Covariance prediction)}$$

for $i = 1, 2 \ldots$

Innovation

$$e_i(t) = y(t) - c[\hat{x}_i(t|t)] \quad \text{(Innovation)}$$
$$R_{e_i e_i}(t) = C[\hat{x}_i(t|t)]\tilde{P}(t|t-1)C'[\hat{x}_i(t|t)] + R_{vv}(t) \quad \text{(Innovation covariance)}$$

Gain

$$K_i(t) = \tilde{P}(t|t-1)C'[\hat{x}_i(t|t)]R^{-1}_{e_i e_i}(t) \quad \text{(Gain or weight)}$$

Correction

$$\hat{x}_{i+1}(t|t) = \hat{x}(t|t-1) + K_i(t)\,[e_i(t) - C[\hat{x}_i(t|t)] \times (\hat{x}(t|t-1) - \hat{x}_i(t|t))] \quad \text{(State correction)}$$
$$\tilde{P}_i(t|t) = [I - K_i(t)C[\hat{x}_i(t|t)]]\,\tilde{P}(t|t-1) \quad \text{(Covariance correction)}$$

Initial conditions

$$\hat{x}(0|0), \quad \tilde{P}(0|0), \hat{x}_o(t|t) = \hat{x}(t|t-1)$$

Jacobians

$$A[\hat{x}(t|t-1)] \equiv \frac{da[x(t-1)]}{dx(t-1)}\Big|_{x=\hat{x}(t|t-1)}, \quad C[\hat{x}_i(t|t)] \equiv \frac{dc[x(t)]}{dx(t)}\Big|_{x=\hat{x}_i(t|t)}$$

Stopping rule

$$\| \hat{x}_{i+1}(t|t) - \hat{x}_i(t|t) \| < \epsilon \quad \text{and} \quad \hat{x}_i(t|t) \to \hat{x}(t|t)$$

That is, we first formulate a parametric optimization problem to develop the generic structure of the iterator (as in Section 4.5), then apply it to the underlying linearized state estimation problem.

Let us assume that we have a nonlinear cost function, $\mathcal{J}(\Theta)$, that we would like to maximize relative to the parameter vector, $\Theta \in R^{N_\Theta \times 1}$. We begin by expanding the cost in terms of a Taylor series about $\Theta_i$, that is,

$$\mathcal{J}(\Theta) = \mathcal{J}(\Theta_i) + (\Theta - \Theta_i)'\,[\nabla_\Theta \mathcal{J}(\Theta_i)] + \tfrac{1}{2}(\Theta - \Theta_i)'\,[\nabla_{\Theta\Theta}\mathcal{J}(\Theta_i)]\,(\Theta - \Theta_i) + \text{H.O.T.} \tag{6.52}$$

where $\nabla_\Theta$ is the $N_\Theta$-gradient vector defined by

$$\nabla_\Theta \mathcal{J}(\Theta_i) := \frac{\partial \mathcal{J}(\Theta)}{\partial \Theta}\Big|_{\Theta=\Theta_i} \tag{6.53}$$

The corresponding $N_\Theta \times N_\Theta$ *Hessian* matrix is defined by

$$\nabla_{\Theta\Theta}\mathcal{J}(\Theta_i) := \frac{\partial^2 \mathcal{J}(\Theta)}{\partial \Theta^2}\Big|_{\Theta=\Theta_i} \tag{6.54}$$

Now, if we approximate this expression by neglecting the H.O.T. and assume that $\Theta_i$ is close to the *true* parameter vector ($\Theta_i \approx \Theta_{\text{true}}$), then differentiating Eq. (6.52) using the chain rule, we obtain

$$\nabla_\Theta \mathcal{J}(\Theta) = 0 + I\,[\nabla_\Theta \mathcal{J}(\Theta_i)] + \tfrac{1}{2}(2\,[\nabla_{\Theta\Theta}\mathcal{J}(\Theta_i)]\,(\Theta - \Theta_i)) = 0$$

or

$$[\nabla_{\Theta\Theta}\mathcal{J}(\Theta_i)]\,(\Theta - \Theta_i) = -\,[\nabla_\Theta \mathcal{J}(\Theta_i)]$$

Solving for $\Theta$ and letting $\Theta \to \Theta_{i+1}$, we obtain the well-known *Newton-Rhapson* iterator (*NRI*) as ([1], [10], [11])

$$\Theta_{i+1} = \Theta_i - [\nabla_{\Theta\Theta}\mathcal{J}(\Theta_i)]^{-1}\,[\nabla_\Theta \mathcal{J}(\Theta_i)] \tag{6.55}$$

This is the form that our *IX-MBP* will assume with $x_i \to \Theta_i$. So we return to the basic problem of improved state estimation using the *NRI*.

Under the usual gaussian assumptions, we would like to calculate the *MAP* estimate of the state at time $t$ based on the data up to time $t$. Therefore the a posteriori probability (see Sec. 5.4) is given by

$$\Pr(x(t)|Y(t)) = \frac{1}{\eta} \times \Pr(y(t)|x(t)) \times \Pr(x(t)|Y(t-1)) \tag{6.56}$$

where $\eta$ is a normalizing probability (not a function of the state) and can be ignored in this situation. As in the linear case, we have

1. $\Pr(y(t)|x(t)) \quad : \mathcal{N}(c[x(t)], R_{vv}(t))$
2. $\Pr(x(t)|Y(t-1)) \; : \mathcal{N}\left(\hat{x}(t|t-1), \tilde{P}(t|t-1)\right)$

Maximizing the a posteriori probability is equivalent to minimizing its logarithm. Therefore we have that

$$\begin{aligned} J(x(t)) &= \ln \Pr(y(t)|x(t)) + \ln \Pr(x(t)|Y(t-1)) - \ln \eta \\ &= -\tfrac{1}{2}(y(t) - c[x(t)])' R_{vv}^{-1}(t)(y(t) - c[x(t)]) \\ &\quad -\tfrac{1}{2}(x(t) - \hat{x}(t|t-1))' \tilde{P}^{-1}(t|t-1)(x(t) - \hat{x}(t|t-1)) - \ln \eta \end{aligned} \tag{6.57}$$

As before in the *LZ-MBP* case, we linearize the measurement nonlinearity about $x^*(t)$, that is,

$$c[x(t)] \approx c[x^*(t)] + C[x^*(t)]\left(x(t) - x^*(t)\right)$$

Substitute in Eq. (6.57) and differentiate to obtain

$$\nabla_x \mathcal{J}(x(t)) = -C'[x^*(t)]R_{vv}^{-1}(t)\left(y(t) - c[x^*(t)] - C[x^*(t)](x(t) - x^*(t))\right) - \tilde{P}^{-1}(t|t-1)\,[x(t) - \hat{x}(t|t-1)] \tag{6.58}$$

Letting $x^*(t) \rightarrow x_i^*(t)$ and $x(t) \rightarrow x_i(t)$ (the $i$th-iterates), we can write Eq. (6.58) as

$$\nabla_x \mathcal{J}(x(t)) = -C'[x_i^*(t)]R_{vv}^{-1}(t)\left(y(t) - c[x_i^*(t)] - C[x_i^*(t)](x_i(t) - x_i^*(t))\right) - \tilde{P}^{-1}(t|t-1)\,[x_i(t) - \hat{x}(t|t-1)] \tag{6.59}$$

which is the same form of the linear *MAP* estimator of Eq. (6.29) with $C[x_i^*(t)] \rightarrow C[x^*(t)]$.

Continuing with the *NRI* derivation, we differentiate Eq. (6.59) again to obtain the Hessian

$$\nabla_{xx} \mathcal{J}(x(t)) = C'[x_i^*(t)]R_{vv}^{-1}(t)C[x_i^*(t)] + \tilde{P}^{-1}(t|t-1) \tag{6.60}$$

Note that the iteration is only over the correction loop, so the predicted state and error covariance do *not* have $i$-subscripts.

Applying the *matrix inversion lemma* as in Section 5.4 (see Eq. 5.53 for details), we can easily show that

$$[\nabla_{xx} \mathcal{J}(x(t))]^{-1} = (I - K_i(t)C[x_i^*(t)])\tilde{P}(t|t-1) \equiv \tilde{P}_i(t|t) \tag{6.61}$$

for

$$K_i(t) := K_i(t; x_i^*(t)) = \tilde{P}(t|t-1)C'[x_i^*(t)]R_{e_i e_i}^{-1}(t) \tag{6.62}$$

with

$$R_{e_i e_i}(t) = C[x_i^*(t)]\tilde{P}(t|t-1)C'[x_i^*(t)] + R_{vv}(t) \tag{6.63}$$

Now, using Eq. (6.59) and Eq. (6.61) in terms of the corrected error covariance and solving for $\tilde{P}^{-1}(t|t-1)$, we obtain

$$\tilde{P}^{-1}(t|t-1) = \tilde{P}_i^{-1}(t|t) - C'[x_i^*(t)]R_{vv}^{-1}(t)C[x_i^*(t)] \tag{6.64}$$

which can be substituted back into Eq. (6.59) to give

$$\nabla_x \mathcal{J}(x(t)) = C'[x_i^*(t)]R_{vv}^{-1}(t)e_i(t) - \left[\tilde{P}_i^{-1}(t|t) - C'[x_i^*(t)]R_{vv}^{-1}(t)C[x_i^*(t)]\right](x_i(t) - \hat{x}(t|t-1)) \tag{6.65}$$

The *NRI* can now be written in terms of the *iterated state* at time $t$ as

$$x_{i+1}(t) = x_i(t) - [\nabla_{xx}\mathcal{J}(x(t))]^{-1}\,\nabla_x\mathcal{J}(x(t)) \tag{6.66}$$

Using the expressions for the Hessian and gradient and state estimation error $\tilde{x}_i(t|t-1) := x_i(t) - \hat{x}(t|t-1)$, we have

$$\begin{aligned} x_{i+1}(t) = x_i(t) + \tilde{P}_i(t|t) \times \Big(C'[x_i^*(t)]R_{vv}^{-1}(t)e_i(t) \\ - \Big[\tilde{P}_i^{-1}(t|t) - C'[x_i^*(t)]R_{vv}^{-1}(t)C[x_i^*(t)]\Big]\tilde{x}_i(t|t-1)\Big) \end{aligned} \tag{6.67}$$

Multiplying through by the Hessian (corrected-iterated error covariance) and recognizing the alternate expression for the gain (see Eq. 5.62), we obtain the expression

$$x_{i+1}(t) = x_i(t) + K_i(t)e_i(t) - \big[I - K_i(t)C[x_i^*(t)]\big]\,(x_i(t) - \hat{x}(t|t-1)) \tag{6.68}$$

Now multiplying by $\tilde{x}_i(t|t-1)$ of the second term, factoring out the gain, and performing the additions, we obtain

$$x_{i+1}(t) = \hat{x}(t|t-1) + K_i(t)\,\big(e_i(t) - C[x_i^*(t)]\big)\,\big(\hat{x}(t|t-1) - x_i^*(t)\big) \tag{6.69}$$

Defining the iterate in terms of the current iterated-corrected state estimate, that is, $x_i^*(t) \to \hat{x}_i(t|t)$ gives the *NRI* iterator of Eq. (6.50) and Table 6.4.

So we see that for strong measurement nonlinearities the *IX-MBP* can be used at little cost to the *XMBP*. A further extension of these results is called the *iterator-smoother XMBP* in which the entire processor is iterated to mitigate strong nonlinearities in the predicted estimates as well [1]. Here the measurement is relinearized and then a "smoothed" state estimate backward to the prediction loop. This completes the discussion of the processor, next we demonstrate its performance.

***Example 6.4*** Consider the discrete nonlinear process and measurement system described in the previous example. The simulated measurement using *SSPACK_PC* [8] is shown in Figure 6.6*c* . The *IX-MBP* is designed from the following jacobians:

$$A[x(t-1)] = 1 - 0.05\triangle T + 0.08\triangle T x(t-1) \quad \text{and} \quad C[x(t)] = 2x(t) + 3x^2(t)$$

The *IX-MBP* algorithm is then given by

$$\begin{aligned} \hat{x}(t|t-1) &= (1 - 0.05\triangle T)\hat{x}(t-1|t-1) + 0.04\hat{x}^2(t-1|t-1) \\ \tilde{P}(t|t-1) &= [1 - 0.05\triangle T + 0.08\triangle T x(t-1)]^2\tilde{P}(t-1|t-1) \\ e_i(t) &= y(t) - \hat{x}_i^2(t|t) - \hat{x}_i^3(t|t) \\ R_{e_ie_i}(t) &= [2\hat{x}_i(t|t) + 3\hat{x}_i^2(t|t)]^2\tilde{P}(t|t-1) + 0.09 \\ K_i(t) &= \tilde{P}(t|t-1)[2\hat{x}_i(t|t) + 3\hat{x}_i^2(t|t)]/R_{e_ie_i}(t) \end{aligned}$$

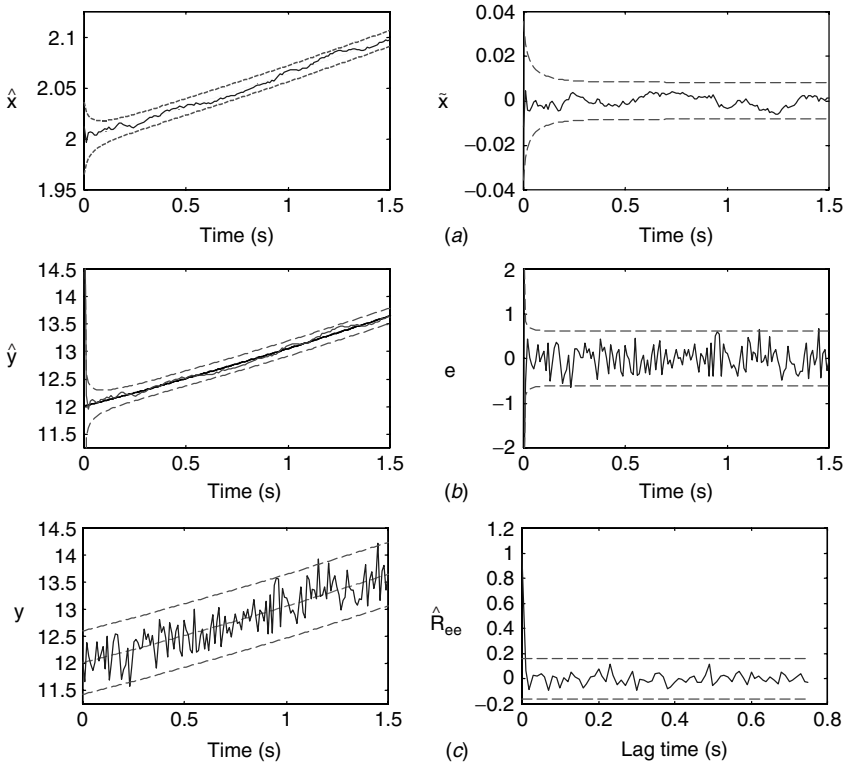



**Figure 6.6.** Iterated-extended MBP simulation. (*a*) Estimated state (∼ 0% out) and error (∼ 0% out). (*b*) Filtered measurement (∼ 1% out) and error (innovation) (∼ 2.6% out). (*c*) Simulated measurement and zero-mean/whiteness test ($4 \times 10^{-2} < 10.7 \times 10^{-2}$ and 0% out).

$$\hat{x}_{i+1}(t|t) = \hat{x}(t|t-1) + K_i(t)[e_i(t) - [2\hat{x}_i(t|t) + 3\hat{x}_i^2(t|t)](\hat{x}(t|t-1) - \hat{x}_i(t|t)$$

$$\tilde{P}_i(t|t) = \left(1 - K_i(t)[2\hat{x}_i(t|t) + 3\hat{x}_i^2(t|t)]\right)\tilde{P}(t|t-1)$$

$$\hat{x}(0|0) = 2.3 \quad \text{and} \quad \tilde{P}(0|0) = 0.01$$

A *IX-MBP* run is depicted in Figure 6.6. Here we see that the state estimate (∼0% lie outside limits) begins tracking the true state after the initial iterations (3). The estimation error is reasonable (∼ 0% out) indicating the filter is performing properly for this realization. The filtered measurement (∼1% out) and innovations (∼ 2.6% out) are shown in Figure 6.6*b*. The innovations are zero-mean ($4 \times 10^{-2} < 10.7 \times 10^{-2}$) and white (0% lie outside limits) as shown in Figure 6.6*c*, indicating proper tuning and matching the *LZ-MBP* result almost exactly.

Running an ensemble of 101 realizations of this processor yields similar results for the ensemble estimates: innovations zero mean test decreased slightly ($3.7 \times 10^{-2} < 10.3 \times 10^{-2}$) and whiteness was identical. So the overall effect of the

*IX-MBP* was to decrease the measurement nonlinearity effect especially in the initial transient of the algorithm. These results are almost identical to those of the *LZ-MBP*. This completes the nonlinear filtering example. We summarize the results as follows:

Criterion: $J = \text{trace } \tilde{P}(t|t)$

Models:

Signal: $x(t) = (1 - 0.05\triangle T)x(t-1) + 0.04x^2(t-1) + w(t-1)$

Measurement: $y(t) = x^2(t) + x^3(t) + v(t)$

Noise: $w \sim \mathcal{N}(0, 0)$ and $v \sim \mathcal{N}(0, 0.09)$

Initial conditions: $\hat{x}(0|0) = 2.3$ and $\tilde{P}(0|0) = 0.01$

Algorithm: $\hat{x}_i(t|t) = \hat{x}(t|t-1) + K_i(t)[e_i(t) - [2\hat{x}_i(t|t) + 3\hat{x}_i^2(t|t)]$
$(\hat{x}(t|t-1) - \hat{x}_i(t|t))$

Quality: $\tilde{P}(t|t) = \left(1 - K_i(t)[2\hat{x}_i(t|t) + 3\hat{x}_i^2(t|t)]\right)\tilde{P}(t|t-1)$

Next we consider a new "statistical approach" to the nonlinear *MBP* problem.

## 6.4 UNSCENTED *MBP* (KALMAN FILTER)

In this section we discuss an extension of the approximate nonlinear *MBP* suite of processors which takes a distinctly different approach to the nonlinear gaussian problem. Instead of attempting to improve on the linearized approximation in the nonlinear *XMBP* schemes discussed in the previous section or increasing the order of the Taylor series approximations [2], [20], a statistical *transformation* approach is developed. It is founded on the basic idea that "it is easier to approximate a probability distribution, then to approximate an arbitrary nonlinear function of transformation" [21], [22], [23]. The nonlinear model-based processors discussed so far are based on linearizing nonlinear functions of the state and input to provide estimates of the underlying statistics (using jacobians), while the transformation approach is based on selecting a set of sample points that capture certain properties of the underlying distribution. This set of samples is then nonlinearly transformed or propagated to a new space. The statistics of the new samples are then calculated to provide the required estimates. Note that this method differs from the well known *sampling-resampling* approach in which *random* samples are drawn from the prior distribution and updated through the likelihood function to produce a sample from the posterior distribution [24]. Here the samples are *not* drawn at random, but according to a specific *deterministic* algorithm. Once this transformation is performed, the resulting processor, the unscented model-based processor (*UMBP*) or equivalently the unscented Kalman filter (*UKF*) evolves. The *UMBP* is a recursive processor that resolves some of the approximation issues [21] and deficiencies of the *XMBP* of the previous sections. We first develop the idea of nonlinearly transforming the probability distribution and then apply it to our gaussian problem leading to the *UMBP* algorithm. We apply the processor to the previous nonlinear state estimation problem and investigate its performance.

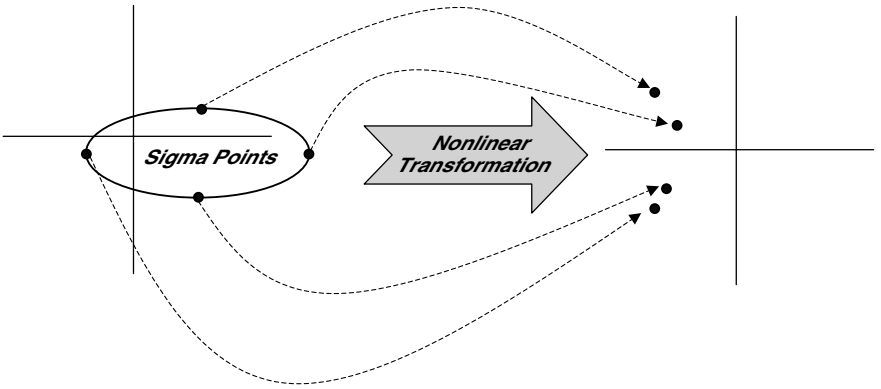



**Figure 6.7.** Unscented transformation. The set of distribution points on an error ellipsoid are selected and transformed into a new space where their underlying statistics are estimated.

### 6.4.1 Unscented Transformations

The unscented transformation ($UT$) is a technique for calculating the statistics of a random variable that has been nonlinearly transformed and use this transform to establish the basis of our processor. The approach is illustrated in Figure 6.7. Here the set of samples, called *sigma points*, are chosen so that they capture the specific properties of the underlying distribution. Each of the $\sigma$-points is nonlinearly transformed to create a set of samples in the new space. The statistics of the transformed samples are then calculated to provide the desired estimates.

Following the development of Julier [25], consider propagating an $N_x$-dimensional random vector, $\mathbf{x}$, through an arbitrary nonlinear transformation $\mathbf{a}[\cdot]$ to generate a new random vector,

$$\mathbf{z} = \mathbf{a}[\mathbf{x}] \tag{6.70}$$

The set of $\sigma$-points, $\{\mathcal{X}_i\}$, consists of $N_x + 1$ vectors with appropriate weights, $\{\mathcal{W}_i\}$, given by $\Sigma = \{i = 0, \ldots, N_x : \mathcal{X}_i, W_i\}$. The weights can be positive or negative but *must satisfy* the *normalization constraint* that

$$\sum_{i=0}^{N_x} W_i = 1$$

The problem then is recast:

> **GIVEN** these $\sigma$-points and the nonlinear transformation of Eq. (6.70). **FIND** the statistics of the transformed samples,
>
> $$\mu_z \equiv E\{\mathbf{z}\} \text{ and } \mathbf{R}_{zz} = \text{cov } (\mathbf{z})$$

The *unscented transformation* (*UT*) approach is as follows:

1. Nonlinearly *transform* each point to obtain the set of new $\sigma$-points:

$$\mathcal{Z}_i = \mathbf{a}[\mathcal{X}_i] \tag{6.71}$$

2. Estimate the posterior *mean* by its weighted average

$$\mu_{\mathbf{z}} = \sum_{i=0}^{N_x} W_i \mathcal{Z}_i \tag{6.72}$$

3. Estimate the posterior *covariance* by its weighted outer product

$$\mathbf{R}_{zz} = \sum_{i=0}^{N_x} W_i \left(\mathcal{Z}_i - \mu_z\right)\left(\mathcal{Z}_i - \mu_z\right)' \tag{6.73}$$

The critical issues to decide are (1) $N_x$, the number of $\sigma$-points, (2) $W_i$, the weights assigned to each $\sigma$-point, and (3) *where* the $\sigma$-points are to be located. The $\sigma$-points should be selected to capture the "most important" properties of the random vector, $\mathbf{x}$; that is, let $\Pr(\mathbf{x})$ be its density function, then the $\sigma$-points capture the properties by satisfying the *necessary condition*

$$\mathbf{g}\left[\sigma, \Pr(\mathbf{x})\right] = \mathbf{0} \tag{6.74}$$

Since it is possible to meet this condition and still have some degree of freedom in the choice of $\sigma$-points, assigning a *penalty* function

$$\mathbf{p}\left[\sigma, \Pr(\mathbf{x})\right] \tag{6.75}$$

resolves any ambiguity in the choice. This function is to incorporate desirable features that do not necessarily have to be satisfied. Decreasing the penalty function leads to more and more desirable solutions. Thus, in general, the $\sigma$-point set relative to our problem that is the most desirable is the set that conforms to the necessary conditions of Eqs. (6.74) and (6.75), that is, the $\sigma$-points must *satisfy*

$$\min_{\sigma} \ \mathbf{p}\left[\sigma, \Pr(\mathbf{x})\right] \ni \mathbf{g}\left[\sigma, \Pr(\mathbf{x})\right] = \mathbf{0} \tag{6.76}$$

The decision as to which properties of the random vector $\mathbf{x}$ to capture precisely or approximately depends on the particular application. For our problems, we wish to match the *moments* of the underlying distribution of $\sigma$-points with those of $\mathbf{x}$.

To be more precise and parallel the general philosophy, we choose "to approximate the underlying gaussian distribution rather than approximate its underlying nonlinear transformation," in contrast to the *XMBP*. This parameterization captures the *mean* and *covariance* information and permits the direct propagation of this information through the arbitrary set of nonlinear functions. Here we accomplish

this approximately by generating the discrete distribution having the same first and second (and potentially higher) moments where each point is directly transformed. The mean and covariance of the transformed ensemble can then be computed as *the estimate* of the nonlinear transformation of the original distribution.

Following Julier [22] consider the following one-dimensional example.

***Example 6.5*** Suppose that we have a scalar random variable $x$, gaussian distributed with mean $\mu_x$ and variance, $\sigma_x^2$. We would like to know the statistics (mean and variance) of $z$ which nonlinearly transforms $x$ according to

$$z = a[x] = x^2$$

It is well known [26] that the solution to this problem gives the true mean and variance as

$$\mu_z = E\{z\} = E\{x^2\} = \sigma_x^2 + \mu_x^2$$

and

$$\begin{aligned}\sigma_z^2 &= E\{z^2\} - \mu_z^2 = E\{x^4\} - \mu_z^2 = (3\sigma_x^4 + 6\sigma_x^2\mu_x^2 + \mu_x^4) - (\sigma_x^4 + 2\sigma_x^2\mu_x^2 + \mu_x^4)) \\ &= 2\sigma_x^4 + 4\sigma_x^2\mu_x^2\end{aligned}$$

According to Eqs. (6.71) through (6.73) the position of the $\sigma$-points are determined first. Since this is a scalar problem only three samples are of interest: the two $\sigma$-points and the mean. Therefore we have

$$\{\mathcal{X}_0, \mathcal{X}_1, \mathcal{X}_2\} = \{\mu_x, \mu_x - \sigma, \mu_x + \sigma\} \text{ where } \sigma = \sqrt{(1+\kappa)\sigma_x^2}$$

Propagating these samples through $a[\cdot]$ gives the transformed samples, say $\mathcal{X}_i'$ that lie at

$$\{\mathcal{X}_0', \mathcal{X}_1', \mathcal{X}_2'\} = \{\mu_x^2, (\mu_x - \sigma)^2, (\mu_x + \sigma)^2\}$$

The mean of $z$ is given by

$$\mu_z = \frac{1}{2(1+\kappa)}\left(2\kappa\mu_x^2 + 2{\mu_x}^2 + 2(1+\kappa)\sigma_x^2\right) = \mu_x^2 + \sigma_x^2$$

which is precisely the true mean. Next the covariance is given by

$$\sigma_z^2 = \frac{1}{2(1+\kappa)}\left(\sum_{i=1}^{2}\left((\mathcal{X}_i' - \mu_z)^2 + 2\kappa\sigma_x^4\right)\right) = \kappa\sigma_x^4 + 4\mu_x^2\sigma_x^2$$

To find the solution, $\kappa$ must be specified. The kurtosis of the true distribution is $2\sigma_x^4$, and that of the $\sigma$-points is $\sigma_x^2$. Since the kurtosis of the points is scaled by an amount $(1+\kappa)$, the kurtosis of both distributions only agree when $\kappa = 2$, which gives the exact solution. This completes the gaussian example of the *UT*.

It is important to recognize that the *UT* has specific properties when the underlying distribution is gaussian [27]. The gaussian has two properties that play a significant role in the form of the $\sigma$-points *selected*. First, since the distribution is *symmetric*, the $\sigma$-points can be selected with this symmetry. Second, the problem of approximating $\mathbf{x}$ with an arbitrary mean and covariance can be reduced to that of a standard zero-mean, unit variance gaussian, since

$$\mathbf{x} = \mu_{\mathbf{x}} + U\mathbf{s} \quad \text{for } U \text{ the matrix square root of } P_{xx} \tag{6.77}$$

where $\mathbf{s} \sim \mathcal{N}(0, I)$. Therefore in the gaussian case, the second-order *UT* uses a set of $\sigma$-points that captures the first two moments of $\mathbf{s}$ correctly. That is, they must capture the mean, covariance, and symmetry. Let $s_i$ be the $i$th component of $\mathbf{s}$. Then the covariance is given by

$$E\{s_i^2\} = 1 \quad \forall i \tag{6.78}$$

and all *odd*-ordered moments are zero.

The minimum number of points whose distribution obeys these conditions has two types of $\sigma$-points: (1) a single point at the origin of the $\mathbf{s}$-axis with weight, $W_o$, and (2) $2N_x$ symmetrically distributed points on the coordinate $\mathbf{s}$-axis a distance $r$ from the origin all having the same weight, $W_1$. Thus there are $(2N_x + 1)$ $\sigma$-points for a two-dimensional distribution. The values of $W_o$, $W_1$, and $r$ are selected to ensure that their covariance is the identity. Therefore, due to their symmetry, it is only necessary to specify one direction on the $\mathbf{s}$-axis, say $s_1$. The constraint function will consist of the moment for $E\{s_1^2\}$ and the normalization condition must be satisfied. Therefore we have that

$$\mathbf{g}[\sigma, \Pr(\mathbf{s})] = \begin{pmatrix} 2W_1 r^2 - 1 \\ W_o + 2N_x W_1 - 1 \end{pmatrix} = \mathbf{0} \tag{6.79}$$

The solution to these equations is given by

$$r = \frac{1}{\sqrt{2W_1}} \quad \text{and} \quad W_o = 1 - 2N_x W_1, \qquad W_o \text{ free} \tag{6.80}$$

By reparameterizing $W_1 := 1/2(N_x + \kappa)$, it can be shown that after premultiplying by $U$, the matrix square root of $P_{xx}$, the $\sigma$-points for $\mathbf{x}$ are

$$\begin{aligned}
\mathcal{X}_o &= \mu_x, \qquad W_o = \frac{\kappa}{(N_x + \kappa)} \\
\mathcal{X}_i &= \mu_x + \left(\sqrt{(N_x + \kappa)\ P_{xx}}\right)_i, \qquad W_i = \frac{1}{2(N_x + \kappa)} \\
\mathcal{X}_{i+N_x} &= \mu_x - \left(\sqrt{N_x + \kappa}\ P_{xx}\right)_i, \qquad W_{i+N_x} = \frac{1}{2(N_x + \kappa)}
\end{aligned}$$

where $\kappa$ is a scalar, $\left(\sqrt{(N_x+\kappa)\ P_{xx}}\right)_i$ is the $i$th row or column of the matrix square root of $(N_x+\kappa)P_{xx}$ and $W_i$ is the weight associated with the $i$th sigma-point. The parameter $\kappa$ is free; however, it can be selected to minimize the mismatch between the fourth-order moments of the $\sigma$-points and the true distribution [27]. For instance from the properties of the gaussian, we have that

$$E\{s_i^4\} = 3 \qquad \forall\ i \tag{6.81}$$

The cost function penalizes the discrepancy between the $\sigma$-points and the true value along one direction ($s_1$), in this case, due to the symmetry. Therefore we have that

$$\mathbf{p}[\sigma, \Pr(\mathbf{s})] = |2W_1 r^4 - 3| \quad \text{giving} \quad W_1 = \frac{3}{2r^4} = \frac{1}{6} \quad \text{or} \quad \kappa = N_x - 3 \tag{6.82}$$

It is important to note that errors in kurtosis are minimized to the best of the ability of the $2N_x + 1$ samples; however, the kurtosis cannot be matched exactly without developing a larger set of $\sigma$-points (see [27] for details).

Next, using the principles behind the *UT*, we develop the unscented *MBP* under the multivariate gaussian assumptions.

### 6.4.2 Unscented Processor

The *UMBP* or unscented Kalman filter is a recursive processor developed to eliminate some of the deficiencies created by the failure of the linearization process to first order (Taylor series) in solving the state-estimation problem. Unlike the *XMBP*, the *UMBP* does not approximate the nonlinear process and measurement models. It employs the true nonlinear models and approximates the underlying *gaussian* distribution function of the state variable. In the *UMBP* the state is still represented as gaussian, but it is specified using the minimal set of deterministically selected samples or $\sigma$-points. These points completely capture the true mean and covariance of the gaussian distribution. When they are propagated through the true nonlinear process, the *posterior* mean and covariance are accurately captured to the second order for *any* nonlinearity with errors only introduced in the third and higher order moments.

In this subsection we develop the *UMBP* under the *multivariate* gaussian assumptions. That is, suppose that we are given an $N_x$-dimensional gaussian distribution having covariance, $P_{xx}$. Then we can generate a set of $O(N_x)$ $\sigma$-points having the same sample covariance from the columns (or rows) of the matrices $\pm\sqrt{(N_x+\kappa)\ P_{xx}}$. Here $\kappa$ is a scaling factor. This set is zero mean, but if the original distribution has mean $\mu_x$, then simply adding $\mu_x$ to each of the $\sigma$-points yields a symmetric set of $2N_x$ samples having the desired mean and covariance. Since the set is symmetric, its odd central moments are null, so its first three moments are identical to those of the original gaussian distribution. This is the *minimal* number of $\sigma$-points capable of capturing the essential statistical information. The basic *UT* technique for a multivariate gaussian distribution [27] is therefore as follows:

1. Compute the set of $2N_x$ $\sigma$-points from the rows or columns of $\pm\sqrt{(N_x+\kappa)\ P_{xx}}$. For the nonzero mean case compute $\mathcal{X}_i = \sigma + \mu_x$;

$$\mathcal{X}_o = \mu_x, \qquad W_o = \frac{\kappa}{(N_x+\kappa)}$$

$$\mathcal{X}_i = \mu_x + \left(\sqrt{(N_x+\kappa)\ P_{xx}}\right)_i, \qquad W_i = \frac{1}{2(N_x+\kappa)}$$

$$\mathcal{X}_{i+N_x} = \mu_x - \left(\sqrt{(N_x+\kappa)\ P_{xx}}\right)_i, \qquad W_{i+N_x} = \frac{1}{2(N_x+\kappa)}$$

   where $\kappa$ is a scalar, $\left(\sqrt{(N_x+\kappa)\ P_{xx}}\right)_i$ is the $i$th row or column of the matrix square root of $(N_x+\kappa)P_{xx}$ and $W_i$ is the weight associated with the $i$th $\sigma$-point;
2. Nonlinearly *transform* each point to obtain the set of $\sigma$-points: $\mathcal{Z}_i = \mathbf{a}[\mathcal{X}_i]$
3. Estimate the posterior *mean* of the new samples by its weighted average

$$\mu_{\mathbf{z}} = \sum_{i=0}^{N_x} W_i \mathcal{Z}_i$$

4. Estimate the posterior *covariance* of the new samples by its weighted outer product

$$\mathbf{R}_{zz} = \sum_{i=0}^{N_x} W_i \left(\mathcal{Z}_i - \mu_{\mathbf{z}}\right)\left(\mathcal{Z}_i - \mu_{\mathbf{z}}\right)'$$

There are a wealth of properties of this processor that we summarize below:

1. The transformed statistics of $\mathbf{z}$ are captured *precisely* up to second order.
2. The $\sigma$-points capture the identical mean and covariance regardless of the choice of matrix square-root method.
3. The posterior mean and covariance are calculated using standard linear algebraic methods, and it is *not* necessary to evaluate any jacobians as required by the *XMBP* methods.
4. $\kappa$ is a "tuning" parameter used to tune the higher order moments of the approximation that can be used to reduce overall prediction errors. For $\mathbf{x}$ multivariate gaussian, $N_x + \kappa = 3$ is a useful heuristic.
5. A modified form for $\kappa$ can be used to overcome a nonpositive definiteness of the covariance (see [28], [29], [30]).

We use the *XMBP* formulation and its underlying statistics as our prior distribution specified by the following nonlinear model with the conditional gaussian distributions. Recall that the original discrete nonlinear process model is given by

$$x(t) = \mathbf{a}[x(t-1)] + \mathbf{b}[u(t-1)] + w(t-1) \tag{6.83}$$

with corresponding measurement model

$$y(t) = \mathbf{c}[x(t)] + v(t) \tag{6.84}$$

for $w \sim \mathcal{N}(0, R_{ww})$ and $v \sim \mathcal{N}(0, R_{vv})$. It was demonstrated previously in Section 6.3 that the critical conditional gaussian distribution for the *state variable* statistics was the prior

$$x \sim \mathcal{N}\left(\hat{x}(t|t-1), \tilde{P}(t|t-1)\right)$$

and with the eventual measurement statistics specified by

$$y \sim \mathcal{N}\left(\hat{y}(t|t-1), R_{\xi\xi}(t|t-1)\right)$$

where $\hat{x}(t|t-1)$, and $\tilde{P}(t|t-1)$ are the respective predicted state and error covariance based on the data up to time $(t-1)$ and $\hat{y}(t|t-1)$, and $R_{\xi\xi}(t|t-1)$ are the predicted measurement and residual covariance. The idea then is to use the "prior" statistics and perform the *UT* (under gaussian assumptions) with *both* the process and nonlinear transformations (models) as specified above to obtain a set of transformed $\sigma$-points. Then the selected $\sigma$-points for the gaussian are transformed using the process and measurement models yielding the corresponding set of $\sigma$-points in the new space. The predicted means are weighted sums of the transformed $\sigma$-points and covariances are merely weighted sums of their mean-corrected, outer products.

We develop the *UMBP* in three steps:

- PREDICT the next state and error covariance, $(\hat{x}(t|t-1), \tilde{P}(t|t-1))$, by *UT* transforming the prior, $[\hat{x}(t-1|t-1), \tilde{P}(t-1|t-1)]$, including the process noise using the $\sigma$-points $\mathcal{X}(t|t-1)$ and $\mathcal{X}(t-1|t-1)$.
- PREDICT the measurement and residual covariance, $[\hat{y}(t|t-1), R_{\xi\xi}(t|t-1)]$ by using the *UT* transformed $\sigma$-points $\mathcal{Y}(t|t-1)$ and performing the weighted sums.
- PREDICT the cross-covariance, $R_{\tilde{x}\xi}(t|t-1)$ in order to calculate the corresponding gain for the subsequent correction step.

We use these steps as our road map to develop the *UMBP*. The $\sigma$-points for the *UT* transformation of the "prior" state information is specified with $\mu_x = \hat{x}(t-1|t-1)$ and $P_{xx} = \tilde{P}(t-1|t-1)$. Therefore we have the *UMBP* algorithm given as follows:

1. Select the $(2N_x + 1)$-points as

$$\mathcal{X}_o = \mu_x = \hat{x}(t-1|t-1), \quad W_o = \frac{\kappa}{(N_x + \kappa)}$$

$$\mathcal{X}_i = \mu_x + \left(\sqrt{(N_x+\kappa)\ P_{xx}}\right)_i$$
$$= \hat{x}(t-1|t-1) + \left(\sqrt{(N_x+\kappa)\ (\tilde{P}(t-1|t-1) + R_{ww}(t-1))}\right)_i,$$
$$W_i = \frac{1}{2(N_x+\kappa)}$$
$$\mathcal{X}_{i+N_x} = \mu_x - \left(\sqrt{(N_x+\kappa)\ P_{xx}}\right)_i$$
$$= \hat{x}(t-1|t-1) - \left(\sqrt{(N_x+\kappa)\ (\tilde{P}(t-1|t-1) + R_{ww}(t-1))}\right)_i,$$
$$W_{i+N_x} = \frac{1}{2(N_x+\kappa)}$$

2. Transform ($UT$) process model

$$\mathcal{X}_i(t|t-1) = \mathbf{a}[\mathcal{X}_i(t-1|t-1)] + \mathbf{b}[u(t-1)]$$

3. Estimate the posterior predicted (state) mean by

$$\hat{x}(t|t-1) = \sum_{i=0}^{2N_x} W_i \mathcal{X}_i(t|t-1)$$

4. Estimate the posterior predicted (state) residual and error covariance by

$$\tilde{\mathcal{X}}_i(t|t-1) = \mathcal{X}_i(t|t-1) - \hat{x}(t|t-1)$$
$$\tilde{P}(t|t-1) = \sum_{i=0}^{2N_x} W_i \tilde{\mathcal{X}}_i(t|t-1)\tilde{\mathcal{X}}_i'(t|t-1)$$

5. Transform ($UT$) measurement (model) with *augmented* $\sigma$-points to account for process noise as

$$\hat{\mathcal{X}}_i(t|t-1) = \Big[\mathcal{X}_i(t|t-1)\ \ \mathcal{X}_i(t|t-1) + \kappa\sqrt{R_{ww}(t-1)}\ \ \mathcal{X}_i(t|t-1)$$
$$-\kappa\sqrt{R_{ww}(t-1)}\Big]$$
$$\mathcal{Y}_i(t|t-1) = \mathbf{c}\left[\hat{\mathcal{X}}_i(t|t-1)\right]$$

6. Estimate the predicted measurement as

$$\hat{y}(t|t-1) = \sum_{i=0}^{2N_x} W_i \mathcal{Y}_i(t|t-1)$$

7. Estimate the predicted residual and covariance as

$$\xi_i(t|t-1) = \mathcal{Y}_i(t|t-1) - \hat{y}(t|t-1)$$
$$R_{\xi\xi}(t|t-1) = \sum_{i=0}^{2N_x} W_i \xi_i(t|t-1)\xi_i'(t|t-1) + R_{vv}(t)$$

8. Estimate the predicted cross-covariance as

$$R_{\tilde{x}\xi}(t|t-1) = \sum_{i=0}^{2N_x} W_i \tilde{\mathcal{X}}_i(t|t-1)\xi_i'(t|t-1)$$

Clearly, with these vectors and matrices available the corresponding gain and correction equations follow immediately as

$$\mathcal{K}(t) = R_{\tilde{x}\xi}(t|t-1)R_{\xi\xi}^{-1}(t|t-1)$$
$$e(t) = y(t) - \hat{y}(t|t-1)$$
$$\hat{x}(t|t) = \hat{x}(t|t-1) + \mathcal{K}(t)e(t)$$
$$\tilde{P}(t|t) = \tilde{P}(t|t-1) - \mathcal{K}(t)R_{\xi\xi}(t|t-1)\mathcal{K}(t)$$

We note in passing that there are *no* jacobians calculated and the nonlinear models are employed directly to transform the $\sigma$-points to the new space. Also in the original problem definition (see Section 6.1) both process and noise sources, $(w, v)$ were assumed *additive*. The *UT* enables us to "generalize" the noise terms to also be injected in a nonlinear manner (e.g., multiplication). Thus the noise is not treated separately but can be embedded into the problem by defining an augmented state vector, say $\overline{x}(t) = [x(t) \quad w(t-1) \quad v(t)]'$. We chose to ignore the general case to keep the development of the *UMBP* simple. For more details of the general process, see [28], [29], [30]. We summarize the *UMBP* algorithm in Table 6.5.

Before we conclude this discussion, let us apply the *UMBP* to the nonlinear trajectory estimation problem and compare its performance to the other nonlinear processors discussed previously.

***Example 6.6*** We revisit the nonlinear trajectory estimation problem of the previous examples with the dynamics specified by the discrete nonlinear process given by

$$x(t) = (1 - 0.05\triangle T)x(t-1) + 0.04x^2(t-1) + w(t-1)$$

and corresponding measurement model

$$y(t) = x^2(t) + x^3(t) + v(t)$$

**Table 6.5. Discrete Unscented *MBP* (Kalman Filter) Algorithm**

$\sigma$-Points and weights

$$\mathcal{X}_o = \hat{x}(t-1|t-1), \quad W_o = \frac{\kappa}{(N_x+\kappa)}$$

$$\mathcal{X}_i = \hat{x}(t-1|t-1) + \left(\sqrt{(N_x+\kappa)\,(\tilde{P}(t-1|t-1) + R_{ww}(t-1))}\right)_i,$$

$$W_i = \frac{1}{2(N_x+\kappa)}$$

$$\mathcal{X}_{i+N_x} = \hat{x}(t-1|t-1) - \left(\sqrt{(N_x+\kappa)\,(\tilde{P}(t-1|t-1) + R_{ww}(t-1))}\right)_i,$$

$$W_{i+N_x} = \frac{1}{2(N_x+\kappa)} \qquad (\sigma\text{-Points})$$

Prediction

$$\mathcal{X}_i(t|t-1) = \mathbf{a}[\mathcal{X}_i(t-1|t-1)] + \mathbf{b}[u(t-1)] \qquad \text{(Unscented transform)}$$

$$\hat{x}(t|t-1) = \sum_{i=0}^{2N_x} W_i \mathcal{X}_i(t|t-1) \qquad \text{(State prediction)}$$

$$\tilde{\mathcal{X}}_i(t|t-1) = \mathcal{X}_i(t|t-1) - \hat{x}(t|t-1) \qquad \text{(State residual)}$$

$$\tilde{P}(t|t-1) = \sum_{i=0}^{2N_x} W_i \tilde{\mathcal{X}}_i(t|t-1)\tilde{\mathcal{X}}_i'(t|t-1) \qquad \text{(Covariance prediction)}$$

Residual

$$\hat{\mathcal{X}}_i(t|t-1) = \left[\mathcal{X}_i(t|t-1),\ \mathcal{X}_i(t|t-1) + \kappa\sqrt{R_{ww}(t-1)}, \mathcal{X}_i(t|t-1) - \kappa\sqrt{R_{ww}(t-1)}\right] \qquad (\sigma\text{-Points})$$

$$\mathcal{Y}_i(t|t-1) = \mathbf{c}\left[\hat{\mathcal{X}}_i(t|t-1)\right] \qquad \text{(Measurement transform)}$$

$$\hat{y}(t|t-1) = \sum_{i=0}^{2N_x} W_i \mathcal{Y}_i(t|t-1) \qquad \text{(Predicted measurement)}$$

$$\xi_i(t|t-1) = \mathcal{Y}_i(t|t-1) - \hat{y}(t|t-1) \qquad \text{(Predicted residual)}$$

$$R_{\xi\xi}(t|t-1) = \sum_{i=0}^{2N_x} W_i \xi_i(t|t-1)\xi_i'(t|t-1) + R_{vv}(t) \qquad \text{(Residual covariance)}$$

**Table 6.5. (*continued*)**

Gain

$$R_{\tilde{x}\xi}(t|t-1) = \sum_{i=0}^{2N_x} W_i \tilde{\mathcal{X}}_i(t|t-1)\xi_i'(t|t-1) \qquad \text{(Cross-covariance)}$$

$$\mathcal{K}(t) = R_{\tilde{x}\xi}(t|t-1)R_{\xi\xi}^{-1}(t|t-1) \qquad \text{(Gain)}$$

Correction

$$e(t) = y(t) - \hat{y}(t|t-1) \qquad \text{(Innovation)}$$

$$\hat{x}(t|t) = \hat{x}(t|t-1) + \mathcal{K}(t)e(t) \qquad \text{(State correction)}$$

$$\tilde{P}(t|t) = \tilde{P}(t|t-1) - \mathcal{K}(t)R_{\xi\xi}(t|t-1)\mathcal{K}'(t) \qquad \text{(Covariance correction)}$$

Initial conditions

$$\hat{x}(0|0) \quad \tilde{P}(0|0)$$

Here recall that $v(t) \sim \mathcal{N}(0, 0.09)$, $x(0) = 2.0$, $P(0) = 0.01$, $\triangle T = 0.01$ s, and $R_{ww} = 0$. The simulated measurement is shown in Figure 6.2*c*. The *UMBP* and *XMBP* and *IX-MBP* (3 iterations) were applied to this problem. We used the square-root implementations of the *XMBP* and *IX-MBP* in *SSPACK_PC* [8] and compared them to the square-root version of the *UMBP* in *REBEL* [34]. The results are shown in Figure 6.8 where we see the corresponding trajectory (state) estimates in 6.8*a* and the "filtered" measurements in 6.8*b*. From the figure it appears that all of the estimates are quite reasonable with the *UMBP* estimate (thick solid line) converging most rapidly to the true trajectory (dashed line). The *XMBP* (thick dotted line) appears slightly biased while the *IX-MBP* (thin dotted line) converges rapidly, but then wanders slightly from the truth. The measurements also indicate the similar performance. The zero-mean/whiteness tests confirm these observations. The *XMBP* and *IX-MBP* perform similarly with respective zero-mean/whiteness values of ($1.04 \times 10^{-1} < 1.73 \times 10^{-1}$/1% out) and ($3.85 \times 10^{-2} < 1.73 \times 10^{-1}$/0% out), while the *UMBP* is certainly comparable at ($5.63 \times 10^{-2} < 1.73 \times 10^{-1}$/0% out). This completes the example. We summarize the model-based approach as follows:

| | |
|---|---|
| Criterion: | $J = \text{trace } \tilde{P}(t\|t)$ |
| Models: | |
| Signal: | $x(t) = (1 - 0.05\triangle T)x(t-1) + 0.04x^2(t-1) + w(t-1)$ |
| Measurement: | $y(t) = x^2(t) + x^3(t) + v(t)$ |
| Noise: | $w \sim \mathcal{N}(0, 0)$ and $v \sim \mathcal{N}(0, 0.09)$ |
| Initial conditions: | $\hat{x}(0\|0) = 2.3$ and $\tilde{P}(0\|0) = 0.01$ |

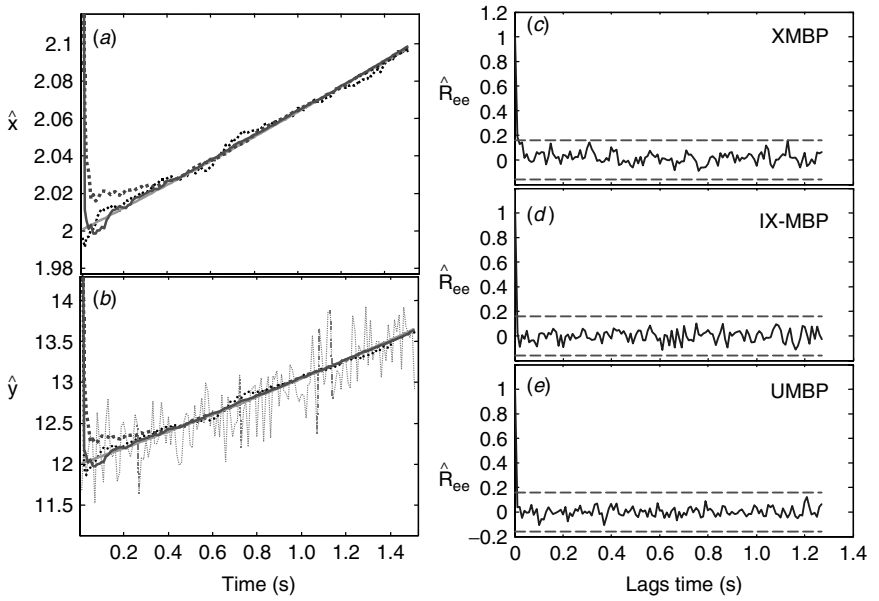



**Figure 6.8.** Nonlinear Trajectory Estimation. (*a*) Trajectory (state) estimates using the *XMBP* (thick dotted), *IX-MBP* (thin dotted), and *UMBP* (thick solid). (*b*) Filtered measurement estimates using the *XMBP* (thick dotted), *IX-MBP* (thin dotted), and *UMBP* (thick solid). (*c*) Zero-mean/whiteness tests for *XMBP* ($1.04 \times 10^{-1} < 1.73 \times 10^{-1}$/1% out). (*d*) Zero-mean/whiteness tests for *IX-MBP* ($3.85 \times 10^{-2} < 1.73 \times 10^{-1}$/0% out). (e) Zero-mean/whiteness tests for *UMBP* ($5.63 \times 10^{-2} < 1.73 \times 10^{-1}$/0% out).

$$\text{Algorithm:} \qquad \hat{x}(t|t) = \hat{x}(t|t-1) + \mathcal{K}(t)e(t)$$
$$\text{Quality:} \qquad \tilde{P}(t|t) = \tilde{P}(t|t-1) - \mathcal{K}(t)R_{\xi\xi}(t|t-1)\mathcal{K}'(t)$$

Next we consider the application of the nonlinear processors to the tracking problem.

## 6.5 CASE STUDY: 2D-TRACKING PROBLEM

In this section we investigate the design of nonlinear *MBP* to solve a two-dimensional (2D) tracking problem. The hypothetical scenario discussed will demonstrate the applicability of these processors to solve such a problem. It could easily be extrapolated to many other problems (air traffic control operations, position estimation, etc.) and develops the "basic" thinking behind constructing such a problem and solution.

In contrast to the "bearings-only" problem discussed earlier, let us investigate the tracking of a large tanker entering a busy harbor with a prescribed navigation path. In this case the pilot of the vessel must adhere strictly to the path that has been filed with the harbor master (controller). Here we assume that the ship has a transponder

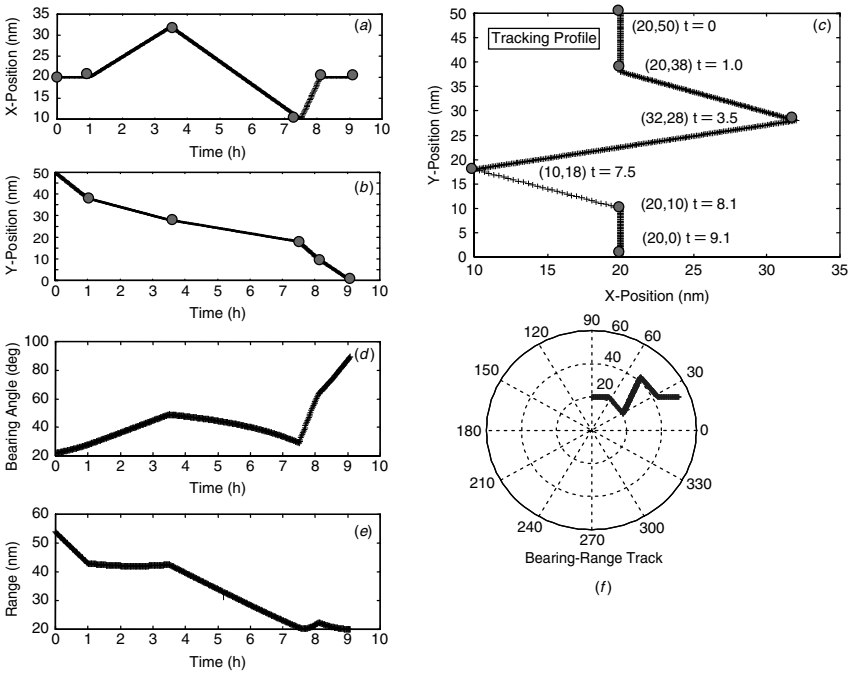



**Figure 6.9.** Tanker ground track geometry for the harbor docking application: (*a*) Instantaneous *X*-position (nm). (*b*) Instantaneous *Y*-position (nm). (*c*) Filed XY-path (nm). (*d*) Instantaneous bearing (deg). (*e*) Instantaneous range (nm). (*f*) Polar bearing-range track from sensor measurement.

frequently signaling accurate information about its current position. The objective is to safely dock the tanker without any "traffic" incidents. We observe that the ship's path should track the prescribed trajectory (cartesian coordinates) shown in Figure 6.9 with corresponding instantaneous $XY$-positions (versus time) shown as well.

Our fictitious measurement instrument (e.g., low ground clutter phased array radar or a sattelite communications receiver) is assumed to instantly report on the tanker position in bearing and range with high accuracy. That is, the measurements are given by

$$\Theta(t) = \arctan\left(\frac{Y(t)}{X(t)}\right) \quad \text{and} \quad R(t) = \sqrt{X^2(t) + Y^2(t)}$$

We use the usual state-space formulation for a constant velocity model (see Section 6.2) with state vector defined in terms of the physical variables as $x'(t) := [X(t)\ \ Y(t)\ \ V_x(t)\ \ V_y(t)]$ along with the incremental velocity input as $u' := [-\triangle V_{x_o} - \triangle V_{y_o}]$.

By this information (as before) the entire system can be represented as a Gauss-Markov model with the noise sources representing uncertainties in the states and

measurements. Thus we have the equations of motion given by

$$x(t) = \begin{bmatrix} 1 & 0 & \triangle T & 0 \\ 0 & 1 & 0 & \triangle T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x(t-1) + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -\triangle V_{x_o}(t-1) \\ \\ -\triangle V_{y_o}(t-1) \end{bmatrix} + w(t-1) \tag{6.85}$$

with the corresponding measurement model given by

$$y(t) = \begin{bmatrix} \arctan \dfrac{x_2(t)}{x_1(t)} \\ \sqrt{x_1^2(t) + x_2^2(t)} \end{bmatrix} + v(t)$$

for $w \sim \mathcal{N}(0, R_{ww})$ and $v \sim \mathcal{N}(0, R_{vv})$.

The *SSPACK_PC* software [8] was used to simulate this Gauss-Markov system for the tanker path. In this scenario we assume that the instrument is capable of making measurements every $\triangle T = 0.02$ h with a bearing precision of $\pm 0.02$ degree and range precision of $\pm 0.005$ nm (or equivalently $R_{vv} = \text{diag}[4 \times 10^{-4} \quad 1 \times 10^{-4}]$. The model uncertainty was represented by $R_{ww} = \text{diag}(1 \times 10^{-6})$. An impulse-incremental step change in velocity, such as $V_y$ going from $-12\ k$ to $-4\ k$, is an incremental change of $+8\ k$ corresponding to $\triangle V_{y_o} = [8 \quad 1.5 \quad -10.83 \quad 3.33]$ knots and $\triangle V_{x_o} = [0 \quad 4.8 \quad -10.3 \quad 22.17]$ knots. These impulses (changes) occur at time fiducials of $t = [0 \quad 1 \quad 3.5 \quad 7.5 \quad 8.1 \quad 9.1]$ h corresponding to the filed harbor path. Note that the velocity changes are impulses of height $(\triangle V_x, \triangle V_y)$ corresponding to a known deterministic input, $u(t)$. These changes relate physically to instantaneous direction changes of the tanker and create the path change in the constant velocity model.

The simulated bearing measurements are generated using the initial conditions $x'(0) := [\ 20 \text{ nm} \quad 50 \text{ nm} \quad 0 \text{ k} \quad -12 \text{ k}\ ]$ and $R_{ww} = \text{diag}\ (1 \times 10^{-6})$ with the corresponding initial covariance given by $\tilde{P}(0) = 1 \times 10^{-6}$. The *XMBP* algorithm of Table 6.3 is implemented using these model parameters and the following jacobian matrices derived from the Gauss-Markov model above:

$$A[x] = A \quad \text{and} \quad C[x] = \begin{bmatrix} \dfrac{x_2(t)}{R^2(t)} & \dfrac{-x_1(t)}{R^2(t)} & 0 & 0 \\ \dfrac{x_1(t)}{R(t)} & \dfrac{x_2(t)}{R(t)} & 0 & 0 \end{bmatrix}$$

The *XMBP*, *IX-MBP*, and *LZ-MBP* were run under the constraint that all of the a priori information for the tanker harbor path is "known." Each of the processors performed almost identically with a typical single realization output shown for the *XMBP* in Figure 6.10. In 6.10*a* and 6.10*b* we observe the estimated states X ($\sim 0\%$ out), Y ($\sim 0\%$ out), $V_x$ ($\sim 0\%$ out), and $V_y$ ($\sim 3\%$ out). Note that the velocities are piecewise constant functions with step changes corresponding to the impulsive incremental velocities. The filtered measurements: bearing ($\sim 1\%$

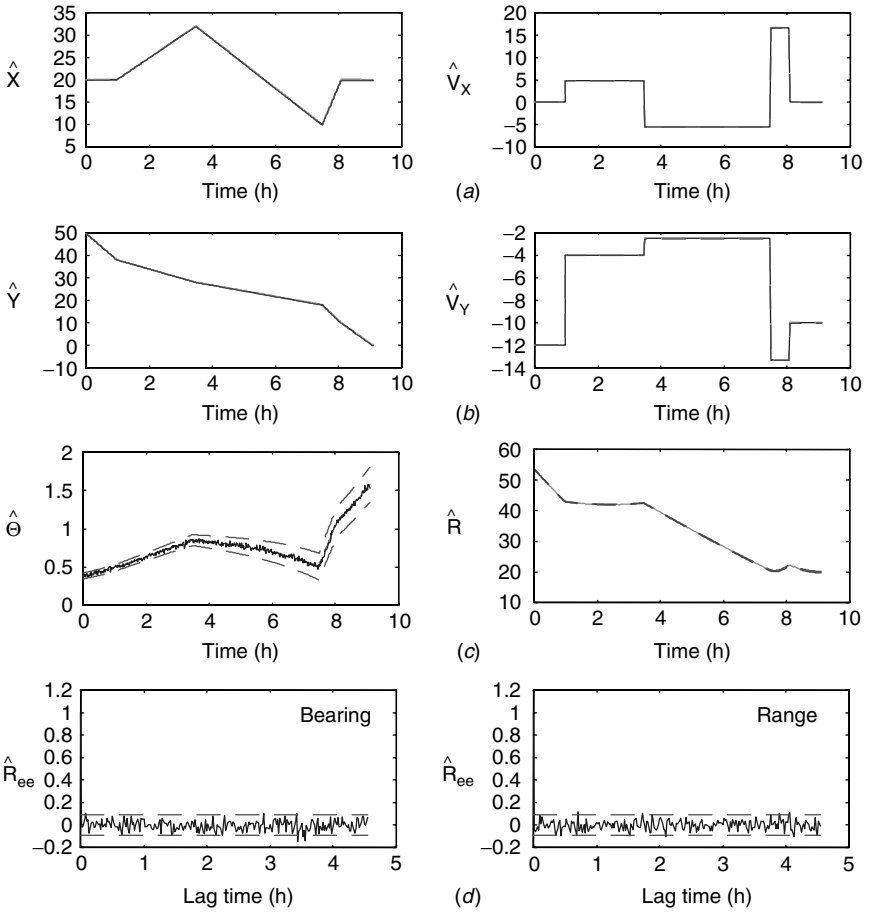



**Figure 6.10.** XMBP design for harbor docking problem (input known). (*a*) *X*-position and velocity estimates with bounds (0% out). (*b*) *Y*-position and velocity estimates with bounds (0% and 3% out). (*c*) Bearing and range estimates with bounds (1% and 2% out). (*d*) Innovations zero-mean/whiteness tests for bearing ($6 \times 10^{-4} < 26 \times 10^{-4}$ and 3% out) and range ($2 \times 10^{-4} < 42 \times 10^{-4}$ and 5% out).

out) and range ($\sim 2\%$ out) are shown in Figure 6.10*c* with the resulting innovations zero-mean/whiteness tests depicted in Figure 6.11*d*. The processor is clearly tuned with bearing and range innovations zero-mean and white ($6 \times 10^{-4} < 26 \times 10^{-4}/3\%$ out) and ($2 \times 10^{-4} < 42 \times 10^{-4}/5\%$ out), respectively. This result is not unexpected, since all of the a priori information is given including the precise incremental velocity input, $u(t)$. An ensemble of 101 realizations of the estimator were run by generating random initial condition estimates from the gaussian assumption. The 101 realization ensemble averaged estimates closely follow the results shown in the figure with the zero-mean/whiteness tests ($2 \times 10^{-4} < 25 \times 10^{-4}/4\%$ out), ($2 \times 10^{-4} < 15 \times 10^{-4}/7\%$ out) slightly worse.

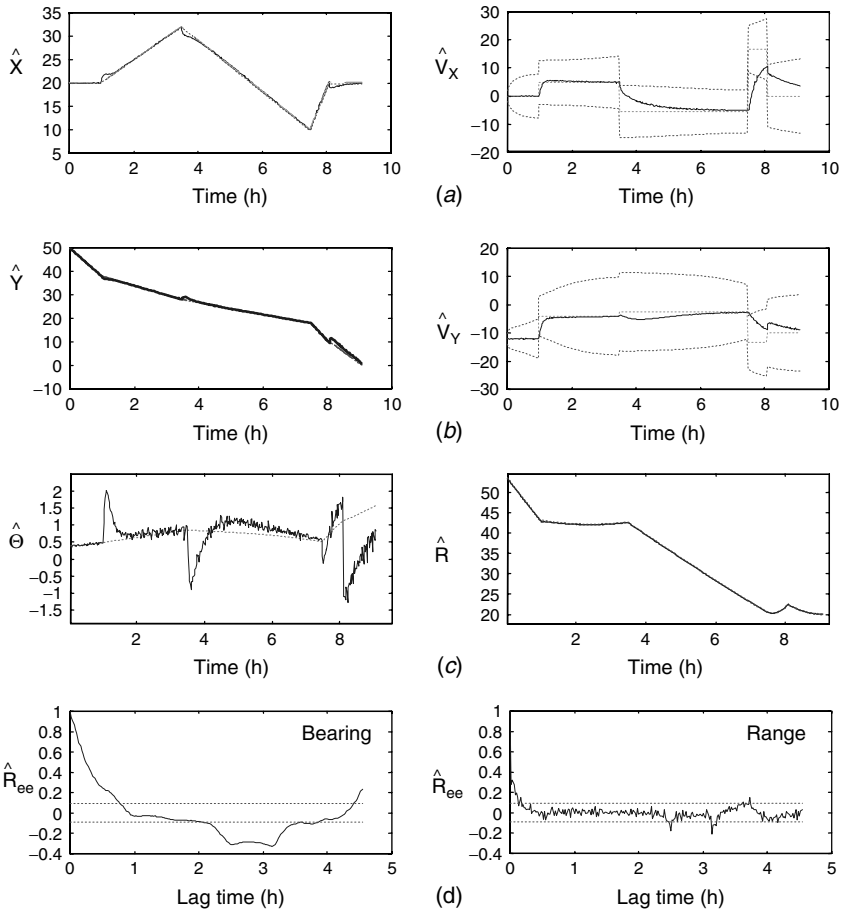



**Figure 6.11.** LZMBP design for harbor docking problem (input known). (*a*) *X*-position and velocity estimates with bounds (68% and 4% out). (*b*) *Y*-position and velocity estimates with bounds (49% and 1% out). (*c*) Bearing and range estimates with bounds (0% and 3% out). (*d*) Innovations zero-mean/whiteness tests for bearing ($75 \times 10^{-3} < 81 \times 10^{-3}$ and 59% out) and range ($2 \times 10^{-3} < 4 \times 10^{-3}$ and 8% out).

Next we investigate the usual case where all of the information is known a priori *except* the impulsive incremental velocity changes represented by the deterministic input. Note that without the input, the processor cannot respond instantaneously to the velocity changes and therefore will lag (in time) behind in predicting the tanker path. The solution to this problem requires a joint estimation of the states *and* the unknown input, which is really a solution to the deconvolution problem [31]. It is also a problem that is ill-conditioned especially, since $u(t)$ is impulsive.

In any case, we ran the nonlinear *MBP* algorithms over the simulated data and the best results were obtained using the *LZ-MBP*. This is expected, since we used the exact state reference trajectories, but not the input. Note that the other

nonlinear *MBP* have no knowledge of this trajectory inhibiting their performance in this problem. The results are shown in Figure 6.11 where we observe the state estimates as before. We note that these position estimates (65% out, 49% out) appear reasonable, primarily because of the reference trajectories. The *LZ-MBP* is able to compensate for the unknown impulsive input with a slight lag as shown at each of the fiducials. The velocity estimates (4% out, 1% out) are actually low-pass versions of the true velocities caused by the slower *LZ-MBP* response even with the exact step changes available. These lags are more vividly shown in the bearing estimate of Figure 6.11*c*, which shows the processor has great difficulty with the instantaneous velocity changes in bearing (0% out). The range (0% out) appears insensitive to this lack of knowledge primarily because the *XY*-position estimates are good and do not have step changes like the velocity for the *LZ-MBP* to track. Both processors are not optimal and the innovations sequences are zero-mean but *not* white ($75 \times 10^{-3} < 81 \times 10^{-3}$/59% out), ($2 \times 10^{-3} < 4 \times 10^{-3}$/8% out).

For the final case of this study, we investigate using the nonlinear *MBP* in the classical bearings-only scenario where we deny the processors not only knowledge of deterministic input (incremental velocity changes), but also the range measurements. We ran all of the processors over the synthesized data with the most reasonable results produced by the *IX-MBP* and *LZ-MBP* as shown in Figure 6.12. In Figure 6.12*a* we see the estimated bearing and corresponding innovations sequence produced by the *IX-MBP*. Here 10-iterations per time step were used along with a process covariance matrix, $R_{ww} = I$, to "tune" the processor. Note that because of the large values of process noise covariance, the bandwidth of the *IX-MBP* is increased enabling it to respond more rapidly to bearing changes at the cost of noisier estimates. The processor is "tuned" since the zero-mean and whiteness tests ($9 \times 10^{-3} < 15 \times 10^{-3}$/3.5% out) satisfy the optimality criteria as shown in Figure 6.12*c*.

The *LZ-MBP* also produces reasonable results as depicted in Figure 6.12*b*. We see that the bearing estimate is reasonable and not as noisy as that of the *IX-MBP*, but the processor is not able to respond as rapidly to the bearing change at 3.5 hours producing a *bias* in the estimate. The corresponding innovations sequence indicates this as well. The biased result is demonstrated by the zero-mean/whiteness test results shown in 6.12*c* with ($21 \times 10^{-3} > 7 \times 10^{-3}$/54% out). This completes the case study.

We summarize the results as follows:

Criterion: $J = \text{trace } \tilde{P}(t|t)$

Models:

Signal: $x(t) = Ax(t-1) + bu(t-1) + w(t-1)$

Measurement:

$$y(t) = \begin{bmatrix} \arctan \dfrac{x_2(t)}{x_1(t)} \\ \sqrt{x_1^2(t) + x_2^2(t)} \end{bmatrix} + v(t)$$

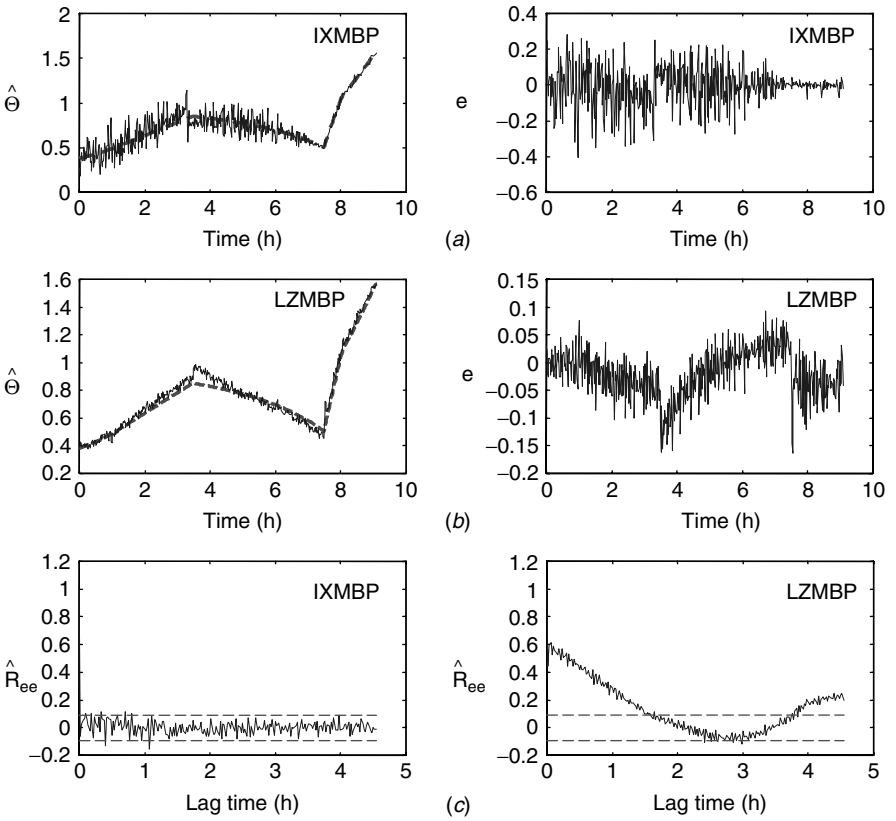



**Figure 6.12.** IXMBP/LZMBP design for ``bearings-only'' harbor docking problem (input unknown). (*a*) IXMBP bearing estimates and innovations. (*b*) LZMBP bearing estimates and innovations. (*c*) IXMBP/LZMBP innovations zero-mean/whiteness tests ($9 \times 10^{-3} < 15 \times 10^{-3}$ and 4% out) and ($21 \times 10^{-3} > 7 \times 10^{-3}$ and 54% out), respectively.

Noise: $w \sim \mathcal{N}(0, \text{diag}[10^{-6}])$ and $v \sim \mathcal{N}(0, \text{diag}[4 \times 10^{-4} \ 1 \times 10^{-4})$

Initial conditions: $x'(0) := [20 \text{ nm} \ \ 50 \text{ nm} \ \ 0 \text{ k} \ \ -12 \text{ k}]$ and $\tilde{P}(0|0) = \text{diag}\ [10^{-6}]$

Algorithm: $\hat{x}(t|t) = \hat{x}(t|t-1) + K(t)e(t)$

Quality: $\tilde{P}(t|t) = (I - K(t)C[x])\,\tilde{P}(t|t-1)$

It is clear from this study that nonlinear processors can be "tuned" to give reasonable results especially when they are provided with accurate a priori information. If the a priori information is provided in terms of prescribed reference trajectories as in this hypothetical case study, then the *LZ-MBP* appears to provide superior performance. But in the real-world tracking problem (as discussed in Section 6.2) when this information on the target is not available, then the *XMBP* and *IX-MBP* can be tuned to give reasonable results.

There are many variants possible for these processors to improve their performance whether it be in the form of improved coordinate systems [17], [18] or in the form of a set of models each with its own independent processor [10]. One might also consider using estimator/smoothers as in the seismic case [14] because of the unknown impulsive input. This continues to be a challenging problem. The interested reader should consult [10] and the references cited therein for more details.

This completes the chapter on nonlinear state-space-based *MBP*. Next we consider the performance of adaptive schemes for *MBP*.

## 6.6 SUMMARY

In this chapter we introduced the concepts of nonlinear model-based signal processing using state-space models. After developing the idea of linearizing a nonlinear state-space system, we developed the linearized model-based processor (*LZ-MBP*). It was shown that the resulting processor provides an approximate solution (time-varying) to the nonlinear state estimation problem. We then developed the extended model-based processor (*XMBP*) or equivalently the extended Kalman filter (*EKF*), as a special case of the *LZ-MBP* linearizing about the most currently available estimate. Next we investigated a further enhancement of the *XMBP* by introducing a local iteration of the nonlinear measurement system using a Newton-Rhapson iteration method. Here the processor is called the iterated-extended model-based processor (*IX-MBP*) and is shown to produce improved estimates at a small computational cost in most cases. Finally we examined one of the modern nonlinear processing schemes, the unscented *MBP* ([32], [33]). Examples were developed throughout to demonstrate the concepts ended in a hypothetical investigation based on tracking a tanker entering a busy harbor. We summarized the results by applying some of the processors to this case study implementing a 2D-tracking filter.

## *MATLAB* NOTES

*SSPACK_PC* [8] is a third-party toolbox in *MATLAB* that can be used to design model-based signal processors. This package incorporates the major *nonlinear MBP* algorithms discussed in this chapter—all implemented in the *UD*-factorized form [16] for stable and efficient calculations. It performs the discrete approximate Gauss-Markov simulations using (**SSNSIM**) and both extended (*XMBP*) and iterated-extended (*IX-MBP*) processors using (**SSNEST**). The linearized model-based processor (*LZ-MBP*) is also implemented (**SSLZEST**). Ensemble operations are seamlessly embodied within the GUI-driven framework, where it is quite efficient to perform multiple design runs and compare results. Of course, the heart of the package is the command or GUI-driven postprocessor (**SSPOST**) that is used to analyze and display the results of the simulations and processing. (see **http://www.techni-soft.net** for more details).

*REBEL* is a recursive Bayesian estimation package in *MATLAB* available on the Web that performs similar operations including the new statistical-based unscented algorithms including the *UMBP* including the unscented transformations [34]. It also has included the new particle filter designs as discussed in [33] (see **http://choosh.ece.ogi.edu/rebel** for more details).

## REFERENCES


1. A. Jazwinski, *Stochastic Processes and Filtering Theory*, New York: Academic Press, 1970.
2. A. P. Sage and J. L. Melsa, *Estimation Theory with Applications to Communications and Control*, New York: McGraw-Hill, 1971.
3. A. Gelb, *Applied Optimal Estimation*, Cambridge: MIT Press, 1975.
4. B. D. Anderson and J. B. Moore, *Optimal Filtering*, Englewood Cliffs, NJ: Prentice-Hall, 1979.
5. P. Maybeck, *Stochastic Models, Estimation, and Control*, New York: Academic Press, 1979.
6. M. S. Grewal and A. P. Andrews, *Kalman Filtering: Theory and Practice*, Englewood Cliffs, NJ: Prentice-Hall, 1993.
7. J. M. Mendel, *Lessons in Estimation Theory for Signal Processing, Communications and Control*, Englewood Cliffs, NJ: Prentice-Hall, 1995.
8. J. V. Candy and P. M. Candy, "SSPACK_PC: A model-based signal processing package on personal computers," *DSP Applications*, **2** (3), 33–42 1993 (see **http://www.technisoft.net** for more details).
9. J. V. Candy, *Signal Processing: The Model-Based Approach*, New York: McGraw-Hill, 1986.
10. Y. Bar-Shalom and X. Li, *Estimation and Tracking: Principles, Techniques, and Software*, Norwood, MA: Artech House, 1993.
11. B. Bell and F. Cathey, "The iterated Kalman filter update as a Gauss-Newton method," *IEEE Trans. Autom. Contr.*, **38** (2), 294–297 1993.
12. J. V. Candy and R. B. Rozsa, "Safeguards for a plutonium-nitrate concentrator—An applied estimation approach,"*Automatica*, **16**, 615–627 1980.
13. J. V. Candy and E. J. Sullivan, "Ocean acoustic signal processing: A model-based approach," *J. Acoust. Soc. Am.*, **92**, 3185–3201 1992.
14. J. Mendel, J. Kormylo, J. Lee, and F. Ashirafi, "A novel approach to seismic signal processing and modeling," *Geophysics*, **46**, 1398–1414 1981.
15. H. W. Sorenson, ed., "Special issue on applications of Kalman filtering," *IEEE Trans. Auto. Contr.*, **28** (3), 253–427 1983.
16. G. J. Bierman, *Factorization Methods of Discrete Sequential Estimation*, New York: Academic Press, 1977.
17. V. J. Aidala and S. M. Hammel, "Utilization of modified polar-coordinates for bearings-only tracking," *IEEE Trans. Autom. Contr.*, **28**, 283–294 1983.
18. V. J. Aidala, "Kalman filter behavior in bearings-only velocity and position estimation," *IEEE Trans. Aerosp. Electron. Syst.*, **15**, 29–39 1979.

19. L. J. Ljung, *System Identification: Theory for the User*, Englewood Cliffs, NJ: Prentice-Hall, 1987.
20. M. Athans, R. Wishner, and A. Berolini, "Suboptimal state estimation for continuous-time nonlinear systems from discrete noisy measurements," *IEEE Trans. Autom. Contr.*, **13** (4), 504–514 1968.
21. S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte, "A new approach for filtering in nonlinear systems," *Proc. Am. Contr. Conf.*, 1995.
22. S. J. Julier and J. K. Uhlmann, *A General Method for Approximating Nonlinear Transformations of Probability Distributions*, Univ. of Oxford Report, http://www.robots.ox.ac.uk/∼siju 1996.
23. S. J. Julier, J. K. Uhlmann and H. F. Durrant-Whyte, "A new method for the nonlinear transformation of means and covariances in filters and estimators," *IEEE Trans. Autom. Contr.*, **45** (3), 477–482 2000.
24. A. F. Smith and A. E. Gelfand,"Bayesian statistics without tears: A sampling-resampling perspective," *Am. Statistic.*, **46** (2), 84–88 1992.
25. S. J. Julier and J. K. Uhlmann, "A new extension of the Kalman filter to nonlinear systems," *Proc. SPIE Conf.*, Orlando, FL, 1997.
26. A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, New York: McGraw-Hill, 1991.
27. S. J. Julier and J. K. Uhlmann, "A consistent, debiased method for converting between polar and cartesian coordinate systems," *Proc. SPIE Conf.*, 1997.
28. S. Haykin, *Kalman Filtering and Neural Networks*, New York: Wiley, 2001.
29. E. A. Wan and R. van der Merwe, "The unscented Kalman filter for nonlinear estimation," *Proc. IEEE Symp. Adaptive Sys. Sig. Proc., Comm. Contr.*, Lake Louise, Alberta, 2000.
30. R. van der Merwe, A. Doucet, N. de Freitas, and E. A. Wan, "The unscented particle filter," *Cambridge University Tech. Report*, CUED/F-INFENG-380, 2000.
31. J. V. Candy and J. E. Zicker, "Deconvolution of noisy transient signals: A Kalman filtering application," *LLNL Rep.*, UCID-87432; and *Proc. CDC Conf.*, Orlando, 1982.
32. P. Djuric, J. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. Bugallo, and J. Miguez, "Particle filtering," *IEEE Signal Proc. Mag.*, **20** (5), 19–38 2003.
33. S. Haykin and N. de Freitas, eds., "Sequential state estimation: From Kalman filters to particle filters," *Proc. IEEE*, **92** (3), 399–574 2004.
34. R. van der Merwe and E. A. Wan, *REBEL: Recursive Bayesian Estimation Library University of Oregon*, (see **http://choosh.ece.ogi.edu/rebel** for more details), 2002.

## PROBLEMS

**6.1** Develop the discrete linearized (perturbation) models for each of the following nonlinear systems ([7]):

- *Synchronous (unsteady) motor:* $\ddot{x}(t) + C\dot{x}(t) + p \sin x(t) = L(t)$
- *Duffing equation:* $\ddot{x}(t) + \alpha x(t) + \beta x^3(t) = F \cos \omega t)$
- *Van der Pol equation:* $\ddot{x}(t) + \epsilon \dot{x}(t) \left[1 - \dot{x}^2(t)\right] + x(t) = m(t)$
- *Hill equation:* $\ddot{x}(t) - \alpha x(t) + \beta p(t) x(t) = m(t)$

**(a)** Develop the *LZ-MBP*.
**(b)** Develop the *XMBP*.
**(c)** Develop the *IX-MBP*.
**(d)** Develop the *UMBP*.

**6.2** Suppose that we are given a continuous-time Gauss-Markov model characterized as by

$$\dot{x}(t) = A_c(t)x(t) + B_c(t)u(t) + W_c(t)w(t)$$

and discrete (sampled) measurement model such that $t \to t_k$. Then

$$y(t_k) = C(t_k)x(t_k) + v(t_k)$$

where the continuous process $w(t) \sim \mathcal{N}(0, R_{ww})$ and $v(t_k) \sim \mathcal{N}(0, R_{vv})$ with gaussian initial conditions.

**(a)** Determine the state mean $(m_x(t))$ and covariance $(P(t))$.
**(b)** Determine the measurement mean $(m_y(t_k))$ and covariance $(R_{yy}(t_k))$.
**(c)** Develop the relationship between the continuous and discrete Gauss-Markov models based on the solution of the continuous state equations and approximation using a first-order Taylor series for the state transition matrix, $\Phi(t, t_o)$, and the associated system matrices.
**(d)** Derive the continuous-discrete *MBP* using first difference approximations for derivatives and the discrete (sampled) system matrices derived above.

**6.3** Given a continuous-discrete Gauss-Markov model

$$\dot{x}(t) = \alpha x(t) + u(t) + w(t)$$
$$y(t_k) = \beta x(t_k) + v(t_k)$$

where $w(t)$ and $v(t_k)$ are zero-mean and white with respective covariances, $R_{ww}$ and $R_{vv}$, along with a piecewise constant input, u(t).

**(a)** Develop the continuous-discrete mean and covariance propagation models for this system.
**(b)** Suppose $w(t)$ is processed by a coloring filter that exponentially correlates it, $R_{ww}(\tau) = Ge^{-|\lambda|\tau}$. Develop the continuous-discrete Gauss-Markov model in this case.
**(c)** Sketch out the continuous-discrete predictor for this problem.

**6.4** Develop the continuous-discrete Gauss-Markov models for the following systems:

**(a)** Wiener process: $\dot{z}(t) = w(t)$; $z(0) = 0$, $w$ is zero-mean, white with $R_{ww}$.

**(b)** Random bias: $\dot{z}(t) = 0$; $z(0) = z_o$ where $z_o \sim \mathcal{N}(0, R_{z_o z_o})$.

**(c)** Random ramp: $\ddot{z}(t) = 0$; $\dot{z}(0) = z_1$; $z(0) = z_o$

**(d)** Random oscillation: $\ddot{z}(t) + \omega_o^2 z(t) = 0$; $\dot{z}(0) = z_1$; $z(0) = z_o$

**(e)** Random second order: $\ddot{z}(t) + 2\zeta\omega_n \dot{z} + \omega_n^2 z(t) = \omega_n^2 w(t)$; $\dot{z}(0) = z_1$; $z(0) = z_o$

**6.5** Develop the continuous-discrete Gauss-Markov model for correlated process noise, that is,

$$\dot{w}(t) = A_{cw} w(t) + B_{cw} u(t) + W_{cw} w^*(t) \quad \text{for} \quad w^* \sim \mathcal{N}(0, R_{w^* w^*})$$

**6.6** Suppose that we are given the following discrete system:

$$x(t) = -\omega^2 x(t-1) + \sin x(t-1) + \alpha u(t-1) + w(t-1)$$
$$y(t) = x(t) + v(t)$$

with $w$ and $v$ zero-mean, white gaussian with usual covariances, $R_{ww}$ and $R_{vv}$.

**(a)** Develop the *LZ-MBP* for this process.

**(b)** Develop the *XMBP* for this process.

**(c)** Develop the *IX-MBP* for this process.

**(d)** Develop the *UMBP* for this process.

**(e)** Suppose that the parameters $\omega$ and $\alpha$ are unknown develop the *XMBP* such that the parameters are jointly estimated along with the states. (*Hint*: Augment the states and parameters to create a new state vector.)

**6.7** Assume that we have the following nonlinear continuous-discrete Gauss-Markov model:

$$x(t) = f[x(t)] + g[u(t)] + w(t)$$
$$z(t_k) = h[x(t_k)] + v(t_k)$$

with $w$ and $v$ zero-mean, white gaussian with usual covariances, $Q$ and $R$.

**(a)** Develop the perturbation model for $\delta x(t) := x(t) - x^*(t)$ for $x^*(t)$ the given reference trajectory.

**(b)** Develop the *LZ-MBP* for this process.

**(c)** Choose $x^*(t) = \hat{x}(t)$ whenever $\hat{x}(t)$ is available during the recursion. Therefore develop the continuous-discrete *XMBP*.

**6.8** Suppose that we assume that the target of Example 6.69 is able to maneuver. That is, we assume that the target velocity satisfies a first order *AR* model given by

$$v_\tau(t) = -\alpha v_\tau(t-1) + w_\tau(t-1) \quad \text{for} \quad w \sim \mathcal{N}(0, R_{w_\tau w_\tau})$$

**(a)** Develop the cartesian tracking model for this process.

**(b)** Develop the corresponding *XMBP* assuming all parameters are known a priori.

**(c)** Develop the corresponding *XMBP* assuming $\alpha$ is unknown.

**6.9** Nonlinear processors (*LZ-MBP*, *XMBP*, *IX-MBP*, *UMBP*) can be used to develop neural networks used in many applications. Suppose that we model a generic neural network behavior by

$$x(t) = x(t-1) + w(t-1)$$
$$y(t) = c[x(t), u(t), \alpha(t)] + v(t)$$

where $x(t)$ is the network weights (parameters), $u(t)$ is the input or training sequence, $\alpha(t)$ is the node activators with $w$ and $v$ zero-mean, white gaussian with covariances, $R_{ww}$ and $R_{vv}$.

**(a)** Develop the *LZ-MBP* for this process.

**(b)** Develop the *XMBP* for this process.

**(c)** Develop the *IX-MBP* for this process.

**(d)** Develop the *UMBP* for this process.

**6.10** The Mackey-Glass time delay differential equation is given by

$$\dot{x}(t) = \frac{\alpha x(t-\tau)}{1 + x(t-\tau)^N} - \beta x(t) + w(t)$$
$$y(t) = x(t) + v(t)$$

where $\alpha$, $\beta$ are constants, $N$ is a positive integer with $w$ and $v$ zero-mean, white gaussian with covariances, $R_{ww}$ and $R_{vv}$. For the parameter set: $\alpha = 0.2$, $\beta = 0.1$, $\tau = 7$, and $N = 10$ with $x(0) = 1.2$ (see [28], [34])

**(a)** Develop the *LZ-MBP* for this process.

**(b)** Develop the *XMBP* for this process.

**(c)** Develop the *IX-MBP* for this process.

**(d)** Develop the *UMBP* for this process.

**6.11** Consider the problem of estimating a random signal from an AM modulator characterized by

$$s(t) = \sqrt{2P}a(t)\sin\ \omega_c t$$
$$r(t) = s(t) + v(t)$$

where $a(t)$ is assumed to be a gaussian random signal with power spectrum

$$S_{aa}(\omega) = \frac{2k_a P_a}{\omega^2 + k_a^2}$$

also assume that the processes are contaminated with the usual additive noise sources: $w$ and $v$ zero-mean, white gaussian with covariances, $R_{ww}$ and $R_{vv}$.

**(a)** Develop the continuous-time Gauss-Markov model for this process.

**(b)** Develop the corresponding discrete-time Gauss-Markov model for this process using first differences.

**(c)** Develop the *MBP*.

**(d)** Assume the carrier frequency, $\omega_c$ is unknown. Develop the *XMBP* for this process.

**(e)** Develop the *UMBP* for this process.

**6.12** Derive the XMBP (EKF) directly from the Bayesian viewpoint by assuming that the processes are approximately gaussian governing by the following distributions.

$$\begin{aligned}
&\Pr(y(t)|x(t)) &&: \mathcal{N}(c[x(t)], R_{vv}(t)) \\
&\Pr(x(t)|y(t-1)) &&: \mathcal{N}(\hat{x}(t|t-1), \tilde{P}(t|t-1)) \\
&\Pr(y(t)|y(t-1)) &&: \mathcal{N}(\hat{y}(t|t-1), R_{ee}(t))
\end{aligned}$$

That is, calculate the posterior distribution

$$\Pr(x(t)|Y(t))$$

and $\hat{X}_{\text{map}}(t)$.

# 7

# ADAPTIVE *AR, MA, ARMAX,* EXPONENTIAL MODEL-BASED PROCESSORS

## 7.1 INTRODUCTION

Adaptive methods of signal processing have evolved from various disciplines with their roots firmly embedded in numerical optimization theory. Adaptive techniques have been very popular in the such areas as signal processing ([1], [2]), array design [3], control [4], and system identification ([5], [6]). These techniques find application in a variety of problems when, in contrast to optimal estimation (see Section 3.4), signal characteristics are not known very well. Specifically, if the signal has unknown statistics, is nonstationary, or time-varying or the underlying phenomenology is unknown or partially known, then adaptive processors offer a potential solution. In fact, if the signal is stationary but the statistics are unknown, the adaptive processor will in fact converge to the optimal estimator. The main problem in adaptive processing is to find an algorithm to adjust parameters (statistics, gain, coefficients, etc.) where incomplete knowledge of the signal characteristics exist.

The adaptive approach to signal processing is summarized in Figure 7.1. Once the appropriate model set is selected, an adaption algorithm must be developed based on a particular criterion to produce the desired signal estimate.

We summarize the *adaptive signal processing* method by the following steps:

1. Select a model set (*ARMAX*, lattice, state-space, etc.)
2. Select an adaption algorithm (gradient, Newton, etc.) to iteratively estimate model parameters as

$$\underline{\hat{\Theta}}(i+1) = \underline{\hat{\Theta}}(i) + \triangle_i \underline{d}_i$$

for the $(i+1)$th iteration of step-size $\triangle_i$ with direction vector $\underline{d}_i$.

*Model-Based Signal Processing*, by James V. Candy

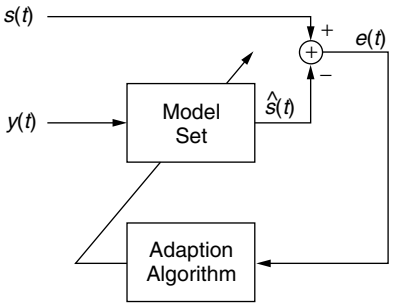



**Figure 7.1.** Adaptive signal processing method.

3. Construct the processor to adaptively estimate the signal, that is,

$$\hat{s}\left(t, \underline{\hat{\Theta}}(i)\right) = f(\underline{\hat{\Theta}}(i), y(t))$$

Thus with this structure in mind, we consider various adaption schemes.

## 7.2 ADAPTION ALGORITHMS

In this section we briefly discuss some of the popular numerical optimization techniques that have been applied to the design of adaptive processors. We make no attempt to develop the underlying theory which is quite extensive, but refer the reader to the following references for details [7], [8], [9], [10].

Most popular adaptive algorithms are based on iteratively updating the $N_\Theta$-parameter vector, $\underline{\Theta}(i)$, that is,

$$\underline{\Theta}(i+1) = \underline{\Theta}(i) + \triangle\underline{\Theta}(i) \tag{7.1}$$

where $\triangle\underline{\Theta}$ is the vector correction added to the previous iterate to obtain the update. The *gradient* algorithm approach is to adjust the parameters iteratively to minimize the criterion function by descending along this performance surface to reach the minimum. The iteration of Eq. (7.1) is adjusted so that the criterion is decreased at each step, that is,

$$\mathcal{J}(\underline{\Theta}(i+1)) \leq \mathcal{J}(\underline{\Theta}(i)) \tag{7.2}$$

must decrease until a minimum is achieved. Suppose that we expand the criterion in a Taylor series about $\underline{\Theta}(i)$

$$\mathcal{J}(\underline{\Theta}(i+1)) = \mathcal{J}(\underline{\Theta}(i)) + (\underline{\Theta}(i+1) - \underline{\Theta}(i))'\nabla_\Theta \mathcal{J}(\underline{\Theta}(i)) + \text{H.O.T.}$$

Assuming $\triangle\underline{\Theta}(i)$ is small, we can neglect the H.O.T., and we have

$$\mathcal{J}(\underline{\Theta}(i+1)) \approx \mathcal{J}(\underline{\Theta}(i)) + \triangle\underline{\Theta}'(i)\nabla_\Theta \mathcal{J}(\underline{\Theta}(i)) \leq \mathcal{J}(\underline{\Theta}(i)) \tag{7.3}$$

Suppose that we choose the negative *gradient direction* with step-size $\triangle_i$, then the gradient parameter iteration becomes

$$\underline{\Theta}(i+1) = \underline{\Theta}(i) + \triangle\underline{\Theta}(i) = \underline{\Theta}(i) - \triangle_i \nabla_\Theta \mathcal{J}(\underline{\Theta}(i)) \tag{7.4}$$

where the correction term is identified as

$$\triangle\underline{\Theta}(i) = -\triangle_i \underline{d}_i = -\triangle_i \nabla_\Theta \mathcal{J}(\underline{\Theta}(i)) \tag{7.5}$$

This choice guarantees that the inequality of Eq. (7.2) holds, since

$$\mathcal{J}(\underline{\Theta}(i+1)) \approx \mathcal{J}(\underline{\Theta}(i)) - \left(\triangle_i \nabla_\Theta \mathcal{J}(\underline{\Theta}(i)) \nabla_\Theta \mathcal{J}(\underline{\Theta}(i))\right) \leq \mathcal{J}(\underline{\Theta}(i))$$

Alternatively, subtracting $\mathcal{J}(\underline{\Theta}(i))$ from both sides, we have

$$-\triangle_i \left(\nabla_\Theta \mathcal{J}(\underline{\Theta}(i))\right)^2 \leq 0$$

since the step-size $\triangle_i$ is a positive scalar chosen in a suitable way. However, the gradient method becomes inefficient, when the parameter estimates approach the optimum. We illustrate the gradient technique in Figure 7.2.

If the direction vector is selected as an inverse Hessian matrix times the gradient, then the *Newton direction* occurs as

$$\triangle\underline{\Theta}(i) = -\triangle_i \underline{d}_i = -\triangle_i \left[\nabla_{\Theta\Theta} \mathcal{J}(\underline{\Theta}(i))\right]^{-1} \left[\nabla_\Theta \mathcal{J}(\underline{\Theta}(i))\right] \tag{7.6}$$

These Newton or quasi-Newton methods perform much better than the gradient algorithms closer to the minimum. In fact, if the criterion function is quadratic,

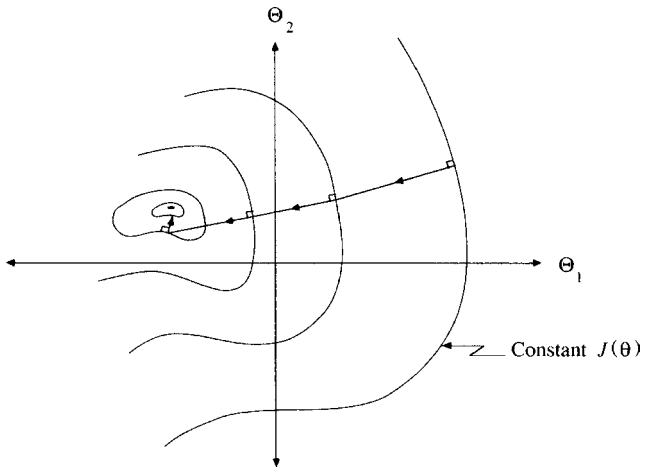



**Figure 7.2.** Typical gradient solution for a two-parameter surface.

then this algorithm will converge in one-step to the optimum solution, $\underline{\Theta}_{\text{opt}}$. Far from the minimum, these methods may be inefficient and the Hessian is replaced by a positive definite approximation to assure a downward point search direction [8], [9]. Other techniques based on different choices of step-size evolve as well (e.g., conjugate gradient method), but we will confine our discussion to these two cases.

If our criterion includes an expectation to account for randomness, then we obtain stochastic variants of these adaption schemes. *Stochastic gradient techniques* are based on replacing the deterministic criterion function with its stochastic counterpart which incorporates the expectation, that is,

$$\mathcal{J}(\underline{\Theta}) = E\{f(\underline{\Theta}, y)\} \longrightarrow f(\underline{\Theta}, y)$$

and the algorithms remain identical except for the above changes to the cost function.

Consider the following example to demonstrate these algorithms.

***Example 7.1*** Suppose that we are to estimate a constant $\alpha$ from a signal measurement $y$. We would like to obtain an estimate which is a linear function of the measurement, say,

$$\hat{\alpha} = \Theta y \quad \text{for } \Theta \text{ a constant}$$

We choose the criterion

$$\mathcal{J} = \tfrac{1}{2}e^2 = \tfrac{1}{2}(\alpha - \hat{\alpha})^2$$

and would like to develop the (i) gradient, and (ii) Newton iterates.

*Gradient Estimator*

$$\nabla_{\Theta}\mathcal{J} = -(\alpha - \Theta y)y = -(\alpha y - \Theta y^2) = -r + \Theta R$$

where we have defined $r := \alpha y$ and $R := y^2$. Thus we have the iterator

$$\Theta(i+1) = \Theta(i) - \triangle_i \nabla_{\Theta}\mathcal{J} = \Theta(i) + \triangle_i(r - \Theta(i)R)$$

or combining terms, we have

$$\Theta(i+1) = (1 - \triangle_i R)\Theta(i) + \triangle_i r$$

*Newton Estimator*

$$\nabla_{\Theta\Theta}\mathcal{J} = \nabla_{\Theta}(\nabla_{\Theta}\mathcal{J}) = \nabla_{\Theta}(-(r - \Theta R)) = R$$

Thus the Newton estimator for this problem is

$$\Theta(i+1) = \Theta(i) - \triangle_i [\nabla_{\Theta\Theta}\mathcal{J}]^{-1} [\nabla_{\Theta}\mathcal{J}] = \Theta(i) + \triangle_i R^{-1}(r - \Theta(i)R)$$

or

$$\Theta(i+1) = (1 - \triangle_i)\Theta(i) + \triangle_i r R^{-1}$$

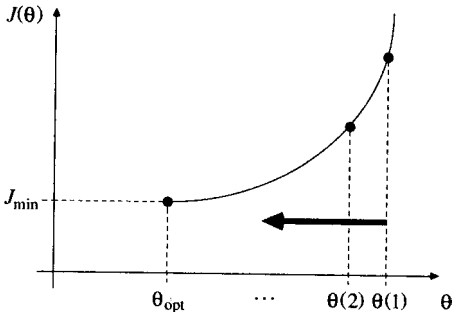



**Figure 7.3.** Gradient solution to constant parameter estimation problem.

If we assume that $\hat{\Theta} = \Theta_{\text{opt}}$, then the criterion can be written

$$\mathcal{J} = \mathcal{J}_{\min} + (\Theta - \Theta_{\text{opt}})^2 R$$

which clearly shows that the minimum is achieved when $\hat{\Theta} = \Theta_{\text{opt}}$. We depict the gradient algorithm in Figure 7.3 for this problem.

This completes the section on adaption algorithms. Although these gradient techniques are limited, they have become very popular in the signal processing area because of their simplicity and real-time applicability. Next we consider the development of all-zero adaptive processors.

## 7.3 ALL-ZERO ADAPTIVE *MBP*

In this section we consider the development of the all-zero or equivalently *FIR* adaptive (Wiener) filter. Actually we develop the *FIR* Wiener solution first, since it will be necessary for steady-state analysis of stationary signal $s(t)$. Then we show how these results can be used to develop and analyze an adaptive filter using the stochastic gradient algorithm. Finally we develop the stochastic approximation or "instantaneous gradient" version of the algorithm.

Following the adaptive signal processing method, we first select the model set to be all-zero. That is, we choose the model

$$s(t) = \sum_{i=0}^{N_h - 1} h(i)x(t-i) = \underline{h}'\underline{X} \tag{7.7}$$

for

$$\underline{h}' = [h(0) \cdots h(N_h - 1)], \quad \underline{X}' = [x(t) \cdots x(t - N_h + 1)]$$

Next we select the adaption algorithm such that the mean-squared error criterion is to be minimized

$$\min_{\underline{h}} \mathcal{J}(\underline{h}) = E\{e^2(t)\} \tag{7.8}$$

where $e(t) = s_d(t) - \hat{s}(t)$ and $s_d(t),\ \hat{s}(t)$ are the respective desired and estimated signals using the *FIR* model of Eq. (7.7).

Using the model, we can expand the criterion function as

$$\mathcal{J}(\underline{h}) := E\{e^2(t)\} = E\{(s_d(t) - \underline{h}'\underline{X})(s_d(t) - \underline{X}'\underline{h})\}$$

or

$$\mathcal{J}(\underline{h}) = E\{s_d^2(t)\} - 2\underline{h}'E\{s_d(t)\underline{X}\} + \underline{h}'E\{\underline{X}\underline{X}'\}\underline{h}$$

where we have used the fact that $\underline{h}'\underline{X} = \underline{X}'\underline{h}$. Now if we recognize these terms as variances and the fact that $\underline{h}'\underline{R} = \underline{R}'\underline{h}$, we obtain

$$\mathcal{J}(\underline{h}) = R_{ss} - 2\underline{R}'_{sx}\underline{h} + \underline{h}'\mathbf{R}_{xx}\underline{h} \tag{7.9}$$

Before we design the adaptive processor, let us first develop the steady-state or Wiener solution to this problem, since it will prove useful for performance analysis. As usual, the Wiener solution is found by minimizing the cost function, setting it to zero, and solving, that is

$$\nabla_h \mathcal{J}(\underline{h}) = \nabla_h \left[E\{e^2(t)\}\right] = \nabla_h \left[R_{ss} - 2\underline{R}'_{sx}\underline{h} + \underline{h}'\mathbf{R}_{xx}\underline{h}\right] = 0$$

Using the chain rule of Eq. (3.12), we obtain

$$\nabla_h \mathcal{J}(\underline{h}) = -2\underline{R}_{sx} + 2\mathbf{R}_{xx}\underline{h} = 0 \tag{7.10}$$

Solving, we obtain the optimal *Wiener solution*

$$\underline{h}_{\text{opt}} = \mathbf{R}_{xx}^{-1}\underline{R}_{sx} \tag{7.11}$$

with corresponding mean-squared error from Eq. (7.9),

$$\mathcal{J}(\underline{h}_{\text{opt}}) = R_{ss} - 2\underline{R}'_{sx}(\mathbf{R}_{xx}^{-1}\underline{R}_{sx}) + (\mathbf{R}_{xx}^{-1}\underline{R}_{sx})'\mathbf{R}_{xx}(\mathbf{R}_{xx}^{-1}\underline{R}_{sx})$$

This gives

$$\mathcal{J}_{\min} := \mathcal{J}(\underline{h}_{\text{opt}}) = R_{ss} - \underline{R}'_{sx}\mathbf{R}_{xx}^{-1}\underline{R}_{sx} = R_{ss} - \underline{h}'_{\text{opt}}\underline{R}_{sx} \tag{7.12}$$

Using this result, we are able to show that an alternate expression for the criterion is given by (see [1] for details)

$$\mathcal{J}(\underline{h}) = \mathcal{J}_{\min} + (\underline{h} - \underline{h}_{\text{opt}})'\mathbf{R}_{xx}(\underline{h} - \underline{h}_{\text{opt}}) \tag{7.13}$$

### 7.3.1 Stochastic Gradient Adaptive Processor

Choosing the stochastic gradient technique of the previous subsection, we have the correction term

$$\triangle\underline{\Theta}(i) := \triangle\underline{h}(i) = -\frac{\triangle_i}{2}\nabla_h \mathcal{J}(\underline{h}(i)) = -\frac{\triangle_i}{2}\nabla_h \left[E\{e^2(t)\}\right] \tag{7.14}$$

giving the parameter iterator

$$\underline{h}(i+1) = \underline{h}(i) + \frac{\triangle_i}{2}\nabla_h \left[E\{e^2(t)\}\right] \tag{7.15}$$

Substituting the expression for the gradient of Eq. (7.10), we have

$$\underline{h}(i+1) = \underline{h}(i) - \frac{\triangle_i}{2}\left[-2\underline{R}_{sx} + 2\mathbf{R}_{xx}\underline{h}(i)\right]$$

or

$$\underline{h}(i+1) = \underline{h}(i) + \triangle_i[\underline{R}_{sx} - \mathbf{R}_{xx}\underline{h}(i)]$$

which can be written as

$$\underline{h}(i+1) = (I - \triangle_i\mathbf{R}_{xx})\underline{h}(i) + \triangle_i\underline{R}_{sx} \tag{7.16}$$

The corresponding mean-squared error is at the $i$th iteration from Eq. (7.12):

$$\mathcal{J}(\underline{h}(i)) = R_{ss} - 2\underline{R}'_{sx}\underline{h}(i) + \underline{h}'(i)\mathbf{R}_{xx}\underline{h}(i) \tag{7.17}$$

Thus, in the model-based framework, the stochastic gradient algorithm can be summarized as follows:

| | |
|---|---|
| Criterion: | $\mathcal{J}(\underline{h}(i)) = E\{e^2(t)\}$ |
| Models: | |
| Signal: | $s(t) = \underline{h}'\underline{X}$ |
| Measurement: | $y(t) = s(t) + n(t)$ |
| Noise: | $n(t)$ random |
| Initial conditions: | $\underline{h}(0)$, $\triangle_i$ |
| Algorithm: | $\underline{h}(i+1) = \underline{h}(i) + \triangle_i[\underline{R}_{sx} - \mathbf{R}_{xx}\underline{h}(i)]$ |
| Quality: | $\mathcal{J}_{\min} = R_{ss} - \underline{h}'_{\text{opt}}\underline{R}_{sx}$ |

We summarize the stochastic gradient algorithm in Table 7.1. Note that implementing it will require that we know or have an estimate of the desired signal a-priori in order to determine the required variances. We must also have an ensemble to determine these variances or at least a good estimate of them. So theoretically the stochastic gradient algorithm cannot be implemented as presented. However, most practical algorithms make approximations of this form. Therefore it is important to analyze its performance to see the "best" it can achieve under ideal conditions.

The parameter estimator of the stochastic gradient algorithm can be analyzed by first subtracting the optimal solution from both sides of Eq. (7.16) and using the Wiener solution for the cross-correlation, that is,

$$\underline{h}(i+1) - \underline{h}_{\text{opt}} = (I - \triangle_i\mathbf{R}_{xx})\underline{h}(i) + \triangle_i(\mathbf{R}_{xx}\underline{h}_{\text{opt}}) - \underline{h}_{\text{opt}}$$

**Table 7.1. Stochastic Gradient All-Zero Algorithm**

For $i = 0, 1, \ldots$

Gradient estimation

$$\nabla_h \mathcal{J}(\underline{h}) = -2\underline{R}_{sx} + 2\mathbf{R}_{xx}\underline{h}(i)$$

Parameter iteration

$$\underline{h}(i+1) = \underline{h}(i) - \tfrac{\triangle_i}{2}\nabla_h \mathcal{J}(\underline{h}(i))$$

Criterion estimation

$$(\underline{h}(i) \rightarrow \underline{h}_{\text{opt}})$$

$$\mathcal{J}(\underline{h}_{\text{opt}}) = R_{ss} - \underline{h}'_{\text{opt}}\underline{R}_{sx}$$

Initial conditions

$$\underline{h}(0),\ \triangle_i$$

where we have solved Eq. (7.11) to obtain an expression for $\underline{R}_{sx}$ and substituted. Grouping terms, we have

$$[\underline{h}(i+1) - \underline{h}_{\text{opt}}] = (I - \triangle_i \mathbf{R}_{xx})[\underline{h}(i) - \underline{h}_{\text{opt}}]$$

or defining $\tilde{\underline{h}}(i) = \underline{h}(i) - \underline{h}_{\text{opt}}$, we have

$$\tilde{\underline{h}}(i+1) = (I - \triangle_i \mathbf{R}_{xx})\tilde{\underline{h}}(i) \tag{7.18}$$

But this is a homogeneous state-space equation (see Section 2.10)

$$\underline{x}(i+1) = \mathbf{A}\underline{x}(i)$$

with the solution

$$\underline{x}(i) = \mathbf{A}^i \underline{x}(0)$$

Therefore for our problem we have the solution

$$\tilde{\underline{h}}(i) = (I - \triangle \mathbf{R}_{xx})^i \tilde{\underline{h}}(0) \tag{7.19}$$

using $\tilde{\underline{h}} \rightarrow \underline{x}$, $\triangle_i \rightarrow \triangle$, and $(I - \triangle \mathbf{R}_{xx}) \rightarrow \mathbf{A}$, the state transition matrix.

To facilitate the investigation of the convergence properties of this algorithm, it is best to diagonalize the transition matrix using eigenvalue-eigenvector techniques. Recall from linear algebra [11] that the *eigenvalues* of the matrix $\mathbf{A}$, $\lambda(\mathbf{A})$ are the solutions of

$$(\mathbf{A} - \lambda I)\underline{v} = \underline{0} \tag{7.20}$$

with corresponding *characteristic equation*

$$\det(\mathbf{A} - \lambda I) = |\mathbf{A}(\lambda)| = \prod_{i=1}^{N_a}(\lambda - \lambda_i) = 0 \tag{7.21}$$

where $A(\lambda) = \lambda^{N_a} + a_1\lambda^{N_a-1} + \cdots + \lambda a_{N_a-1} + a_{N_a}$ and the roots of $A(\lambda)$ are the set of eigenvalues, $\lambda(\mathbf{A}) = \{\lambda_i\},\ i = 1, \ldots, N_a$.

Corresponding to each eigenvalue there exists at least one vector solution of the above equation determined by solving

$$\mathbf{A}\underline{v}_i = \lambda_i \underline{v}_i \tag{7.22}$$

where $\underline{v}_i$ is the $i$th *eigenvector* of $A$ with corresponding *eigenvalue*, $\lambda_i$. Expanding this equation over $i$, we have

$$\mathbf{A}[\underline{v}_1 \cdots \underline{v}_{N_a}] = [\underline{v}_1 \cdots \underline{v}_{N_a}] \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N_a} \end{bmatrix} \qquad \text{for } \underline{v}_i \in \mathcal{R}^{N_a \times 1}$$

or

$$\mathbf{AV} = \mathbf{V}\Lambda \qquad \text{for } \mathbf{V} \in \mathcal{R}^{N_a \times N_a}$$

This gives the so-called "normal" form of $A$ as

$$\mathbf{A} = \mathbf{V}\Lambda\mathbf{V}^{-1} \tag{7.23}$$

with $\Lambda$ the diagonal matrix of eigenvalues and $\mathbf{V}$ the eigenvector (*modal*) matrix.

For our problem, since $\mathbf{R}_{xx}$ is a symmetric and positive definite covariance matrix, we know the eigenvalues are real and orthogonal [11], that is,

$$\underline{v}_i'\underline{v}_j = 0 \quad \text{for } i \neq j$$

If they are normalized, then $\underline{v}_i'\underline{v}_j = \delta(i - j)$. So the modal matrix is orthonormal

$$\mathbf{V}\mathbf{V}' = I$$

implying $\mathbf{V}' = \mathbf{V}^{-1}$, a unitary matrix.

Using these results, we can investigate the convergence properties of the stochastic gradient algorithm by first transforming the state-space representation to modal form and using the modal matrix of $\mathbf{R}_{xx}$. That is, define the modal transformation

$$\underline{\tilde{h}}_M(i) := \mathbf{V}_M^{-1}\underline{\tilde{h}}(i)$$

substitute into Eq. (7.19) and solve for $\underline{\tilde{h}}_M(i+1)$ to obtain

$$\underline{\tilde{h}}_M(i+1) = \mathbf{V}_M^{-1}(I - \triangle\mathbf{R}_{xx})\mathbf{V}_M\underline{\tilde{h}}_M(i) = (I - \triangle\mathbf{V}_M^{-1}\mathbf{R}_{xx}\mathbf{V}_M)\underline{\tilde{h}}_M(i)$$

Alternatively, from Eq. (7.23), we can write

$$\underline{\tilde{h}}_M(i+1) = (I - \triangle\Lambda_{xx})\underline{\tilde{h}}_M(i) \tag{7.24}$$

where $\Lambda_{xx}$ is the eigenvalue matrix of $\mathbf{R}_{xx}$. Solving the state-space equation as before, but in modal coordinates, we obtain

$$\underline{\tilde{h}}_M(i) = (I - \triangle\Lambda_{xx})^i \underline{\tilde{h}}_M(0) \tag{7.25}$$

For convergence to the optimal solution, we require that

$$\lim_{i\to\infty} \underline{\tilde{h}}_M(i) = \lim_{i\to\infty} (I - \triangle\Lambda_{xx})^i \underline{\tilde{h}}_M(0) \longrightarrow 0 \tag{7.26}$$

or

$$\lim_{i\to\infty} \underline{\tilde{h}}_M(i) = \underline{h}_M(i) - \underline{h}_{\text{opt}} \longrightarrow 0$$

This implies that $\underline{h}_M(i) \to \underline{h}_{\text{opt}}$. Therefore we require that

$$\lim_{i\to\infty} (I - \triangle\Lambda_{xx})^i = \begin{bmatrix} \lim\limits_{i\to\infty} (1 - \triangle\lambda_1)^i & & 0 \\ & \ddots & \\ 0 & & \lim\limits_{i\to\infty} (1 - \triangle\lambda_N)^i \end{bmatrix} \longrightarrow 0$$

since $\Lambda_{xx}$ is the eigenvalue matrix, or equivalently

$$\lim_{i\to\infty} (1 - \triangle\lambda_j)^i \longrightarrow 0, \qquad j = 1, \ldots, N_h \tag{7.27}$$

which implies further that

$$|1 - \triangle\lambda_j| < 1, \qquad j = 1, \ldots, N_h$$

Therefore the step-size must be selected such that

$$0 < \triangle < \frac{2}{\lambda_j}, \qquad j = 1, \ldots, N_h \tag{7.28}$$

for convergence. *Stability* of the stochastic gradient algorithm is determined by the largest eigenvalue, since it must lie within the unit circle. Therefore the smallest allowable step-size is ($\lambda_j = \lambda_{\max}$)[1]

$$0 < \triangle < \frac{2}{\lambda_{\max}}$$

[1]Another approach is to use the fact that $\lambda_{\max} < \text{Trace } \mathbf{R}_{xx} = (N_h + 1)R_{xx}(0)$, which leads to the choice of step-size $\triangle < 2/(N_h + 1)R_{xx}(0)$.

The *speed of convergence* of the algorithm is determined by the smallest eigenvalue, since it represents the "slowest" converging mode, that is,

$$\tilde{h}_k(i) = (1 - \triangle\lambda_k)^i \tilde{h}_k(0) \qquad \text{for } k = \min$$

where $\tilde{h}_k$ is the $k$th component of $\underline{\tilde{h}}_M$ and this relation follows from Eq. (7.27).

Suppose that we choose the step-size to ensure stability. Then

$$\tilde{h}_k(i) = (1 - \triangle\lambda_{\min})^i \tilde{h}_k(0) = \left(1 - \left(\frac{1}{\lambda_{\max}}\right)\lambda_{\min}\right)^i \tilde{h}_k(0) = \left(1 - \frac{\lambda_{\min}}{\lambda_{\max}}\right)^i \tilde{h}_k(0) \tag{7.29}$$

Then the speed of convergence is determined by the ratio of the smallest to largest eigenvalue of $\mathbf{R}_{xx}$. Thus *speed of convergence* of the stochastic gradient algorithm is determined by the ratio

$$\frac{\lambda_{\min}}{\lambda_{\max}} = \begin{cases} 1, & \text{Fast } (\lambda_{\min} \approx \lambda_{\max}) \\ 0, & \text{Slow } (\lambda_{\min} \ll \lambda_{\max}) \end{cases} \tag{7.30}$$

It can be shown that the optimum choice of step-size is

$$\triangle = \frac{2}{\lambda_{\max} + \lambda_{\min}} \tag{7.31}$$

This ensures that all of the eigenvalues converge at the same speed [12], since

$$\tilde{h}_k(i) = (1 - \triangle\lambda_{\min})^i \tilde{h}_k(0) = \left(\frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}}\right)^i \tilde{h}_k(0)$$

or

$$\tilde{h}_k(i) = \left(\frac{1 - \dfrac{\lambda_{\min}}{\lambda_{\max}}}{1 + \dfrac{\lambda_{\min}}{\lambda_{\max}}}\right)^i \tilde{h}_k(0) \tag{7.32}$$

We can analyze the mean-squared error criterion in similar fashion using Eq. (7.13), we have

$$\mathcal{J}(\underline{h}) - \mathcal{J}_{\min} = (\underline{h} - \underline{h}_{\text{opt}})' \mathbf{R}_{xx} (\underline{h} - \underline{h}_{\text{opt}}) = \underline{\tilde{h}}' \mathbf{R}_{xx} \underline{\tilde{h}}$$

Transforming to modal coordinates, we obtain

$$\mathcal{J}(\underline{\tilde{h}}_M) - \mathcal{J}_{\min} = \underline{\tilde{h}}'_M (\mathbf{V}_M^{-1} \mathbf{R}_{xx} \mathbf{V}_M) \underline{\tilde{h}}_M = \underline{\tilde{h}}'_M \Lambda_{xx} \underline{\tilde{h}}_M = \text{constant} \tag{7.33}$$

which is the equation of a hyperellipse whose principle axis are defined by the eigenvectors of $\mathbf{R}_{xx}$. We show an example for the two-parameter case in Figure 7.4. Here we see that the algorithm converges to the minimum directly, while for

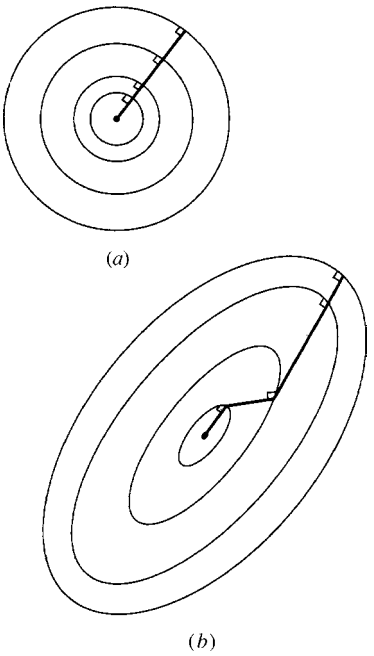



**Figure 7.4.** Stochastic gradient performance surface: *(a)* Two eigenvalues equal. *(b)* Two eigenvalues unequal.

unequal eigenvalues the gradient can diverge. So we see that through the use of eigenvalue-eigenvector and state-space techniques, we can evaluate the performance of the stochastic gradient algorithm.

One of the major problems with the stochastic gradient algorithm is that we rarely have the gradient covariance matrices available. Therefore we cannot implement the algorithm directly.

### 7.3.2 Instantaneous Gradient *LMS* Adaptive Processor

Here we consider an approximation to the stochastic gradient approach the least-mean-squared (*LMS*) algorithm [1]. The *LMS* algorithm employs the stochastic approximation technique to estimate the instantaneous gradient, that is,

$$\nabla_h \mathcal{J}(\underline{h}) = \nabla_h \left[ E\{e^2(t)\} \right] = E\left\{ \nabla_h e^2(t) \right\} \approx 2e(t)\nabla_h e(t)$$

where we have assumed that $e(t)$ and $\underline{X}(t)$ (to follow) are statistically independent and we have essentially *ignored* the ensemble expectation operation.

Thus, from the error, $e(t)$, of Eq. (7.8) we have[2]

$$\nabla_h e(t) = \nabla_h (s_d(t) - \underline{h}'\underline{X}(t)) = -\underline{X}(t)$$

[2]We now introduce time or iteration notation for the data vector, that is, $\underline{X} \rightarrow \underline{X}(t)$.

and the gradient is estimated by its instantaneous value

$$\hat{\nabla}_h \mathcal{J} = -2e(t)\underline{X}(t) \tag{7.34}$$

This is an *unbiased estimate*, since

$$E\left\{\hat{\nabla}_h \mathcal{J}\right\} = -2E\{e(t)\underline{X}\} = \nabla_h \mathcal{J}(\underline{h})$$

In the algorithm we replace the stochastic gradient by its instantaneous approximation leading to the parameter iterator

$$\underline{h}(i+1) = \underline{h}(i) - \frac{\triangle_i}{2}(-2e(t)\underline{X}(t)) = \underline{h}(i) + \triangle_i e(t)\underline{X}(t) \tag{7.35}$$

Expanding further by substituting for $e(t)$, we have

$$\underline{h}(i+1) = (I - \triangle_i \underline{X}(t)\underline{X}'(t))\underline{h}(i) + \triangle_i s_d(i)\underline{X}(t) \tag{7.36}$$

To analyze the performance of this algorithm, let us take the expected value of the iterator,

$$E\{\underline{h}(i+1)\} = E\{(I - \triangle_i \underline{X}(t)\underline{X}'(t))\underline{h}(i)\} + \triangle_i E\{s_d(t)\underline{X}(t)\}$$

Assuming that the data and parameter vectors are independent, we can separate the first term and obtain

$$\underline{m}_h(i+1) = (I - \triangle_i \mathbf{R}_{xx})\underline{m}_h(i) + \triangle_i \underline{R}_{sx} \tag{7.37}$$

where $\underline{m}_h(i) := E\{\underline{h}(i)\}$. Comparing this result to Eq. (7.16), we see that the *LMS* algorithm approximates the stochastic gradient algorithm "on the average." Convergence of the *average trajectory* therefore is identical to the stochastic gradient algorithm, since subtracting the optimal value from both sides of Eq. (7.37) leads to

$$\underline{\tilde{m}}_h(i+1) = (I - \triangle \mathbf{R}_{xx})\underline{\tilde{m}}_h(i) \tag{7.38}$$

As a result

$$\underline{\tilde{m}}_h(i) = (I - \triangle \mathbf{R}_{xx})^i \underline{\tilde{m}}_h(0)$$

where $\underline{\tilde{m}}_h(i) := \underline{m}_h(i) - \underline{m}_{h_{\text{opt}}}(i)$ and $0 < \triangle < 1/\lambda_k, \quad k = 1, \ldots, N_h$, as before. Note that this *only* implies that the *average of all* of the *LMS* trajectories converge to the optimum solution. Therefore we expect the *LMS* parameter estimates to fluctuate about the optimum randomly because the estimated or instantaneous gradient is random. These fluctuations can be controlled by the step-size $\triangle_i$, since it has been shown for large $N_h$ that [13]

$$E\{e^2(t)\} \approx \triangle \mathcal{J}_{\min}(\underline{h}) \tag{7.39}$$

Therefore the step-size is chosen as follows:

$$\triangle = \begin{cases} \text{Large—large fluctuations, rapid convergence} \\ \text{Small—small fluctuations, slow convergence} \end{cases}$$

Before we close this discussion, let us examine properties of the "average" mean-squared error

$$\overline{\mathcal{J}}(\underline{h}) := E\{\mathcal{J}(\underline{h})\}$$

Performing the required analysis, it can be shown that [1] for the *LMS* algorithm to *converge (in mean-square)* the step-size must satisfy

$$0 < \triangle_i < \frac{1}{\text{Trace } \mathbf{R}_{xx}} = \frac{1}{\sum_{j=1}^{N_h} \lambda_j} = \frac{1}{(\text{Total input power})} \tag{7.40}$$

From Eq. (7.38) we see that for *convergence (in the mean)* the step-size must satisfy

$$0 < \triangle_i < \frac{2}{\lambda_{\max}} \tag{7.41}$$

Since $\lambda_{\max} <$ Trace $\mathbf{R}_{xx}$, the convergence *both* in the mean and the mean-squared are satisfied by the step-size constraint specified in Eq. (7.40).

Another common measure of performance is that of *misadjustment* defined (in practice) by

$$M = \frac{\text{Average excess mean-squared error}}{\text{Minimum mean-squared error}} = \triangle \text{ Trace } \mathbf{R}_{xx} \tag{7.42}$$

The corresponding "time constant" or decay of $\overline{\mathcal{J}}(\underline{h})$ can be approximated by

$$\tau_{\text{mse}} \approx \frac{1}{4\triangle\lambda_j}$$

for the $j$th eigenvalue or on the *average*

$$\overline{\tau}_{\text{mse}} \approx \frac{N_h}{4\triangle \text{Trace } \mathbf{R}_{xx}} \tag{7.43}$$

The *LMS* technique is a model-based scheme that can be summarized as follows:

Criterion: $\mathcal{J}(\underline{h}(t)) = E\{e^2(t)\}$
Models:
Signal: $s(t) = \underline{h}'\underline{X}(t)$
Measurement: $y(t) = s(t) + n(t)$
Noise: $n(t)$ random
Initial conditions: $\underline{h}(0), \triangle$

**Table 7.2. LMS Instantaneous Gradient Adaptive Algorithm**

For $t = 0, 1, \ldots, N$

Error estimate

$$e(t) = s_d(t) - \hat{s}(t) = s_d(t) - \underline{h}(t)\underline{X}(t)$$

Instantaneous gradient estimation

$$\nabla_h \mathcal{J}(\underline{h}) = -2e(t)\underline{X}(t)$$

Step-size

$$0 < \triangle < \frac{1}{\text{Trace } \mathbf{R}_{xx}}$$

Parameter iteration

$$\underline{h}(t+1) = \underline{h}(t) - \frac{\triangle}{2}\nabla_h \mathcal{J}(\underline{h}(t)) = \underline{h}(t) + \triangle e(t)\underline{X}(t)$$

Criterion estimation

$$(\underline{h}(i) \rightarrow \underline{h}_{\text{opt}})$$

$$\mathcal{J}(\underline{h}_{\text{opt}}) = E\{e^2(t)\} \approx \frac{1}{N}\sum_{t=1}^{N} e^2(t)$$

Initial conditions

$$\underline{h}(0), \ \triangle$$

Algorithm: $\underline{h}(t+1) = \underline{h}(t) + \triangle e(t)\underline{X}(t)$
Quality: $\mathcal{J}_{\text{mse}} = \frac{1}{N}\sum_{t=1}^{N} e^2(t)$

We summarize the *LMS* algorithm in Table 7.2. Next we consider a modification to this approach.

### 7.3.3 Normalized *LMS* Adaptive Processor

In this subsection we briefly discuss a modification of the conventional *LMS* algorithm discussed above. One of the major difficulties in implementing the *LMS* technique is the selection of the step-size, which directly impacts its overall performance (convergence and stability). We know from Eq. (7.41) that the step-size is bounded above by $(2/\lambda_{\max})$ where $\lambda_{\max}$ is the maximum eigenvalue of the input covariance matrix, $\mathbf{R}_{xx}$. A more conservative and pragmatic bound that is used is based on the fact that

$$\lambda_{\max} < \text{Trace } \mathbf{R}_{xx} = \sum_{i=1}^{N_h} \lambda_i = N_h \times R_{xx}(0) \tag{7.44}$$

for $R_{xx}(0)$ the variance or power of the input signal and $\mathbf{R}_{xx}$ a Toeplitz matrix. Therefore the step-size of Eq. (7.41) is bounded above by the *inverse* of the input signal power. This relation can be expressed simply by using the properties of the

*trace* operation, that is,

$$\text{Trace } \mathbf{R}_{xx} = \text{Trace } E\{\underline{X}(t)\underline{X}'(t)\} = E\{\underline{X}'(t)\underline{X}(t)\} \approx \| \underline{X}(t) \|^2 \tag{7.45}$$

Therefore the step-size can be chosen as

$$\triangle_t = \frac{\alpha}{\| \underline{X}(t) \|^2 + \beta} \qquad \text{for } 0 < \alpha < 2, \ \beta > 0 \tag{7.46}$$

which ensures that it produces a bounded and stable estimate. The positive constant $\beta$ is included to ensure that if $\| \underline{X}(t) \|^2 \rightarrow 0$, the step-size remains bounded and finite. Replacing the parametric iteration with the "normalized" step gives the normalized least mean square *NLMS* algorithm

$$\underline{h}(t+1) = \underline{h}(t) + \alpha \left( \frac{\underline{X}(t)}{\| \underline{X}(t) \|^2 + \beta} \right) e(t) \tag{7.47}$$

The squared norm scales the magnitude of the gradient vector, but *not* its direction. It can be shown that the *NLMS* algorithm converges in the mean-square as long as $0 < \alpha < 2$ [2], [14], [15]. The additional advantage of the normalization is to decrease the gradient noise.

It should also be noted that we motivated the development of the *NLMS* approach from a practical and well-founded viewpoint. However, both Haykin [2] and Sayed [10] derive the algorithm directly from optimization theory. The former uses a constrained optimization approach, while the latter uses the basic Newton recursion with a dyadic update. In any case, the result is identical—the *NLMS* algorithm.

The actual implementation of the *NLMS* algorithm can be accomplished in a batch-type mode where the squared norm is updated at each step by performing the appropriate multiplication, or it can be performed recursively. In fact, if we define the variance of the data $x(t)$ as $V_{xx}(t)$, then we can perform this weighted updating recursively as

$$\hat{V}_{xx}(t) = \gamma \hat{V}_{xx}(t-1) + (1-\gamma)x^2(t) \tag{7.48}$$

for $\gamma$ the forgetting factor [5]. We summarize the recursive normalized least mean square (*RNLMS*) algorithm in Table 7.3. In either case the results are quite similar in the limit. However, the *RNLMS* approach is more versatile with the use of the forgetting factor, especially for truly nonstationary (time-varying) systems.

Following the *LMS* approach, both *NLMS* and *RNLMS* techniques are model-based and can be summarized as follows:

Criterion: $\mathcal{J}(\underline{h}(t)) = E\{e^2(t)\}$
Models:
Signal: $s(t) = \underline{h}'\underline{X}(t)$
Measurement: $y(t) = s(t) + n(t)$

**Table 7.3. Normalized *RNLMS* Algorithm**

For $t = 1, 2, \ldots$

Signal estimation

$$\hat{s}(t) = \sum_{k=0}^{N_h-1} h(k)x(t-k) = \underline{h}'\underline{X}(t)$$

Error estimate

$$e(t) = s_d(t) - \hat{s}(t)$$

Step-size update

$$\hat{V}_{xx}(t) = \gamma \hat{V}_{xx}(t-1) + (1-\gamma)x^2(t)$$

$$\triangle_t = \frac{\alpha}{\hat{V}_{xx}(t) + \beta}$$

Parameter iteration

$$\underline{h}(t+1) = \underline{h}(t) + \triangle_t e(t)\underline{X}(t)$$

Initial conditions

$$\underline{h}(0),\ \hat{V}_{xx}(0),\ \alpha,\ \beta,\ \gamma$$

Noise: $n(t)$ random

Initial conditions: $\underline{h}(0)$, $\triangle_t = \dfrac{\alpha}{\hat{V}_{xx}(t) + \beta}$, $\alpha$, $\beta$

Algorithm: $\underline{h}(t+1) = \underline{h}(t) + \triangle_t e(t)\underline{X}(t)$

Quality: $\mathcal{J}_{\text{mse}} = \frac{1}{N}\sum_{t=1}^{N} e^2(t)$

Before we close this section on *FIR* adaptive filter design, let us briefly investigate the stochastic Newton-like algorithm to understand its relationship to the stochastic gradient and *LMS* approaches. Recall that the stochastic-Newton algorithm employs the Hessian and gradient for its direction vector, that is,

$$\triangle_i \underline{d}(i) = \triangle_i \underline{h}_i = -\frac{\triangle_i}{2}\left[\nabla_{hh}\mathcal{J}(\underline{h})\right]^{-1}\left[\nabla_h \mathcal{J}(\underline{h})\right] \tag{7.49}$$

Therefore for our problem, we have from Eq. (7.10),

$$\nabla_{hh}\mathcal{J}(\underline{h}) = \nabla_h\left(\nabla_h \mathcal{J}(\underline{h})\right)' = \nabla_h(-2\underline{R}_{sx} + 2\mathbf{R}_{xx}\underline{h}) = 2\mathbf{R}_{xx} \tag{7.50}$$

The Newton parameter iterator becomes

$$\underline{h}(i+1) = \underline{h}(i) - \frac{\triangle_i}{2}[2\mathbf{R}_{xx}]^{-1}\left(-2\underline{R}_{sx} + 2\mathbf{R}_{xx}\underline{h}(i)\right)$$

or

$$\underline{h}(i+1) = \underline{h}(i) + \frac{\triangle_i}{2}[\mathbf{R}_{xx}^{-1}\underline{R}_{sx} - \underline{h}(i)] = \left(1 - \frac{\triangle_i}{2}\right)\underline{h}(i) + \frac{\triangle_i}{2}\mathbf{R}_{xx}^{-1}\underline{R}_{sx} \tag{7.51}$$

which shows a one-step convergence, since $\underline{h}_{\text{opt}} = \mathbf{R}_{xx}^{-1}\underline{R}_{sx}$. This leads us to a completes class of stochastic "Newton-like" algorithms in which we specify the iteration

$$\underline{h}(i+1) = \underline{h}(i) - \triangle_i W[-\underline{R}_{sx} + \mathbf{R}_{xx}\underline{h}(i)] \tag{7.52}$$

where we have

$$W = \begin{cases} [2\mathbf{R}_{xx}]^{-1} & \text{for stochastic Newton} \\ I & \text{for stochastic gradient (and } \textit{LMS}) \end{cases} \tag{7.53}$$

### 7.3.4 Recursive Least-Squares (*RLS*) Adaptive Processor

A final approach to this adaptive problem is to use the recursive least-squares (*RLS*) approach which is essentially a special case of the *RPEM* of Section 4.5 ($\Psi = \Phi,\ A = C = 1,\ B = H$) [5]. As before, suppose that we define the cost function as the sum-squared error with

$$\mathcal{J}_t(\underline{h}) = \frac{1}{N_h}\sum_{t=1}^{N_h} \gamma_t e^2(t) \tag{7.54}$$

where $e(t) = s_d(t) - \hat{s}(t)$, $\gamma_t$ is the weighting or forgetting factor, and we have the all-zero model set

$$\hat{s}(t) = \sum_{i=1}^{N_h} h(i)x(t-i) = \underline{h}'\underline{X}(t)$$

Minimizing Eq. (7.54) with respect to $\underline{h}$, we obtain the usual orthogonality conditions

$$\nabla_h \mathcal{J} = -2\sum_{t=1}^{N_h} \gamma_t e(t)\underline{X}'(t) = -2\sum_{t=1}^{N_h} \gamma_t (s_d(t) - \underline{h}'\underline{X}(t))\underline{X}'(t) = 0$$

or

$$\underline{h}'\left(\sum_{t=1}^{N_h} \gamma_t \underline{X}(t)\underline{X}'(t)\right) = \sum_{t=1}^{N_h} \gamma_t s_d(t)\underline{X}'(t)$$

Transposing and noting the scalars, we obtain

$$\left(\sum_{t=1}^{N_h} \gamma_t \underline{X}(t)\underline{X}'(t)\right)\underline{h} = \sum_{t=1}^{N} \gamma_t s_d(t)\underline{X}(t) \tag{7.55}$$

Now, if we define

$$\mathbf{R}(N_h) := \sum_{t=1}^{N_h} \gamma_t \underline{X}(t)\underline{X}'(t)$$

$$\underline{R}(N_h) := \sum_{t=1}^{N_h} \gamma_t s_d(t)\underline{X}(t)$$

then we have the Wiener solution as before

$$\underline{h}(N_h) = \mathbf{R}^{-1}(N_h)\underline{R}(N_h) \tag{7.56}$$

This is the least-squares version of the Wiener solution (compare with Eq. 7.11).

Both of these relations can be placed in recursive-form with a change in notation first. That is, we rewrite the equations with $N_h \to t$ and $t \to i$ to obtain

$$\mathbf{R}(t) = \sum_{i=1}^{t} \gamma_i \underline{X}(i)\underline{X}'(i)$$

$$\underline{R}(t) := \sum_{i=1}^{t} \gamma_i s_d(i)\underline{X}(i)$$

Now by removing the $t$th term from the sums and identifying the remaining term, we have

$$\mathbf{R}(t) = \sum_{i=1}^{t-1} \gamma_i \underline{X}(i)\underline{X}'(i) + \gamma_t \underline{X}(t)\underline{X}'(t) = \mathbf{R}(t-1) + \gamma_t \underline{X}(t)\underline{X}'(t) \tag{7.57}$$

Similarly

$$\underline{R}(t) = \underline{R}(t-1) + \gamma_t s_d(t)\underline{X}(t) \tag{7.58}$$

But from Eq. (7.56) we have

$$\underline{R}(t-1) = \mathbf{R}(t-1)\underline{h}(t-1)$$

Using Eq. 7.57 to substitute for $\mathbf{R}(t-1)$, this becomes

$$\underline{R}(t-1) = [\mathbf{R}(t) - \gamma_t \underline{X}(t)\underline{X}'(t)]\underline{h}(t-1)$$

Substituting Eq. (7.58) into the Wiener solution, we have

$$\underline{h}(t) = \mathbf{R}^{-1}(t)\underline{R}(t) = \mathbf{R}^{-1}(t)[\underline{R}(t-1) + \gamma_t s_d(t)\underline{X}(t)]$$

or

$$\underline{h}(t) = \mathbf{R}^{-1}(t)[\mathbf{R}(t)\underline{h}(t-1) - \gamma_t \underline{X}(t)\underline{X}'(t)\underline{h}(t-1) + \gamma_t s_d(t)\underline{X}(t)]$$

Multiplying through, we obtain the recursion

$$\underline{h}(t) = \underline{h}(t-1) + \gamma_t \mathbf{R}^{-1}(t)[s_d(t)\underline{X}(t) - \underline{X}(t)\underline{X}'(t)\underline{h}(t-1)]$$
$$\mathbf{R}(t) = \mathbf{R}(t-1) + \gamma_t \underline{X}(t)\underline{X}'(t) \tag{7.59}$$

We can interpret this algorithm as a "Newton-like" technique, when we identify

$$\gamma_t \longrightarrow \triangle_t, \quad \mathbf{W} \longrightarrow \mathbf{R}^{-1}(t) \longrightarrow \mathbf{R}_{xx}^{-1}$$

with appropriate gradients. In practice, it is possible to replace this algorithm and avoid inverting $\mathbf{R}(t)$ by using the "matrix inversion lemma" (see Chapter 3 for details). That is, we define

$$\mathbf{P}(t) = \mathbf{R}^{-1}(t)$$

and we obtain the *RLS* recursion

$$\underline{h}(t) = \underline{h}(t-1) + \underline{K}(t)e(t)$$

where

$$\underline{K}(t) = \frac{\mathbf{P}(t-1)\underline{X}(t)}{1/\gamma_t + \underline{X}'(t)\mathbf{P}(t-1)\underline{X}(t)}$$

for

$$\mathbf{P}(t) = \mathbf{P}(t-1) - \frac{\mathbf{P}(t-1)\underline{X}(t)\underline{X}'(t)\mathbf{P}(t-1)}{1/\gamma_t + \underline{X}'(t)\mathbf{P}(t-1)\underline{X}(t)} \tag{7.60}$$

As before, it is possible to analyze the performance of the *RLS* algorithm in terms of its convergence properties [2]. Defining the *average* sum-squared error as

$$\overline{\mathcal{J}}(\underline{h}) := E\{\mathcal{J}(\underline{h})\}$$

with $\gamma_t = 1$, we can show that [2]

$$\overline{\mathcal{J}}_t(\underline{h}) \approx \mathcal{J}_{\min}\left(1 + \frac{N_h}{t}\right) \tag{7.61}$$

This implies that as the number of data samples gets large ($t \to \infty$),

$$\lim_{t\to\infty} \overline{\mathcal{J}}_t(\underline{h}) \approx \mathcal{J}_{\min}$$

Therefore the *RLS* algorithm can achieve "zero" misadjustment, that is,

$$\lim_{t\to\infty} M_{RLS} \longrightarrow 0$$

Comparing the *RLS* to *LMS* approaches, we see that the *RLS* algorithm is an order of magnitude faster in convergence and can actually achieve the optimum (minimum) mean-squared error or "zero misadjustment." However, the cost is in computational complexity since *RLS* requires $3N_h(N_h+3)/2$ multiplications compared to $2N_h+1$ for the *LMS* algorithm. The *RLS* adaptive technique is a model-based processor characterized by the following:

Criterion: $\mathcal{J}(\underline{h}(t)) = \frac{1}{N}\sum_{t=1}^{N}\gamma_t e^2(t)$

Models:

Signal: $s(t) = \underline{h}'\underline{X}(t)$

Measurement: $y(t) = s(t) + n(t)$

Noise: $n(t)$ random

Initial conditions: $\mathbf{P}(0) = \alpha I,\ \underline{h}(0) = \underline{0},\ 0 < \gamma(0) < 1,\ \alpha$ large

Algorithm: $\underline{h}(t) = \underline{h}(t-1) + \underline{K}(t)e(t)$

Quality: $\mathbf{P}(t) = \mathbf{P}(t-1) - \dfrac{\mathbf{P}(t-1)\underline{X}(t)\underline{X}'(t)\mathbf{P}(t-1)}{1/\gamma_t + \underline{X}'(t)\mathbf{P}(t-1)\underline{X}(t)}$

We summarize the adaptive *RLS* algorithm in Table 7.4. Note that this is not the preferred form of implementation. The "square-root" or U-D factorized form (see

**Table 7.4. *RLS* Adaptive All-Zero Algorithm**

Signal estimate

$$\hat{s}(t) = \sum_{i=1}^{N_h} h(i)x(t-k) = \underline{h}'\underline{X}(t)$$

Error estimate

$$e(t) = s_d(t) - \hat{s}(t)$$

Covariance estimate

$$\mathbf{P}(t) = \mathbf{P}(t-1) - \frac{\mathbf{P}(t-1)\underline{X}(t)\underline{X}'(t)\mathbf{P}(t-1)}{1/\gamma_t + \underline{X}'(t)\mathbf{P}(t-1)\underline{X}(t)}$$

Gain update

$$\underline{K}(t) = \frac{\mathbf{P}(t-1)\underline{X}(t)}{1/\gamma_t + \underline{X}'(t)\mathbf{P}(t-1)\underline{X}(t)}$$

Parameter iteration

$$\underline{h}(t) = \underline{h}(t-1) + \underline{K}(t)e(t)$$

Forgetting factor update

$$\gamma_t = \gamma_0\gamma_{t-1} + (1-\gamma_0)$$

Initial conditions

$$\mathbf{P}(0) = \alpha I, \quad \underline{h}(0) = \underline{0}, \quad 0 < \gamma_0 < 1, \quad \alpha \text{ large}$$

Appendix D for details) is the popular approach because of its superior numerical properties. We summarize these ideas with a simple example.

***Example 7.2*** Suppose that we are asked to design a single weight adaptive *FIR* filter using the (i) stochastic gradient, (ii) stochastic Newton, (iii) *LMS* and (iv) *RLS* algorithms, to analyze their performance (speed and stability). The structure of the processor is simply

$$\hat{s}(t) = hx(t)$$

The steady-state Wiener solution is

$$h_{\text{opt}} = \frac{R_{sx}}{R_{xx}}$$

The corresponding gradient and Hessian are given by

$$\nabla_h \mathcal{J} = 2E\{e(t)\nabla_h e(t)\} = -2E\{(s_d(t) - hx(t))x(t)\} = -2R_{sx} + 2R_{xx}h$$

with

$$\nabla_{hh}\mathcal{J} = 2R_{xx}$$

The adaption algorithms are:

i. Stochastic gradient: $h(i+1) = h(i) + \triangle_t(R_{sx} - R_{xx}h(i))$
ii. Stochastic Newton: $h(i+1) = h(i) + \triangle_t\left(\frac{R_{sx}}{R_{xx}} - h(i)\right)$
iii. *LMS*: $h(t+1) = h(t) + \triangle_t e(t)x(t)$
iv. *RLS*: $h(t+1) = h(t) + K(t)e(t)$

We can determine the speed of convergence from the relations

i. Gradient: $\tilde{h}(i) = (1 - \triangle_t R_{xx})^i \tilde{h}(0) \Rightarrow 0 < \triangle_t < \frac{2}{R_{xx}}$
ii. Newton: $\tilde{h}(i) = (1 - \triangle_t)^i \tilde{h}(0) \Rightarrow 0 < \triangle_t$
iii. *LMS*: $\tilde{m}_h(i) = (1 - \triangle_t R_{xx})^i \tilde{m}_h(0) \Rightarrow 0 < \triangle_t < \frac{2}{R_{xx}}$
iv. *RLS*: $\overline{\mathcal{J}}_t(\underline{h}) = \mathcal{J}_{\min}(1 + \frac{1}{t})$

We now generate a simulation using *MATLAB* [16], for our desired signal, we choose a discrete exponential

$$s_d(t) = 0.95s_d(t-1) + e(t) \qquad \text{with } s_d(0) = 1,\ e \sim \mathcal{N}(0, 0.01)$$

so that

$$h_d(t) = (0.95)^t \qquad \text{(Ignoring noise)}$$

Next the measured signal is scaled and with zero-mean, unit variance white noise,

$$x(t) = 0.5s_d(t) + n(t)$$

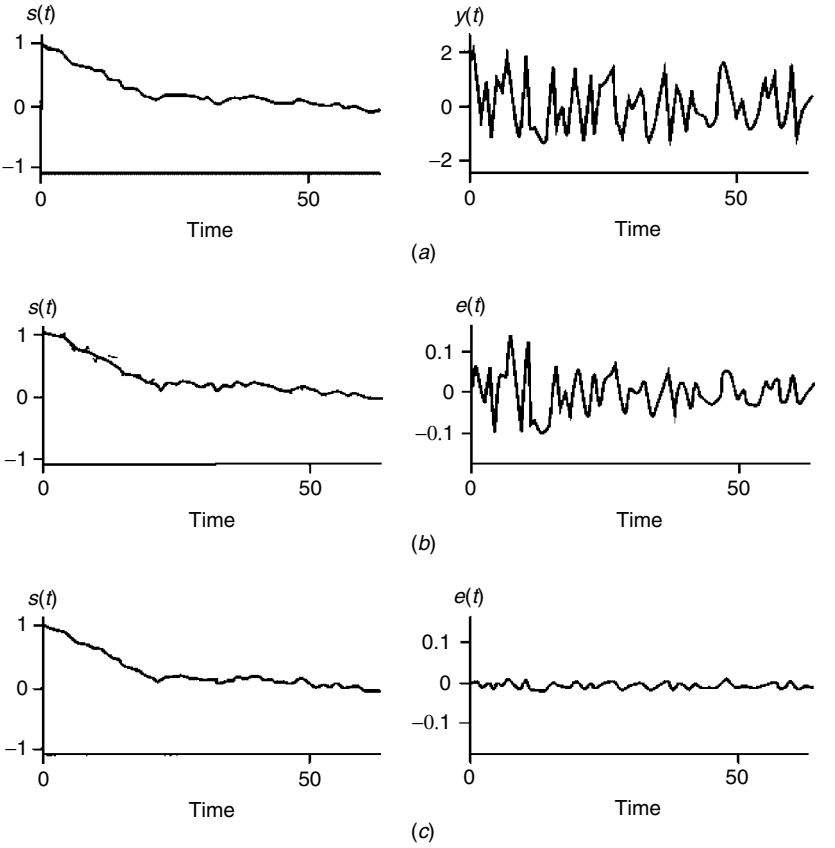



**Figure 7.5.** Adaptive single-weight processing example. *(a)* Simulated signal and measurement. *(b)* Adaptive processor for signal and error for $\triangle_t = 0.01$. *(c)* Adaptive processor for signal and error for $\triangle_t = 0.001$.

The simulation results are shown in Figure 7.5. Here we see the results of the adaptive *LMS* solution and the corresponding errors for two choices of step-size, $\triangle_t = 0.01,\ 0.001$. We see the simulated (noisy) signal and measurement as well as the signal estimated for each choice. In both cases the processor tracks the signal, but as expected from Eq. (7.9), the larger step-size yields noisier results. This completes the example, next we consider another simulation.

***Example 7.3*** Suppose that we consider the design of an adaptive processor to solve the signal estimation problem of the previous chapter. That is, we have an *ARMAX*(2, 1, 1, 1) model with difference equation in operator for ($q^{-i}x(t) := x(t-i)$),

$$(1 - 1.5q^{-1} + 0.7q^{-2})y(t) = (1 + 0.4q^{-1})u(t) + \frac{1 + 0.3q^{-1}}{1 + 0.5q^{-1}}e(t)$$

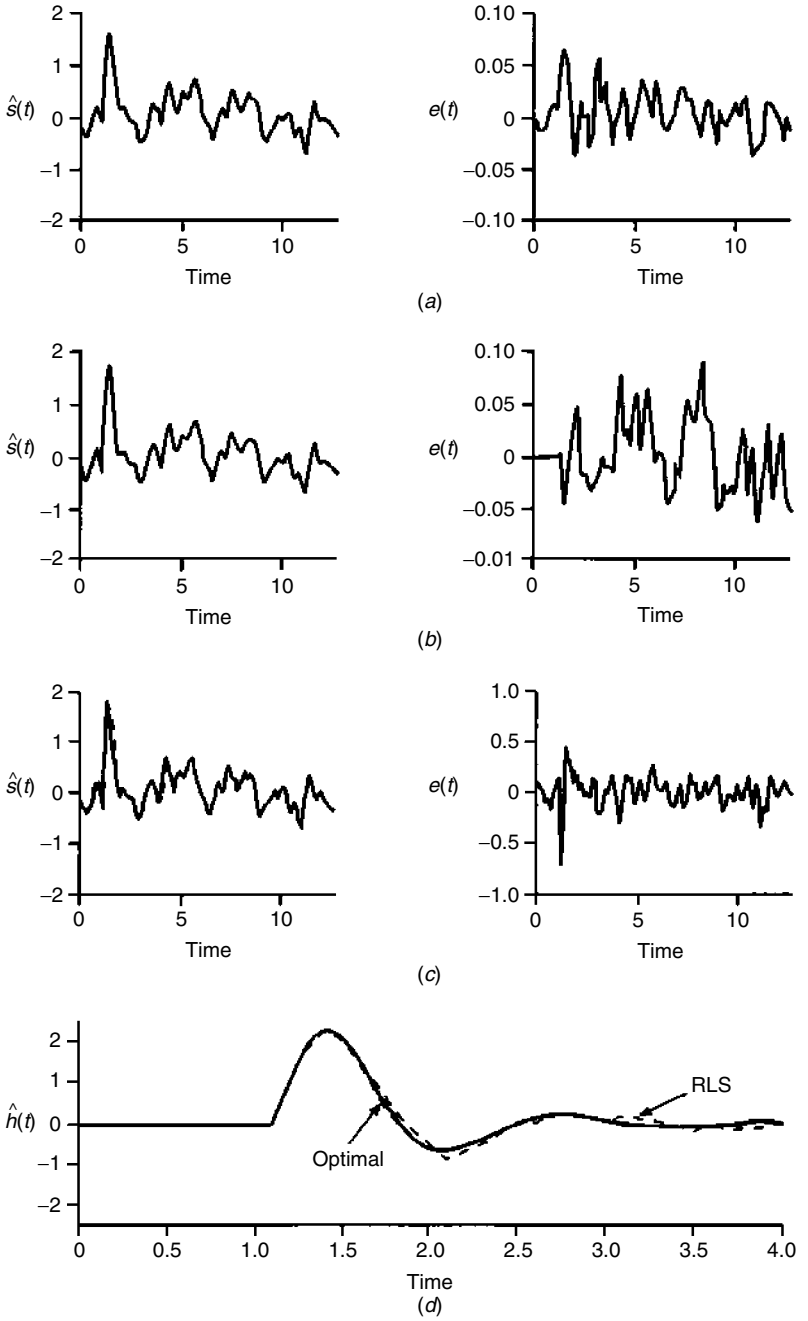



**Figure 7.6.** Adaptive *FIR* processor design for *ARMAX*(2, 1, 1, 1) simulation. *(a)* Optimal Wiener (stationary) solution and error. *(b)* Adaptive (*LMS*) solution and error. *(c)* Adaptive (*RLS*) solution and error. *(d)* True impulse response and optimal Wiener and *RLS* solutions.

or in the equivalent difference equation form,

$$y(t) - 1.5y(t-1) + 0.7y(t-2) = u(t) + 0.4u(t-1) + \eta(t)$$
$$\eta(t) + 0.5\eta(t-1) = e(t) + 0.3e(t-1)$$

where $u(t)$ is a delayed unit impulse and $\eta(t)$ is the *coloring filter* output with $e \sim \mathcal{N}(0, 0.01)$. Using *MATLAB* [16], we develop the optimal Wiener (stationary) solution, *LMS* adaptive solution, and *RLS* adaptive solution. The results are shown in Figure 7.6. Note that the best we can hope to achieve here is the Wiener solution, since the data are stationary. Thus we expect this adaptive results to converge to the Wiener solution. The estimated signals and corresponding residual errors are shown. In each case a 32-weight *FIR* processor was designed, and we see that each processor "tracks" the measurement as indicated by the resulting errors. From the tracking it appears that the *RLS* solution approaches the optimal. This is confirmed in Figure 7.6*d* where we have the true signal (impulse response), optimal Wiener and *RLS* solutions.

This completes the section on adaptive *FIR* processing, next we develop the pole-zero adaptive solutions.

## 7.4 POLE-ZERO ADAPTIVE *MBP*

In this section we consider the design of the pole-zero or equivalently *IIR* adaptive Wiener filters. Adaptive *IIR* filters offer advantages over *FIR* filters, in that, they require significantly fewer weights to achieve equivalent performance. For instance, it is well known [1] that large order *FIR* filters are required to estimate sinusoids in noise, while the *IIR* designs theoretically only require $2N_s$ parameters to estimate $N_s$ sinusoids. *IIR* processors also offer the usual features of sharp cutoffs and resonances that *FIR* designs can realize only with high orders. However, *IIR* processors have disadvantages as well. First, their performance surfaces are not quadratic and can have local minima. The processor can become unstable during adaption as well. The work of Goodwin [4], Ljung [5], and others [15] have greatly aided in understanding the convergence of these techniques and improved performance by incorporating stability tests directly into the algorithms enabling them to be considered alternatives to *FIR* adaptive techniques.

### 7.4.1 IIR Adaptive MBP

Following the approach outlined in the introduction, the underlying model set is selected to be pole-zero with corresponding signal model

$$s(t) = -\sum_{i=1}^{N_a} a_i y(t-i) + \sum_{i=0}^{N_b} b_i u(t-i) \tag{7.62}$$

which can conveniently be expressed in vector form as

$$s(t) = \underline{\Theta}'(t)\underline{X}(t) \tag{7.63}$$

where the data vector is

$$\underline{X}'(t) = [-y(t-1)\cdots - y(t-N_a) \mid u(t)\cdots u(t-N_b)]$$

The parameter vector is given by

$$\underline{\Theta}'(t) = [a_1(t)\cdots a_{N_a}(t) \mid b_0(t)\cdots b_{N_b}(t)]$$

The underlying measurement model is given by

$$y(t) = s(t) + n(t) \tag{7.64}$$

which (substituting Eq. 7.63) leads to the $ARMAX(N_a, N_b, 0)$ or more succinctly *ARX* model as a special case used in the parametric processor ($\underline{X} = \underline{\phi}$). Clearly, this evolves to the various *RPEM* methods or so-called "exact least-squares" methods of adaptive processing. In fact Friedlander [17] shows how to construct various adaptive processors using this approach.

We will take the iterative gradient approach to develop our adaptive algorithm. As before, we select the algorithm such that the mean-squared error criterion is to be minimized

$$\min_{\underline{\Theta}} \mathcal{J}(t) = E\{e^2(t)\} \tag{7.65}$$

where $e(t) = s_d(t) - \hat{s}(t)$ and $s_d(t)$, $\hat{s}(t)$ are the respective desired and estimated (using Eq. 7.63) signals.

Performing the minimization in the usual manner, we have

$$\nabla_\Theta \mathcal{J}(\underline{\Theta}) = 2E\{e(t)\nabla_\Theta e(t)\} = -2E\{e(t)\nabla_\Theta y(t)\} \tag{7.66}$$

but now the error gradient is not as simple as before because of the *IIR* model structure governing $y(t)$. Thus, expanding this expression, we have

$$\nabla_\Theta \mathcal{J}(\underline{\Theta}) = -2E\left\{e(t)\begin{bmatrix} \frac{\partial y(t)}{\partial a_i} \\ --- \\ \frac{\partial y(t)}{\partial b_j} \end{bmatrix}\right\}, \qquad i = 1,\cdots,N_a;\ j = 0,\cdots,N_b \tag{7.67}$$

Using the *IIR* model, we obtain

$$\nabla_a y(t) := \nabla_{a_i} y(t) = -y(t-i) + \sum_{j=1}^{N_a} a_j \nabla_a y(t-j), \qquad i = 1,\cdots,N_a$$

and

$$\nabla_b y(t) := \nabla_{b_i} y(t) = u(t-i) + \sum_{j=1}^{N_a} a_j \nabla_b y(t-j), \qquad i = 0,\cdots,N_b$$

Alternatively, defining $\nabla_\Theta y(t) := [\nabla_a y(t) \mid \nabla_b y(t)]'$, we have the vector recursion

$$\nabla_\Theta y(t) = \underline{X}(t) + \sum_{j=1}^{N_a} a_j \nabla_\Theta y(t-j) \tag{7.68}$$

Therefore the gradient is given by

$$\nabla_\Theta \mathcal{J}(\underline{\Theta}) = -2E\{e(t)\nabla_\Theta y(t)\} \tag{7.69}$$

Expanding and taking expectations leads to a complex expression for the stochastic gradient. Therefore we limit our discussion to the instantaneous gradient or *LMS* approach given by the iteration

$$\underline{\Theta}(t+1) = \underline{\Theta}(t) - \frac{\triangle_t}{2}[-2e(t)\nabla_\Theta y(t)]$$

where the estimated gradient is obtained using the recursion of Eq. (7.68). This is a model-based processing scheme which we summarize succinctly as follows:

Criterion: $\mathcal{J}(\underline{\Theta}(t)) = E\{e^2(t)\}$
Models:
  Signal: $s(t) = \underline{\Theta}'(t)\underline{Y}(t)$
  Measurement: $y(t) = s(t) + n(t)$
  Noise: $n(t)$ random
  Initial conditions: $\underline{a}(0)$, $\nabla_\Theta y(0)$, $\triangle_t$
Algorithm: $\underline{\Theta}(t+1) = \underline{\Theta}(t) + \triangle_t e(t)\nabla_\Theta y(t)$
Quality: $\mathcal{J}_{\text{mse}} = \frac{1}{N}\sum_{t=1}^{N} e^2(t)$

We summarize the algorithm in Table 7.5. Note that the instantaneous gradient estimate is estimated by using $\nabla_\Theta y(t)$, allowing $\{a_j(t)\}$ to vary with time. Thus it is crucial that the step-size $\triangle_t$ be selected very small, implying an overall time averaging of the gradient. An entire class of hyperstable algorithms, called the hyperstable adaptive recursive filter (*HARF*), has been developed. This class "filters" the prediction errors used in the update in the same way as the *RPEM* algorithm of Section 4.5 [1], [18].

### 7.4.2 All-Pole Adaptive Predictor

As a special case of this approach, consider the (all-pole) linear predictor. Recall the predictor signal model

$$\hat{s}(t) = -\sum_{i=1}^{N_a} a_i y(t-i)$$

**Table 7.5. LMS Adaptive IIR Algorithm**

Signal estimate

$$\hat{s}(t) = \sum_{i=1}^{N_a} a_i y(t-i) + \sum_{i=0}^{N_b} b_i u(t-i) = \underline{\Theta}(t)' \underline{X}$$

Error estimate

$$e(t) = s_d(t) - \hat{s}(t)$$

Gradient estimate

$$\nabla_\Theta y(t) = \underline{X}(t) + \sum_{j=1}^{N_a} a_j(t) \nabla_\Theta y(t-j)$$

Parameter iteration

$$\underline{\Theta}(t+1) = \underline{\Theta}(t) + \triangle_t e(t) \nabla_\Theta y(t)$$

Initial conditions

$$\underline{\Theta}(0), \quad \nabla_\Theta y(0)$$

where

$$\underline{X}(t) = [-y(t-1) \cdots - y(t-N_a) | u(t) \cdots u(t-N_b)]'$$
$$\underline{\Theta}(t) = [a_1(t) \cdots a_{N_a}(t) \mid b_0(t) \cdots b_{N_b}(t)]'$$

with the criterion

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

The gradient is easily obtained as

$$\nabla_{a_j} \mathcal{J} = 2E\{e(t)\underline{Y}(t)\}, \qquad j = 1, \cdots, N_a$$

where $\underline{Y}'(t) := [y(t-1) \cdots y(t-N_a)]$.

If we ignore the expectation operation, then the instantaneous gradient approximation leads to the corresponding *LMS* parameter estimator,

$$\underline{a}(t+1) = \underline{a}(t) - \triangle_t e(t) \underline{Y}(t) \tag{7.70}$$

with the prediction error given by

$$e(t) = y(t) - \hat{s}(t) = \sum_{i=0}^{N_a} a_i y(t-i) = \underline{Y}'(t) \underline{a} \tag{7.71}$$

**Table 7.6. All-Pole Adaptive (*LMS*) Algorithm**

For $t = 1, 2, \ldots$

Signal estimate

$$\hat{s}(t) = \sum_{i=1}^{N_a} a_i y(t-i)$$

Error estimate

$$e(t) = \sum_{i=0}^{N} a_i(t) y(t-i) = \underline{Y}'(t)\underline{a}(t)$$

Gradient estimate

$$\nabla_a \mathcal{J}(t) = 2e(t)\underline{Y}(t)$$

Parameter iteration

$$\underline{a}(t+1) = \underline{a}(t) - \tfrac{\triangle_t}{2}\nabla_a \mathcal{J}(t)$$

Initial conditions

$$\underline{a}(0), \quad \nabla_\Theta \mathcal{J}(0)$$

where

$$\underline{Y}'(t) = [y(t-1)\cdots y(t-N_a)]'$$

Substituting this result into Eq. (7.70), we obtain the parameter iterator

$$\underline{a}(t+1) = \underline{a}(t) - \triangle_t \underline{Y}(t)\underline{Y}'(t)\underline{a}(t) = (I - \triangle_t \underline{Y}(t)\underline{Y}'(t))\underline{a}(t) \tag{7.72}$$

leading to the state-space convergence analysis of the previous section. We summarize the all-pole algorithm as a *MBP*—with algorithmic details in Table 7.6.

| | |
|---|---|
| Criterion: | $\mathcal{J}(\underline{a}(t)) = E\{e^2(t)\}$ |
| Models: | |
| Signal: | $s(t) = \underline{a}'(t)\underline{Y}(t)$ |
| Measurement: | $y(t) = s(t) + n(t)$ |
| Noise: | $n(t)$ random |
| Initial conditions: | $\underline{a}(0), \triangle_t$ |
| Algorithm: | $\underline{a}(t+1) = \underline{a}(t) - \triangle_t e(t)\underline{Y}(t)$ |
| Quality: | $\mathcal{J}_{\text{mse}} = \frac{1}{N}\sum_{t=1}^{N} e^2(t)$ |

Before we close this section, let us consider an important application of the all-pole adaptive processor—instantaneous frequency estimation.

***Example 7.4*** Consider the problem of tracking a narrowband sinusoid in noise—a problem that occurs quite frequently in machine condition monitoring where a part (e.g., gear) is wearing. We would like to track the changes through its

associated frequency. Other applications that could also employ this approach are tracking a narrowband target for antisubmarine warfare or tracking a maneuvering target, producing a Doppler spectrum, or just monitoring the output of a power generator to assure it is producing power at a prescribed frequency (e.g., 60 Hz). In any of these applications the problem is identical—track the instantaneous frequency changes. This is essentially a time-varying problem. The adaptive predictor can be used to effectively solve this time-varying spectral estimation problem, since we know that the all-pole representation provides the maximum entropy spectral estimate ([19], [20], [21], [22]).

Consider a sinusoidal given by

$$s(t) = A\cos\Theta(t)$$

$\Theta(t)$ is the *instantaneous phase*, which is related to the corresponding *instantaneous frequency* by

$$\Omega(t) = \frac{\Theta(t) - \Theta(t-1)}{\triangle T} \approx \frac{d\theta(t)}{dt}$$

This is simply the derivative of the phase. For instance, the special case where $\Theta(t) = \omega_o t$ gives $\Omega(t) = \omega_o$. In *frequency modulation* the instantaneous frequency is taken proportional to the *message* signal as

$$\Omega(t) = \omega_o + m(t)$$

Using this relation and substituting the instantaneous phase relation, we obtain a recursion for the phase

$$\Theta(t) = \Theta(t-1) + \Omega_o\triangle T + m(t-1)\triangle T$$

For this application we have that the input to the adaptive processor is given by the measurement

$$y(t) = s(t) + n(t)$$

with $n(t)$ assumed zero-mean, gaussian with variance $\sigma_n^2$. The processor can be expressed in terms of a time-varying *AR* model (linear predictor) for the desired signal. That is, the predictor is

$$\hat{y}(t) = \hat{A}(q^{-1}, t)y(t) = -\sum_{i=1}^{N_a} \hat{a}_i(t)y(t-i)$$

where $\hat{A}(q^{-1}, t)$ is a time-varying polynomial in terms of the backward shift operator expressed with the instantaneous predictor coefficients, that is,

$$A(q^{-1}, t) = 1 + a_1(t)q^{-1} + \cdots + a_{N_a}q^{-N_a}$$

Once we have estimated the coefficients, we can obtain the instantaneous frequency from the roots of the coefficient polynomial at a given time instant

$$\Omega_i(t) = \text{ root } [1 - \hat{A}(q^{-1}, t)], \qquad i = 1, \ldots, N_a$$

The frequency can *also* be obtained in the Fourier domain from the normalized ($\sigma_e^2 = 1$) *instantaneous PSD* or *spectrogram* given by

$$S_{yy}(\Omega, t) = \left| \frac{1}{A(\Omega, t)} \right|^2 \qquad \text{for } A(\Omega, t) = 1 - a_1(t)e^{-j\Omega} + \cdots, -a_{N_a}(t)e^{-jN_a\Omega}$$

The frequencies are determined by estimating the peaks of the spectrum, that is,

$$\Omega_i(t) = \text{ peak } [S_{yy}(\Omega_i, t)], \qquad i = 1, \ldots, N_a$$

We simulate the process using the signal model and recursion for the instantaneous phase above with the following parameters. We chose $\triangle T = 1$ s, $f_o = 0.208$ Hz ($\Omega_o = 1.3$ rad/s), $A = 1$, and the variance of the measurement noise as $\sigma_n^2 = 1 \times 10^{-3}$ corresponding to an *SNR* of 2.8 dB. The random message or modulation signal was selected to be zero-mean, colored gaussian noise generated by low-pass filtering (fifth-order Butterworth with cutoff of 0.05 Hz) a $\mathcal{N}(0, 1)$ input. The simulated data and estimated frequency are shown in Figure 7.7. The instantaneous phase is shown in 7.7*a* with the noisy measurement data shown in Figure 7.7*c*. We used the *RNLMS* algorithm for the adaption algorithm with $N_a = 4$, $\alpha = 0.1$, $\beta = 10^{-4}$, $\gamma = 0.8$, and $\lambda = 0.99$. The peaks of the spectrogram were estimated to obtain the instantaneous frequency and compared to the "true" values in Figure 7.7*b*, demonstrating that the processor is capable of tracking slow changes. We summarize this problem as follows:

Criterion: $\mathcal{J}(\underline{a}(t)) = E\{e^2(t)\}$

Models:

Signal:

$$s(t) = A \cos \Theta(t)$$

$$\Theta(t) = \Theta(t-1) + \Omega_o \triangle T + m(t-1)\triangle T$$

$$\Omega(t) = \frac{\Theta(t) - \Theta(t-1)}{\triangle T}$$

Measurement: $y(t) = s(t) + n(t)$

Noise: $n(t)$ random

Initial conditions: $\underline{a}(0)$, $\triangle_t$

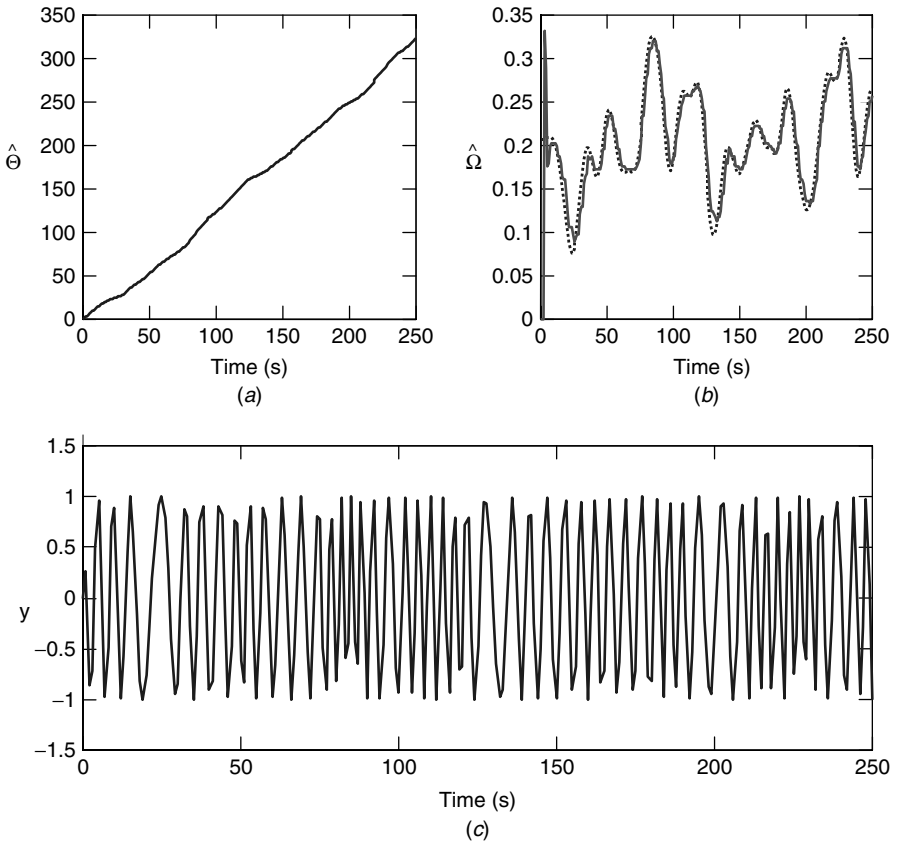



**Figure 7.7.** Instantaneous frequency estimation with adaptive predictor: (*a*) Instantaneous phase. (*b*) True (*dotted*) and instantaneous frequency estimate (*solid*). (*c*) Simulated noisy measurements (SNR = 2.8 dB).

Algorithm:

$$\underline{a}(t+1) = \underline{a}(t) + \triangle_t e(t)\underline{X}(t)$$

$$S_{yy}(\Omega, t) = \left| \frac{1}{A(\Omega, t)} \right|^2$$

$$\hat{\Omega}(t) = \text{ peak } [S_{yy}(\Omega_i, t)], \qquad i = 1, \ldots, N_a$$

Quality: $\mathcal{J}_{\text{mse}} = \dfrac{1}{N} \sum_{t=1}^{N} e^2(t)$

This completes the section on *IIR* adaptive processors. Next we consider another popular approach to solving this problem—the adaptive lattice processor.

## 7.5 LATTICE ADAPTIVE *MBP*

In this section we develop the lattice filter for adaptive signal processing. The lattice, by its inherent orthogonality properties discussed previously, offers many advantages over other techniques. It is numerically well behaved and enables the monitoring of stability during the calculation of each stage. The lattice model can be used to develop an estimator the minimizes either the forward prediction error, or backward prediction error, or both. The "forward-backward" lattice method calculates the forward and backward reflection coefficients individually and then combines them to form the desired estimates. The method suffers from the problem that the reflection coefficient may not be less than unity ([2], [10]). We choose the "Burg" approach developed previously which constrains the forward and backward reflection coefficients to be identical and less than unity. First we briefly review the Burg block processing method developed and then extend it to the adaptive case.

### 7.5.1 All-Pole Adaptive Lattice MBP

The underlying signal model set is *all-pole* given by

$$s(t) = -\sum_{i=1}^{N_a} a_i y(t-i) \tag{7.73}$$

with corresponding measurement characterized by

$$y(t) = s(t) + e(t) = -\sum_{i=1}^{N_a} a_i y(t-i) + e(t)$$

an *AR* model

$$A(q^{-1})y(t) = e(t) \qquad \text{for } q^{-1} \text{ the backward-shift operator}$$

This signal model can be obtained by transforming the estimated lattice parameters to the predictor coefficients. Therefore we start with the lattice recursion

$$\begin{aligned} e_f(t,i) &= e_f(t,i-1) - k(t,i)e_b(t-1,i-1) \\ e_b(t,i) &= e_b(t-1,i-1) - k(t,i)e_f(t,i-1) \end{aligned} \tag{7.74}$$

where $e_f(t,i)$, $e_b(t,i)$ and $k(t,i)$ are the respective forward and backward prediction errors, and reflection coefficient of the $i$th section at time $t$.

For the adaptive algorithm we begin with mean-squared error criterion and minimize it with respect to the reflection coefficient, that is,

$$\min_k \; \mathcal{J}(t,i) = E\{e_f^2(t,i) + e_b^2(t,i)\} \tag{7.75}$$

Performing the required minimization as before (see Section 4.4 for details), we obtain an expression for the gradient

$$\nabla_k \mathcal{J}(t,i) = -2E\{e_f(t,i)e_b(t-1,i-1) + e_b(t,i)e_f(t,i-1)\} \tag{7.76}$$

Alternatively, substituting the model recursions of Eq. (7.74) and combining, we obtain the gradient expression

$$\begin{aligned}\nabla_k \mathcal{J}(t,i) = -2E\{&2e_f(t,i-1)e_b(t-1,i-1) \\ &- k(t,i)\left(e_f^2(t,i-1) + e_b^2(t-1,i-1)\right)\}\end{aligned} \tag{7.77}$$

Setting this expression to zero and solving for $k(t,i)$, we obtain

$$k(t,i) = \frac{2E\{e_f(t,i-1)e_b(t-1,i-1)\}}{E\{e_f^2(t,i-1) + e_b^2(t-1,i-1)\}} \tag{7.78}$$

Again, we replace the ensemble averages with time averages. Anticipating the real-time adaptive algorithm, we can recursively estimate these statistics. Suppose that we estimate the denominator statistic with the exponentially weighted expression

$$V(t,i) = \frac{1}{t}\sum_{j=1}^{t} \lambda^{t-j}(e_f^2(j,i) + e_b^2(j-1,i)) \approx E\{e_f^2(t,i) + e_b^2(t-1,i)\}$$

which can be estimated recursively by removing the $t$th term from the sum and identifying the previous iterate $V(t-1,i)$ to obtain

$$V(t,i) = \lambda\left(\frac{t-1}{t}\right)\, V(t-1,i) + \frac{1}{t}(e_f^2(t,i) + e_b^2(t-1,i)) \tag{7.79}$$

Similarly for the numerator in Eq. (7.78), we have

$$\gamma(t,i) = \frac{1}{t}\sum_{j=1}^{t} \lambda^{t-j} e_f(t,i)e_b(t-1,i) \approx E\{e_f(t,i)e_b(t-1,i)\}$$

which leads to the recursion

$$\gamma(t,i) = \lambda\left(\frac{t-1}{t}\right)\gamma(t-1,i) + \frac{1}{t}e_f(t,i)e_b(t-1,i) \tag{7.80}$$

Note that using these recursions leads us to an alternative block processing method compared to the algorithm of Table 4.3. We summarize the algorithm in Table 7.7 for convenience. Note that for *each* lattice section, the *entire block* of data is processed and then a new section added and the block processed again. This type of processing is essentially deconvolving or removing the effects of each stage before the next one is added, owing to the orthogonality of the lattice processor.

**Table 7.7. Block Lattice Filter**

Statistics estimation

$$\gamma(t,i) = \lambda\left(\frac{t-1}{t}\right)\gamma(t-1,i) + \frac{1}{t}e_f(t,i)e_b(t-1,i)$$

$$V(t,i) = \lambda\left(\frac{t-1}{t}\right)V(t-1,i) + \frac{1}{t}(e_f^2(t,i) + e_b^2(t-1,i))$$

for $i = 1$ to $M$ (Order recursion)

for $t = 1$ to $N$ (Block recursion)

Reflection coefficient estimation

$$k(t,i) = \frac{2\gamma(t,i)}{V(t,i)}$$

Error estimation

$$e_f(t,i) = e_f(t,i-1) - k(t,i)e_b(t-1,i-1)$$

$$e_b(t,i) = e_b(t-1,i-1) - k(t,i)e_f(t,i-1)$$

Initial conditions

$$V(0,i),\ \gamma(0,i),\ \lambda$$

Convert lattice-to-predictor

$$\underline{k}(t) \longrightarrow \underline{a}(t)$$

Signal estimation

$$\hat{s}(t) = \underline{a}'(t)\underline{Y}(t)$$

With these recursions in mind, we can now begin the development of the adaptive algorithm based on the gradient approach. With the lattice model we must update the reflection coefficient iteratively, that is,

$$k(m+1,i) = k(m,i) + \triangle k(m,i)$$

Choosing a gradient-based adaption algorithm gives the correction as by

$$\triangle k(m,i) = \frac{-\triangle(m,i)}{2}\nabla_{k_i}\mathcal{J}(k(m,i))$$

Substituting the gradient for this process gives the corresponding parameter iteration

$$k(m+1,i) = k(m,i) - \frac{\triangle(m,i)}{2}\nabla_{k_i}\mathcal{J}(k(m,i))$$

Equivalently using the gradient of Eq. (7.77) obtains

$$\begin{aligned} k(m+1,i) = k(m,i) + \triangle(m,i)E\{&2e_f(m,i-1)e_b(m-1,i-1) \\ &- k(m,i)(e_f^2(m,i-1) + e_b^2(m-1,i-1))\} \end{aligned} \tag{7.81}$$

Replacing the ensemble statistic with its time-recursive counterparts of Eqs. (7.79) and (7.80), we obtain the parameter iterator with $t \to m$:

$$k(t+1,i) = k(t,i) + \triangle(t,i)[2\gamma(t,i) - k(t,i)V(t,i)] \tag{7.82}$$

Combining like terms, we obtain

$$k(t+1,i) = (1 - \triangle(t,i)V(t,i))k(t,i) + 2\triangle(t,i)\gamma(t,i)$$

Note that this can be written in state-space form, as before, by expanding over $i$, that is,

$$\underline{k}(t+1) = (I - \triangle(t,i)\mathbf{V}(t))\underline{k}(t) + 2\triangle(t,i)\underline{\gamma}(t)$$

where $\mathbf{V}(t) = \text{diag}[V(t,1)\cdots V(t,t)]$ and $\underline{\gamma}(t) = [\gamma(t,1)\cdots\gamma(t,t)]'$.

Tacitly assume stationarity for analysis purposes, and ignore $t$, then with, $\underline{k}_{\text{opt}} = 2\mathbf{V}^{-1}\underline{\gamma}$, we obtain

$$\underline{\tilde{k}}(t+1) = (I - \mathbf{V})\underline{\tilde{k}}(t) \tag{7.83}$$

where $\underline{\tilde{k}}(t) := \underline{k}(t) - k_{\text{opt}}$ and $\mathbf{V} := \text{diag}[\triangle(1)V(1)\cdots\triangle(t)V(t)]$. Solving this equation, as before, we have

$$\underline{\tilde{k}}(t) = (I - \mathbf{V})^t\underline{\tilde{k}}(0) \tag{7.84}$$

But because of the underlying lattice structure, the transition matrix is *already* diagonalized with each reflection coefficient converging at an identical rate governed by

$$\underline{\tilde{k}}(t,i) = (1 - \triangle(i)V(i))^t\underline{\tilde{k}}(0,i) \tag{7.85}$$

Thus the step-size must satisfy

$$|1 - \triangle(i)V(i)| < 1 \tag{7.86}$$

or the choice

$$\triangle(i) = \frac{\alpha}{V(i)} \qquad \text{for } 0 < \alpha < 1 \tag{7.87}$$

**Table 7.8. Stochastic Gradient Lattice Algorithm**

For $i = 1$ to $M$ (Order recursion)
For $t = 1$ to $N$ (Block recursion)

Statistics estimation

$$\gamma(t,i) = \lambda\left(\frac{t-1}{t}\right)\gamma(t-1,i) + \frac{1}{t}e_f(t,i)e_b(t-1,i)$$
$$V(t,i) = \lambda\left(\frac{t-1}{t}\right)V(t-1,i) + \frac{1}{t}(e_f^2(t,i) + e_b^2(t-1,i))$$

Gradient estimation

$$\nabla_k \mathcal{J}(t) = 2\gamma(t,i) - k(t,i)V(t,i)$$

Reflection coefficient iteration

$$k(t+1,i) = k(t,i) + \triangle(i)\nabla_{k_i}\mathcal{J}(t)$$

Error estimation

$$e_f(t+1,i) = e_f(t+1,i-1) - k(t+1,i)e_b(t,i-1)$$
$$e_b(t+1,i) = e_b(t,i-1) - k(t+1,i)e_f(t+1,i-1)$$

Initial conditions

$$V(0,i), \quad \gamma(0,i), \quad \lambda$$

Convert lattice to predictor

$$\underline{k}(t) \longrightarrow \underline{a}(t)$$

Signal estimation

$$\hat{s}(t) = \underline{a}'(t)\underline{Y}(t)$$

will satisfy this constraint. This relation implies that all of the reflection coefficients will converge at essentially the same rate. We summarize the stochastic gradient algorithm in Table 7.8. Note that in implementing the algorithm that all of the data is still processed through the recursion while each coefficient converges.

For real-time application we replace the stochastic gradient with the instantaneous gradient and develop the *LMS* algorithm for the lattice model. The instantaneous gradient is simply Eq. (7.76), if we ignore the expectation

$$\hat{\nabla}_{k_i}\mathcal{J} = -2\left[e_f(t,i)e_b(t-1,i-1) + e_b(t,i)e_f(t,i-1)\right] \tag{7.88}$$

Now replacing the gradient in Eq. (7.81) gives the parameter iteration

$$k(t+1,i) = k(t,i) + \triangle(t,i)[e_f(t,i)e_b(t-1,i-1) + e_b(t,i)e_f(t,i-1)] \tag{7.89}$$

With the step-size adjustment applied to satisfy the convergence constraints of Eq. (7.87), the *Gradient (LMS) Adaptive Lattice Algorithm* evolves with

$$\triangle(t,i) = \frac{\alpha}{V(t,i)} \tag{7.90}$$

and $V(t,i)$ satisfy the time recursion of Eq. (7.79) [22], [23]. We summarize this algorithm in Table 7.9 and the following model-based framework:

Criterion: $\mathcal{J}(\underline{k}(t)) = E\{e_f^2(t) + e_b^2(t)\}$

Models:

Signal: $s(t) = \underline{a}'(t)\underline{Y}(t)$

$e_f(t,i) = e_f(t,i-1) - k(t,i)e_b(t-1,i-1)$

$e_b(t,i) = e_b(t-1,i-1) - k(t,i)e_f(t,i-1)$

$\underline{k}(t) \longrightarrow \underline{a}(t)$

Measurement: $y(t) = s(t) + n(t)$

Noise: $n(t)$ random

Initial conditions: $\underline{a}(0)$, $\triangle(0,i)$

Algorithm: $k(t+1,i) = k(t,i) + \triangle(t,i)[e_f(t,i)e_b(t-1,i-1) + e_b(t,i)e_f(t,i-1)]$

Quality: $\mathcal{J}_{\text{mse}} = \dfrac{1}{N}\sum_{t=1}^{N} e^2(t)$

We note in passing that the exact methods exist using least-squares techniques to exactly estimate the gradient. These computations are more intensive, but offer superior performance [10], [24], [25] at a much higher computational cost. Consider the following example to illustrate the performance of the lattice algorithms.

***Example 7.5*** Suppose that we have data generated from a model given by the *ARMAX*(2, 1, 1, 1) model of the previous example and we apply the block lattice (Table 4.3) and gradient (*LMS*) adaptive lattice (Table 7.9). The results are shown in Figure 7.8. Note that the best we can hope for is that the adaptive results converge to the block solutions. A third-order model was selected with corresponding parameters:

**Table 7.9. Gradient (*LMS*) Adaptive Lattice Algorithm**

For $t = 1, 2, \ldots$
For $i = 1$ to $L$ (Order recursion)

Statistics estimation

$$V(t,i) = \lambda\left(\frac{t-1}{t}\right)V(t-1,i) + \frac{1}{t}e_f^2(t,i)e_b^2(t-1,i)$$

Gradient estimation

$$\nabla_k \mathcal{J}(t,i) = -2[e_f(t,i)e_b(t-1,i-1) + e_b(t,i)e_f(t,i-1)]$$

Step-size estimation

$$2\triangle(t,i) = \frac{\alpha}{V(t,i)}$$

Reflection coefficient iteration

$$k(t+1,i) = k(t,i) + \triangle(t,i)\nabla_{k_i}\mathcal{J}(t,i)$$

Error estimation

$$e_f(t+1,i) = e_f(t+1,i-1) - k(t+1,i)e_b(t,i-1)$$
$$e_b(t+1,i) = e_b(t,i-1) - k(t+1,i)e_f(t+1,i-1)$$

Initial conditions

$$V(0,i), \quad \lambda, \quad \alpha$$

Convert lattice-to-predictor

$$\underline{k}(t) \longrightarrow \underline{a}(t)$$

Signal estimation

$$\hat{s}(t) = \underline{a}'(t)\underline{Y}(t)$$

| Parameter | True Value | Block Estimate | Adaptive Estimate |
|---|---|---|---|
| $a_1$ | $-1.5$ | $-1.47$ | $-1.53$ |
| $a_2$ | 0.7 | 0.73 | 0.84 |
| $a_3$ | — | $-0.10$ | $-0.16$ |
| $b_0$ | 1.0 | 0.155 | 0.133 |
| $b_1$ | 0.5 | — | — |
| $c_1$ | 0.3 | — | — |
| $d_1$ | 0.5 | — | — |

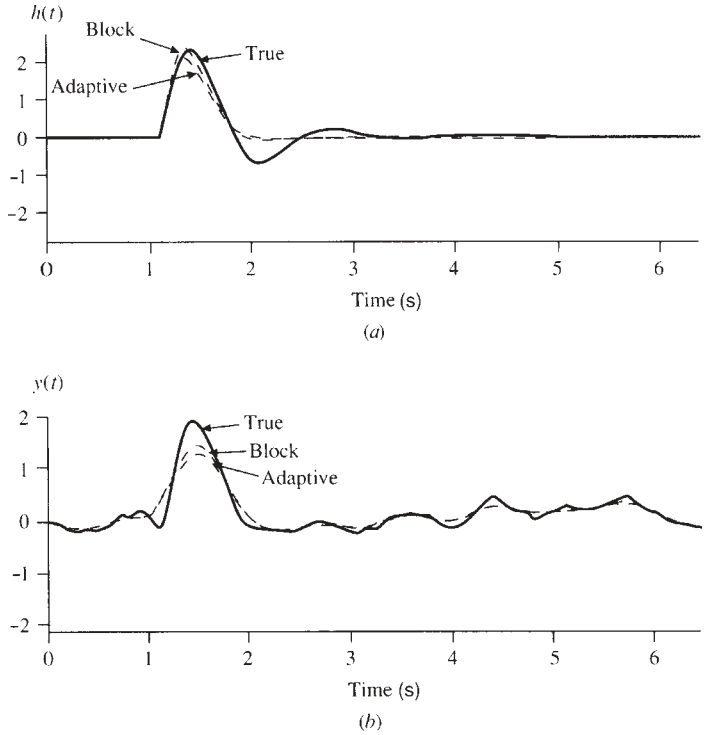



**Figure 7.8.** Block and adaptive lattice filter design for *ARMAX*(2, 1, 1, 1) simulation. *(a)* True and estimated impulse response. *(b)* Actual and filtered measurements (signals).

Using *MATLAB* [16] the simulation results are shown in Figure 7.8. The estimated signal and filtered measurements are shown for both block and adaptive processors. Note that the adaptive results are not quite as good as the block. This completes the example.

### 7.5.2 Joint Adaptive Lattice Processor

Before we close this section, let us consider development of the adaptive lattice filter for the *FIR* (Wiener) or joint process estimator case.

Basically the idea behind the *FIR* adaptive lattice filters is to utilize the lattice model to transform correlated measurement data $\{x(t)\}$ to the uncorrelated backward prediction errors $\{e_b(t)\}$ of the lattice and then solve the standard Wiener problem. In the batch problem of Section 4.2 we transformed the correlated measurements to the uncorrelated innovations.

The underlying signal model is an all-zero (*FIR*) structure given by

$$s(t) = \sum_{i=1}^{N_\ell - 1} \ell(i) e_b(t-i) = \underline{\ell}' \underline{e}_b \tag{7.91}$$

As before, we would like to minimize the criterion

$$\mathcal{J}(t) = E\{e^2(t)\} = E\{s_d^2(t)\} - 2\underline{\ell}' E\{s_d(t)\underline{e}_b\} + \underline{\ell}' E\{\underline{e}_b \underline{e}_b'\}\underline{\ell}$$

where $e(t) := s_d(t) - \hat{s}(t)$. Replacing these operations with covariances, we have

$$\mathcal{J}(t) = E\{s_d^2(t)\} - 2\underline{\ell}' \underline{R}_{se_b} + \underline{\ell}' \mathbf{R}_{e_b e_b} \underline{\ell} \tag{7.92}$$

Following the procedure of Section 7.3, we obtain the gradient

$$\nabla_\ell \mathcal{J}(\underline{\ell}) = E\{e(t)\underline{e}_b\} = -2\underline{R}_{se_b} + 2\mathbf{R}_{e_b e_b}\underline{\ell} \tag{7.93}$$

which yields the equivalent Wiener solution

$$\underline{\ell}_{\text{opt}} = \mathbf{R}_{e_b e_b}^{-1} \underline{R}_{se_b} = (\mathbf{L}^{-1}\mathbf{R}_{xx}^{-1}\mathbf{L}^{-T})(\mathbf{L}^T \underline{R}_{sx}) \tag{7.94}$$

for $\mathbf{L}$ the linear transformation matrix taking $\underline{X} \rightarrow \underline{e}_b$ and $^{-T}$ the inverse transposition operation. Note that the $\mathbf{R}_{e_b e_b}$ is diagonal with entries $r_{e_b}(i) = E\{e_b^2(i)\}$ (see Haykin [2] for more details). Clearly, the same analysis follows as in the previous cases (see Section 7.3) and we obtain a stochastic gradient algorithm identical to that given in Table 7.1 with $X \rightarrow e_b$. The stochastic gradient parameter iteration simplifies considerably as

$$\underline{\ell}(i+1) = \underline{\ell}(i) - \triangle(i)(-2\underline{R}_{se_b} + 2\mathbf{R}_{e_b e_b}\underline{\ell}(i)) \tag{7.95}$$

$R_{e_b e_b} = \text{diag}[r_{e_b}(1) \cdots r_{e_b}(N_\ell)]$, which decouples the iteration into its components. That is, Eq. (7.95) becomes

$$\ell_j(i+1) = \ell_j(i) - \triangle_j(i)(-2r_{se}(j) + 2r_{e_b}(j)\ell_j(i)) \qquad \text{for } j = 1, \ldots, N_\ell \tag{7.96}$$

Using the same analysis as the previous case, the step-size must be selected as

$$2\triangle_j(i) = \frac{\alpha}{r_{e_b}(j)}, \qquad 0 < \alpha < 1$$

which again shows that all lattice weights converge at the same rate.

The component-wise instantaneous gradient approximation of Eq. (7.93) is given by

$$\hat{\nabla}_{\ell_i} \mathcal{J}(\underline{\ell}) = -2e(t)e_b(t,i) \qquad \text{for } i = 1, \ldots, N_\ell \tag{7.97}$$

The parameter iteration becomes

$$\ell_i(t+1) = \ell_i(t) + \frac{\alpha}{\nu(t,i)} e(t) e_b(t,i) \tag{7.98}$$

where $\nu(t,i)$ is given by the time recursion

$$\nu(t,i) = \lambda\left(\frac{t-1}{t}\right)\nu(t-1,i) + \frac{1}{t}e_b^2(t,i) \tag{7.99}$$

We summarize the *FIR* lattice in terms of the model-based framework:

Criterion: $\mathcal{J}(\underline{h}(t)) = E\{e_f^2(t) + e_b^2(t)\}$

Models:

Signal:
$$s(t) = \underline{\ell}'(t)\underline{Y}(t)$$
$$e(t) = s_d(t) - \hat{s}(t)$$
$$e_f(t,i) = e_f(t,i-1) - k(t,i)e_b(t-1,i-1)$$
$$e_b(t,i) = e_b(t-1,i-1) - k(t,i)e_f(t,i-1)$$

Measurement: $y(t) = s(t) + n(t)$

Noise: $n(t)$ random

Initial conditions: $\underline{a}(0), \nabla_\Theta y(0), \triangle_t$

Joint algorithm:
$$k(t+1,i) = k(t,i) + \frac{\alpha}{V(t,i)}[e_f(t,i)e_b(t-1,i-1) + e_b(t,i)e_f(t,i-1)]$$
$$\ell_i(t+1) = \ell_i(t) + \frac{\alpha}{v(t,i)}e(t)e_b(t,i)$$

Quality: $\mathcal{J}_{\text{mse}} = \dfrac{1}{N}\displaystyle\sum_{t=1}^{N} e_f^2(t) + e_b^2(t)$

The complete algorithm includes the lattice recursion as well as the reflection coefficient iteration. We summarize the *FIR* gradient adaptive lattice algorithm in Table 7.10. This completes the section on adaptive lattice filters. Next we consider applications of adaptive signal-processing techniques to the solution of various application problems.

## 7.6 ADAPTIVE *MBP* APPLICATIONS

In this section we briefly discuss some of the popular applications of adaptive signal processing techniques. We have seen in the previous sections that adaption algorithms coupled with a signal model provide an approximate solution to the Wiener filter design problem for stationary as well as nonstationary processes. Here we consider the construction of various adaptive signal processing techniques that incorporate the adaptive algorithms as an integral part of their structure. Specifically, we consider the adaptive noise canceler [1] and its variants, the adaptive predictor, and line enhancer.

### 7.6.1 Adaptive Noise Canceler *MBP*

In its simplest form a noise canceler [1] is an optimal Wiener filter that can be characterized by the usual measurement model

$$y(t) = s(t) + n(t) \tag{7.100}$$

**Table 7.10. Joint Gradient (*LMS*) Adaptive Lattice Algorithm**

For $t = 1, 2, \ldots$
For $i = 1 \ \ldots \ N_\ell$ (Order recursion)

*Adaptive Lattice*

Statistics estimation

$$V(t,i) = \lambda\left(\frac{t-1}{t}\right)V(t-1,i) + \frac{1}{t}e_f^2(t,i)e_b^2(t-1,i)$$

Reflection coefficient iteration

$$k(t+1,i) = k(t,i) + \frac{\alpha}{V(t,i)}[e_f(t,i)e_b(t-1,i-1) + e_b(t,i)e_f(t,i-1)]$$

Error estimation (lattice recursion)

$$e_f(t+1,i) = e_f(t+1,i-1) - k(t+1,i)e_b(t,i-1)$$
$$e_b(t+1,i) = e_b(t,i-1) - k(t+1,i)e_f(t+1,i-1)$$

*FIR Recursion*

Statistics estimation

$$\nu(t,i) = \lambda\left(\frac{t-1}{t}\right)\nu(t-1,i) + \frac{1}{t}e_b^2(t,i)$$

Error estimation

$$e(t) = s_d(t) - \hat{s}(t)$$

Parameter iteration

$$\ell_i(t+1) = \ell_i(t) + \frac{\alpha}{\nu(t,i)}e(t)e_b(t,i)$$

Signal estimation

$$\hat{s}(t+1) = \sum_{i=0}^{N_\ell - 1} \ell_i(t+1)e_b(t+1-i,i)$$

Initial conditions

$$V(0,i), \quad \nu(0,i), \quad \lambda, \ \alpha$$

where $s(t)$, $n(t)$ are the respective signal and random noise sequences. The canceler assumes that there exists a *reference* noise signal, $r(t)$, that is correlated to the measurement noise. The *key* to noise cancelation is this reference, and it *must* be available is some form (e.g., delayed measurement) for implementation. The reference is assumed related (correlated) to the measurement noise through some unknown linear system characterized by its impulse response $h(t)$, that is,

$$r(t) = h(t) * n(t) = H(q^{-1})n(t)$$

If we further assume that $H(q^{-1})$ is invertible, then we have

$$n(t) = H_n(q^{-1})r(t) = \sum_{i=0}^{N_h-1} h(i)r(t-i) = \underline{h}_n'\underline{r}(t) \tag{7.101}$$

where $H_n(q^{-1}) = H^{-1}(q^{-1})$ and $\underline{r}'(t) = [r(t)\cdots r(t-N_h+1)]$. Substituting this relation for $n(t)$ into the measurement model, we obtain

$$y(t) = s(t) + H_n(q^{-1})r(t) \tag{7.102}$$

which is an input–output model relating the reference noise (input) to the measurement (output). The noise canceler is shown in Figure 7.9 (ignoring the dotted line). Here we see that the signal can be extracted from the noisy measurement using the canceler output as

$$e(t) = y(t) - \hat{n}(t) \approx \hat{s}(t) \tag{7.103}$$

where $\hat{n}(t) = \hat{H}_n(q^{-1})r(t)$. Alternatively, substituting and using Eq. (7.102), we have

$$\hat{s}(t) = s(t) + [H_n(q^{-1}) - \hat{H}_n(q^{-1})]r(t) \tag{7.104}$$

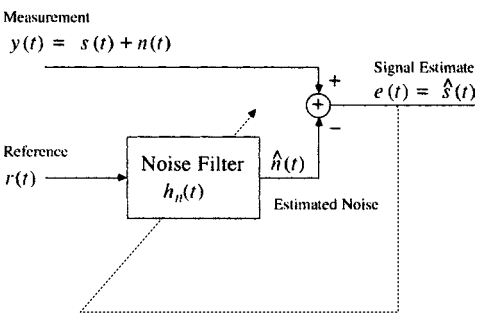



**Figure 7.9.** Noise canceler diagram. *(a)* Stationary case (*solid lines only*). *(b)* Adaptive (quasi-stationary) case (*solid plus dashed lines*).

Thus, when the canceler converges, we have

$$\hat{H}_n(q^{-1}) \rightarrow H_n(q^{-1}) \quad \text{and} \quad \hat{s}(t) \rightarrow s(t)$$

demonstrating that the canceler removes that part of the measurement *correlated* with the reference input. The optimal solution to this problem is obtained by

$$\min_{\underline{h}} \mathcal{J}(t) = E\{e^2(t)\}$$

where $e(t) = y(t) - \hat{n}(t) = \hat{s}(t)$. Performing this minimization, we obtain the gradient as in Section 7.3 as

$$\nabla_h \mathcal{J}(t) = 2E\{e(t)\nabla_h e(t)\} = -2E\{e(t)\underline{r}(t)\} = -2\underline{R}_{yr} + 2\mathbf{R}_{rr}\underline{h}_n \tag{7.105}$$

Setting this expression to zero and solving, we obtain the "noise filter"

$$\underline{\hat{h}}_n = \mathbf{R}_{rr}^{-1}\underline{R}_{yr} \tag{7.106}$$

which leads to an optimal estimate of the noise

$$\hat{n}_{\text{opt}}(t) = \underline{\hat{h}}_n'\underline{r}(t) \tag{7.107}$$

Therefore the signal is

$$\hat{s}_{\text{opt}}(t) = y(t) - \hat{n}_{\text{opt}}(t) \tag{7.108}$$

Note that the noise canceler consists of a two-step procedure:

1. Obtaining the optimal estimate of the noise filter, $\hat{h}_n(t)$, and corresponding noise, $\hat{n}_{\text{opt}}(t)$).
2. Canceling or subtracting the estimated noise to obtain the desired signal, $\hat{s}(t) = y(t) - \hat{n}(t)$.

We see from this construction that the function of the noise filter is to "pass" that noise-correlated part of the reference and remove it from the measurement to produce a signal estimate. If the processes under investigation are stationary, then the canceler can be designed with any of the parametric (joint process) methods of Chapter 4. However, if the processes are nonstationary, then the adaptive techniques of this chapter with $y(t) \rightarrow s_d(t)$, $h_n(t) \rightarrow h(t)$, $r(t) \rightarrow x(t)$ can be applied (see Figure 7.9).

Using the expression of Eq. (7.105), we obtain the stochastic gradient iteration

$$\underline{h}_n(i+1) = \underline{h}_n(i) - \frac{\triangle_i}{2}(-2\underline{R}_{yr} + 2\mathbf{R}_{rr}\underline{h}_n(i)) \tag{7.109}$$

Alternatively, using the instantaneous gradient estimate, we obtain the *LMS* iteration for *adaptive noise canceling* as

$$\underline{h}_n(t+1) = \underline{h}_n(t) + \triangle_t e(t)\underline{r}(t) \tag{7.110}$$

Convergence results are identical with $y(t)$, $r(t)$, $h_n(t)$ and $\hat{n}(t)$ replacing $s_d(t)$, $x(t)$, $h(t)$, and $\hat{s}(t)$, respectively of Table 7.10. We summarize the model-based canceling approach and its implementation using the *NLMS* algorithm in Table 7.11.

Criterion: $\mathcal{J}(\underline{h}(t)) = E\{e^2(t)\}$

Models:

Signal: $e(t) = y(t) - \hat{n}(t)$

Measurement: $y(t) = s(t) + v(t)$

**Table 7.11. Normalized *LMS* Noise Canceling Algorithm**

For $t = 1, 2, \ldots$

Noise estimation

$$\hat{n}(t) = \sum_{k=0}^{N_h-1} h_n(k)r(t-k) = \underline{h}_n'\underline{r}(t)$$

Error estimate

$$e(t) = y(t) - \hat{n}(t)$$

Step-size update

$$\hat{V}_{rr}(t) = \gamma \hat{V}_{rr}(t-1) + (1-\gamma)r^2(t)$$

$$\triangle_t = \frac{\alpha}{\hat{V}_{rr}(t) + \beta}$$

Parameter iteration

$$\underline{h}_n(t+1) = \underline{h}_n(t) + \triangle_t e(t)\underline{r}(t)$$

Signal estimate

$$\hat{s}(t) = y(t) - \hat{n}(t) = e(t)$$

Initial conditions

$$\underline{h}_n(0), \quad \hat{V}_{rr}(0), \quad \alpha, \quad \beta, \quad \gamma$$

| | |
|---|---|
| Noise: | $\hat{n}(t) = \underline{h}_n' \underline{Y}(t), r(t)$ reference |
| Initial conditions: | $\underline{h}_n(0), \triangle_t$ |
| Algorithm: | $\underline{h}(t+1) = \underline{h}(t) + \triangle_t e(t)\underline{r}(t)$ |
| Quality: | $\mathcal{J}_{\text{mse}} = \frac{1}{N}\sum_{t=1}^{N} e^2(t)$ |

We also note in closing that the noise-canceling filter design can also be considered a system identification problem as noted in [5], since $\{r(t)\}$ and $\{y(t)\}$ are the respective input and output sequences and $\{\hat{h}(t)\}$ the "identified" system. This is the approach taken in [17], [19].

Consider the following example of the noise canceling.

***Example 7.6*** Suppose that we have a triangular pulse contaminated with a harmonic disturbance at 10 Hz and random noise, that is,

$$y(t) = s(t) + n(t)$$

where $n(t) = \sin 2\pi(10)t + e(t)$ for $e$ uniform with variance 0.001. We would like to design a noise-canceling filter to pass the desired signal (triangular pulse) and eliminate the noise. We have a reference measurement of the harmonic disturbance. We design the stationary (optimal Wiener) and the results are shown in Figure 7.10. Here we see the signal and noise and corresponding spectrum. Note the low-pass Fourier spectrum (triangular pulse), harmonic disturbance at 10 Hz, and random noise. The optimal canceling *FIR* filter design is shown in Figure 7.10*b*. From the corresponding spectrum we observe that the filter will pass the sinusoid unattenuated to produce the estimated noise. The canceled signal and spectrum are shown in Figure 7.10*c*. Note the triangular pulse evolves including the random noise and the sinusoidal disturbance at 10 Hz has been removed as shown in the spectrum. This completes the example.

### 7.6.2 Adaptive D-Step Predictor *MBP*

Next let us consider a special case of the noise canceling filter—the adaptive predictor [26], [27]. Suppose that we have a reference signal that is a delayed version of the primary signal, that is,

$$r(t) = s(t-d)$$

or equivalently

$$r(t+d) = s(t) \tag{7.111}$$

Substituting into the gradient expression of Eq. (7.105) for the canceler with $y(t) = s(t)$, we obtain

$$\nabla_h \mathcal{J}(t) = -2E\{e(t)\underline{r}(t+d)\} = -2\underline{R}_{yr}(d) + 2\mathbf{R}_{rr}\underline{h}$$

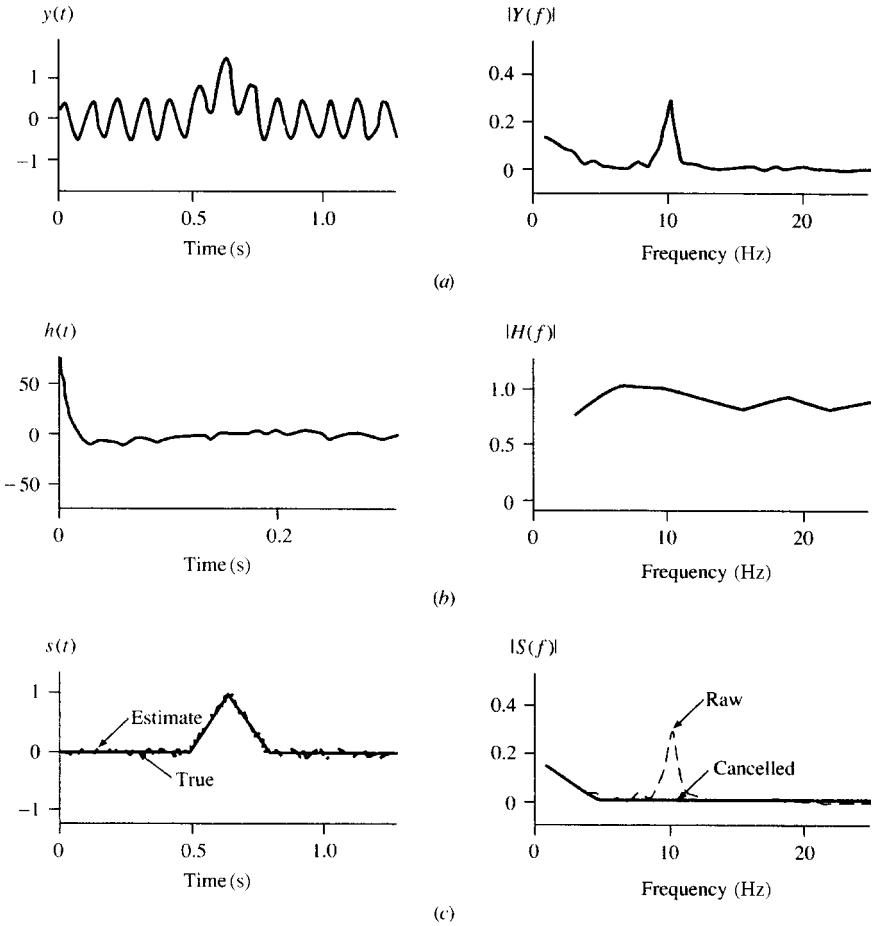



**Figure 7.10.** Noise-canceling filter design for sinusoid signal. *(a)* Measured signal and spectrum. *(b)* Canceling filter impulse response and spectrum. *(c)* Signal estimates (stationary) and spectrum.

which leads to the optimal *D-step predictor.*

$$\underline{\hat{h}}_{\text{opt}} = \mathbf{R}_{rr}^{-1}\underline{R}_{yr}(d) \tag{7.112}$$

The $D$-step predictor is shown in Figure 7.11.

The predictor implementation has two outputs, which are used in various applications:

1. Uncorrelated channel
2. Correlated channel

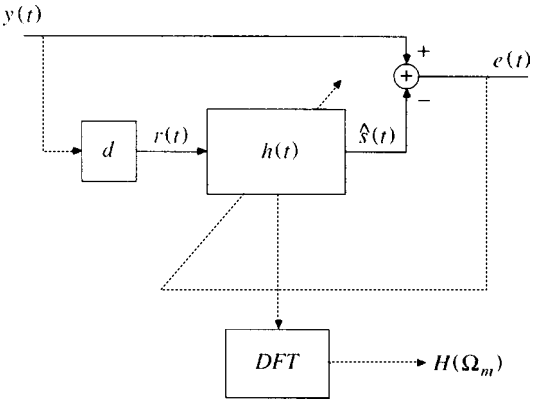



**Figure 7.11.** *D*-Step predictor (noise canceller) diagram. *(a)* Stationary case (*solid lines only*). *(b)* Adaptive (quasi-stationary) case (*solid and dotted lines*).

By selecting the prediction distance, $d$, we can control that part of the signal available at the error or uncorrelated output channel ($e(t)$), since the predictor is in fact a "correlation canceler." The correlated channel ($\hat{s}(t)$) consists of that portion of the signal that is strongly autocorrelated at lag "$d$" or greater and is therefore the output of the predictor. This configuration can also be used to enhance spectral lines as indicated by the dotted lines in Figure 7.11. Here we see that taking the *DFT* of the canceling filter impulse response leads to an enhanced line spectrum [26].

Clearly, the predictor can be implemented adaptively using the *NLMS* algorithm of Table 7.11. The predictor can be used to separate a signal that consists of both narrowband and wideband components, depending on the choice of the *delay* or *prediction distance*, $d$. We summarize the model-based predictor as follows:

| | |
|---|---|
| Criterion: | $\mathcal{J}(\underline{h}(t)) = E\{e^2(t)\}$ |
| Models: | |
| Signal: | $e(t) = y(t) - \hat{s}(t)$ |
| Measurement: | $y(t) = s(t) + n(t)$ |
| Noise: | $\hat{s}(t) = \underline{h}'\underline{Y}(t+d)$ |
| | $y(t+d)$ reference |
| | $n(t)$ random |
| Initial conditions: | $\underline{h}_n(0), \triangle_t, d$ |
| Algorithm: | $\underline{h}(t+1) = \underline{h}(t) + \triangle_t e(t)\underline{r}(t)$ |
| Quality: | $\mathcal{J}_{\text{mse}} = \frac{1}{N}\sum_{t=1}^{N} e^2(t)$ |

Consider the following example to illustrate the operation of the processor.

***Example 7.7*** Suppose that we have a signal that is composed of two discrete uncorrelated exponentials such that

$$y(t) = \alpha^t + \beta^t, \qquad 0 < \alpha < \beta < 1$$

We would like to design a predictor to separate the signals for both stationary as well as quasi-stationary cases using the Wiener solution of Eq. (7.112) and the adaptive *LMS* algorithm. The correlation function of this process can be approximated by the components as

$$R_{yy}(k) = \alpha^{|k|} + \beta^{|k|}, \qquad 0 < \alpha < \beta < 1$$

where the broadband signal correlation time,[3], $\tau_\alpha$ decays much more rapidly than the narrowband, $\tau_\beta$. If we select the prediction distance according to

$$\tau_\alpha < d \leq \tau_\beta$$

then $d$ will be longer than $\tau_\alpha$ of the broadband. Therefore the reference signal will be completely uncorrelated with the broadband component, that is, (assuming $\alpha$ is uncorrelated with $\beta$)

$$y(t) = s_\alpha(t) + s_\beta(t)$$

and

$$r(t) = s_\alpha(t-d) + s_\beta(t-d) \qquad \text{for } \tau_\alpha < d < \tau_\beta$$

$$E\{y(t)r(t)\} = E\{s_\alpha(t)s_\alpha(t-d)\} + E\{s_\beta(t)s_\beta(t-d)\}$$

or

$$E\{y(t)r(t)\} = R_{s_\alpha s_\alpha}(d) + R_{s_\beta s_\beta}(d) = R_{s_\alpha s_\alpha}(d) \qquad \text{for } \tau_\alpha < d$$

Thus the predictor will produce an estimate of $\hat{s}_\beta(t)$ and remove it from the output, that is

$$\hat{n}(t) \approx s_\beta(t)$$

and, of course,

$$\hat{s}(t) = e(t) = y(t) - \hat{n}(t) = s_\alpha(t)$$

Suppose that we select $\alpha = 0.5$ and $\beta = 0.9$, which implies that $\tau_\alpha = 1.5$ and $1.5 < d$. We simulate the data using *MATLAB* [16]. The results are shown in Figure 7.12. Here we see the signal and individual functions and the outputs of the predictor, both narrow- and broadband signals. In Figure 7.12*a* we see the simulated transient data and corresponding correlation function with autocorrelations for both $\alpha$ and $\beta$, while Figure 7.12*b* shows the separated or extracted transient signal estimates and true signals for comparison. Clearly, the predictor has been able to separate and estimate the signals.

[3]Correlation time is roughly the time it takes for the correlation function to decay to zero.

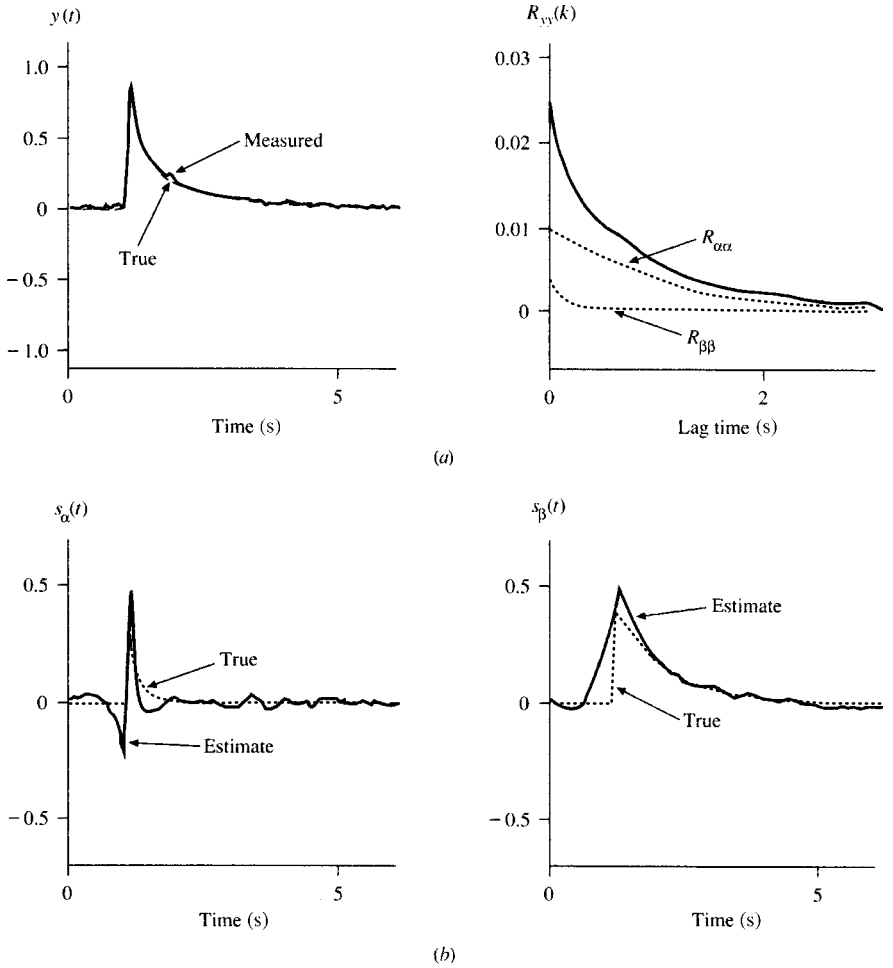



**Figure 7.12.** *D*-Step predictor simulation. *(a)* Simulated data composite (*solid*) and signal covariances. *(b)* Estimated and true (*dashed*) broadband and narrowband signals.

### 7.6.3 Adaptive Harmonic *MBP*

The adaptive filter has proved to be an extremely versatile processor and has been successfully applied in many useful applications. Here we present a few more applications to again further exemplify its performance for some very important signal-processing problems. In the following example, we use the adaptive D-step predictor structure, which can be considered (simply) a noise canceler *without* a reference signal available. The reference signal holds the key to the canceler operation, but it also presents a burden to the signal processor who must first determine a meaningful reference signal and then perform an additional (in most

cases) measurement to acquire it. In some situations a reference channel evolves quite naturally, although it is not explicitly intended. For example, consider a ship towing an array of hydrophone sensors. It is well known that the proximity of the ship to the array induces noisy measurements because its engines create strong narrow- and broadband interference signals, contaminating the data and masking the true target. One approach to alleviate this problem is to use the closest hydrophone sensor as a reference signal to cancel the ship interference. We will discuss another transient application in the case study to follow where the same sensor is used for both signal and reference (Section 7.7).

In any case, let us suppose a reference channel is not available and we must extract a harmonic (sinusoidal) signal from its noisy broadband background. Suppose further that the harmonic signal is slowly changing in amplitude and phase. An adaptive processor can provide a solution to this problem. It is important to realize that this form of the noise canceling structure has been given a wide variety of names depending on the particular application. For instance, in the example to follow, the D-step adaptive predictor is termed an *adaptive line enhancer* (*ALE*) as in Figure 7.11, since its task is to extract a spectral line (sinusoid) or delta function in the frequency domain from noisy broadband data [2], [26], [27], [28]. It is also called a *self-tuning filter*, since it adaptively adjusts its weights to the slowly changing signal parameters.

Recall that the key to successful noise cancellation is the reference signal, $r(t)$, while for prediction it is the *delay* or *prediction horizon*, $d$. The same basic idea is employed in both processors: "decorrelate" or "separate" the signal from the noise by choosing $d$ appropriately as we demonstrated in the previous example. If we have the measurement

$$y(t) = s(t) + n(t) \tag{7.113}$$

where $s(t)$ is the harmonic signal given by

$$s(t) = \sum_{i=1}^{N_s} A_i(t) \sin(\Omega_i(t) + \phi_i(t)) \tag{7.114}$$

for $A_i$, $\Omega_i$, $\phi_i$ the respective (varying) amplitude, frequency and phase parameters, then the basic problem[4] to be solved is as follows:

> **GIVEN** the set of noisy measurement data, $\{y(t)\}$, $t = 0, \ldots, N-1$. **FIND** the best estimate of the harmonic signal, $\hat{s}(t)$, characterized by the set of parameters, $\{A_i(t), \Omega_i(t), \phi_i(t)\}$; $i = 1, \ldots, N_s$.

[4]This is an extremely important problem in passive sonar processing where the weak target is modeled in this manner. Note also, from our previous discussion of wave-based *MBP* that this problem is identical to the spatial estimation problem for direction of arrival (*DOA*), that is, estimating plane waves (spatial temporal sinusoids) in noise (see Chapter 4 for more details). In the spatial case these processors are placed in the general class of adaptive arrays [1], [2], [10], [12].

If we calculate the covariance of the harmonic signal (as in Chapter 2), then the resulting function will give us some insight into the operation of the adaptive processor. Recall that the harmonic covariance for constant model parameters is given by

$$R_{yy}(k) = \sum_{i=1}^{N_s} \frac{A_i^2}{2} \cos(k\Omega_i \triangle T) + \sigma_{nn}^2 \delta(k) \tag{7.115}$$

The optimal solution for the predictor weights of Eq. (7.112) implies that its components for a single sinusoid will take the form

$$\underline{R}_{yr}(d) \rightarrow \frac{A_i^2}{2} \cos\left((k+d)\Omega_i \triangle T\right) \qquad \text{for } k = 0, \ldots, N_h - 1 \tag{7.116}$$

Comparing this relation to an assumed sinusoidal covariance model of Eq. (7.115) implies that a set of simultaneous equations can be solved such that the prediction horizon $d$ is directly related to the phase and frequency of the unknown sinusoid [28], that is,

$$d = \frac{\phi}{\Omega_i \triangle T} \tag{7.117}$$

The correct value of $d$ will produce a coherent (correlated) signal estimate, since it is a function of phase, while decorrelating the noise at its error output. To see this idea, consider the following example of both harmonic and noise-canceling processors.

***Example 7.8*** Suppose that we have a sonar system that we deploy to extract a target signature for eventual localization and tracking. We use a helicopter to process the data as well as a suite of surface ships as depicted in Figure 7.13. Assume that this signal is a 50 Hz, unit amplitude, spectral line residing in broadband ambient noise characterized by a bandpass spectrum between 30 and 70 Hz. For comparative purposes we generate another independent broadband noise source measured from another sensor far from the target, but residing within the same noise field. From our helicopter-based sonar we assume that this "noise-only" channel is *not* available but is available to the accompanying surface ships. We will apply the ($D$-step predictor) *ALE* aboard the helicopter and a noise canceler aboard the surface ship to extract the target signature. The measurement data (256 samples) were simulated using *MATLAB* with a sampling interval of $\triangle T = 20$ ms at a $-25$ dB *SNR* using unit variance, bandpass gaussian noise and the 50 Hz target signal. The data are shown in Figure 7.14*a* along with its corresponding power spectrum. It is clear that the signal line, although visible, is obscured in the broadband noise. We choose a 16-weight ($N_h = 16$) *FIR* design for the *ALE* using a fixed step-size of $\triangle_t = 0.1$ and the *NLMS* adaptive processor. The results are as expected. Selecting a prediction horizon of $d = 3$ samples or 60 ms, the *ALE* is

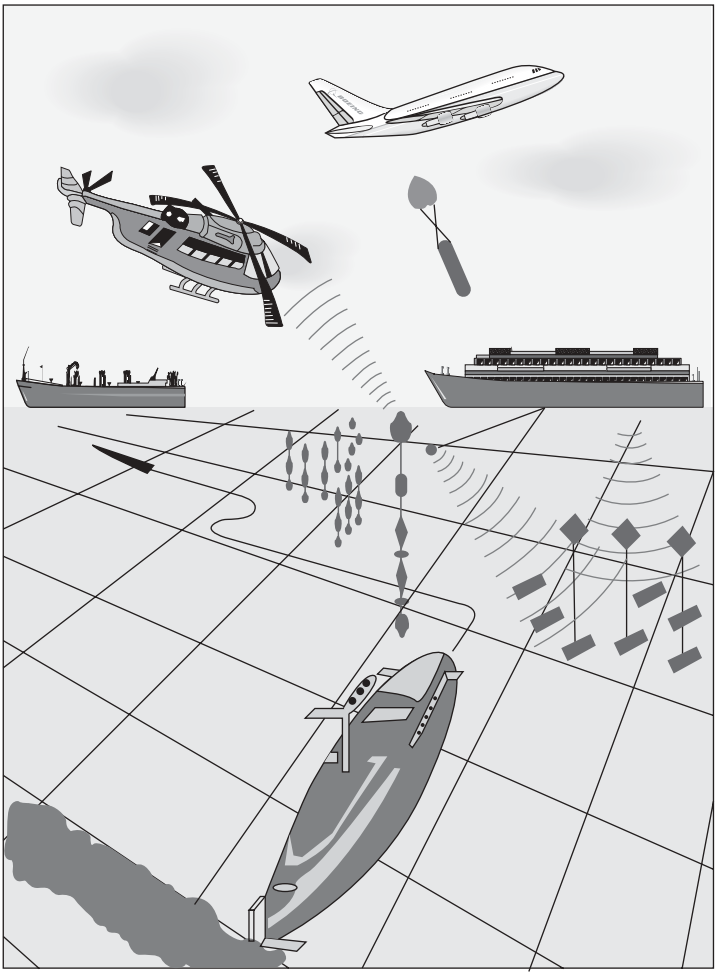


**Figure 7.13.** Sonar simulation scenario using helicopters, aircraft, and surface ships to track and localize a target submarine.

able to extract the 50 Hz line. The results are shown respectively in Figure 7.14*b* corresponding to the helicopter and in Figure 7.14*c* corresponding to the surface ship. The *ALE* suppresses (reasonably) the bandpass noise, enhancing the spectral line at 50 Hz indicating to the helicopter search team that a target is present as observed in Figure 7.14*b*. The surface ship confirms the target much more emphatically as seen by its cancelled spectrum in Figure 7.14*c* where the majority of the broadband noise is suppressed (dotted line) with the spectrum dominated by the target line at 50 Hz (solid line). These results occur because the surface ship is able to use the *NLMS* noise-canceling algorithm, since the broadband reference channel is available. The canceler clearly produces superior results compared to

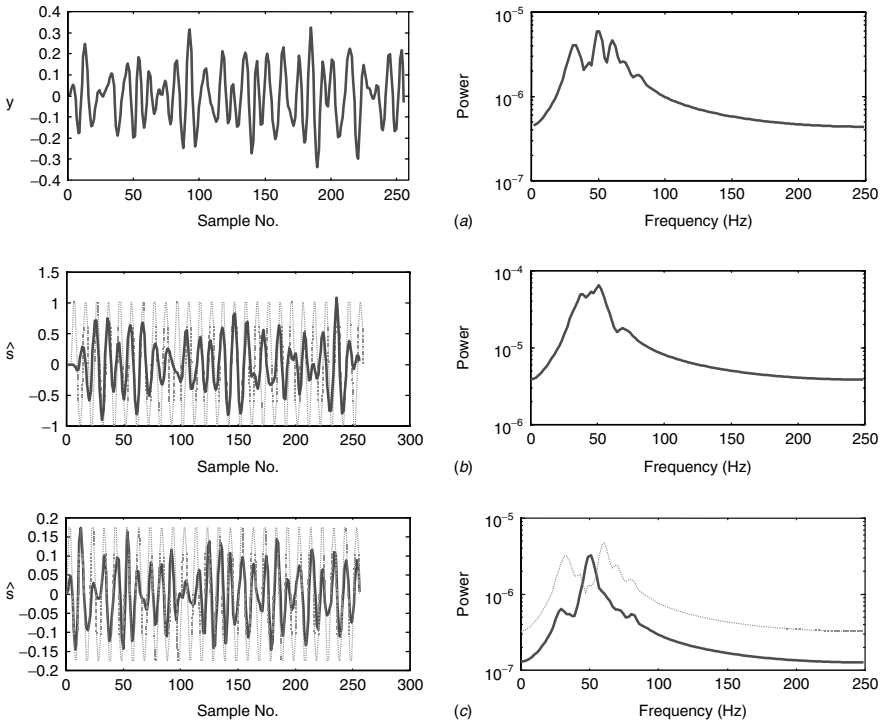



**Figure 7.14.** Sonar processing simulation with the adaptive predictor (helicopter) noise canceler (surface ship). (*a*) Simulated data and power spectrum (50 Hz target line). (*b*) *NLMS* predictor estimated sinusoid and spectrum ($N_h = 16$, $d = 3$, $\triangle_t = 0.1$, $\beta = 10^{-4}$). (*c*) *NLMS* canceler estimated sinusoid and spectrum ($N_h = 16$, $\triangle_t = 0.1$, $\beta = 10^{-4}$).

the predictor. This performance is expected because the reference channel provides additional information (*ALE* is not privy) that is incorporated into the processor. This completes the example.

### 7.6.4 Adaptive Time-Frequency *MBP*

In this subsection we develop the recursive-in-time approach to instantaneous spectral estimation enabling us to produce the desired spectrogram (amplitude vs. time vs. frequency) for event detection. Suppose that we parametrically model the enhanced measurement data by an instantaneous time-frequency representation specified by an autoregressive moving average model. This model takes the general difference equation form, $ARMA(N_a, N_c)$ given by

$$A(q^{-1}, t)y(t) = C(q^{-1}, t)\varepsilon(t) \tag{7.118}$$

for the enhanced measurement, $y(t)$, contaminated with zero-mean, white gaussian noise, $\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$, with the corresponding instantaneous polynomials at the

instant $t$ defined by

$$\begin{aligned} A(q^{-1},t) &= 1 + a_1(t)q^{-1} + \cdots + a_{N_a}(t)q^{-N_a} \\ C(q^{-1},t) &= c_o + c_1(t)q^{-1} + \cdots + c_{N_c}(t)q^{-N_c} \end{aligned} \tag{7.119}$$

Here the backward shift or delay operator is defined by, $q^{-i}y(t) := y(t-i)$ and therefore, we can write Eq. (7.118) simply as

$$y(t) = -\sum_{k=1}^{N_a} a_k(t)y(t-k) + \sum_{k=0}^{N_c} c_k(t)\varepsilon(t-k) \tag{7.120}$$

If we take the DFT of the difference equation with $\Omega = 2\pi f$, then we obtain the *instantaneous transfer function* (ignoring stochastic aspect)

$$H(e^{j2\pi f},t) = \frac{Y(e^{j2\pi f},t)}{\mathrm{E}(e^{j2\pi f},t)} = \frac{C(e^{j2\pi f},t)}{A(e^{j2\pi f},t)} \tag{7.121}$$

or more appropriately the corresponding *instantaneous power spectrum* given by

$$S(f,t) \equiv \left|H(e^{j2\pi f},t)\right|^2 = \left|\frac{C(e^{j2\pi f},t)}{A(e^{j2\pi f},t)}\right|^2 \tag{7.122}$$

So we see that if we use the parametric $ARMA(N_a, N_c)$ representation of the enhanced measurement signal and transform it to the spectral domain, then we can obtain the instantaneous spectral estimate.

There are a wealth of adaptive ARMA algorithms available in the literature ([19], [5]), but since we are primarily interested in estimating the spectrum at each time instant, we confine our choices to those that are recursive-in-time, enabling us to achieve our goal without the loss of temporal resolution evolving from window-based methods such as the short-time Fourier transform. Recall that recursive-in-time algorithms take on the following generic form:

$$\begin{aligned} &\underline{\hat{\Theta}}(t+1) = \underline{\hat{\Theta}}(t) + \underline{K}(t)e(t) && \text{(Parameter update)} \\ &e(t) = y(t) - \hat{y}(t) = y(t) - \underline{\varphi}'(t)\underline{\hat{\Theta}}(t) && \text{(Prediction error)} \end{aligned}$$

$$\begin{aligned} \underline{\varphi}(t) &\equiv \left[y(t-1) \quad \cdots \quad y(t-N_a-1) \mid e(t) \quad \cdots \quad e(t-N_c)\right], \\ \underline{\hat{\Theta}}(t) &\equiv \left[-\hat{a}_1(t-1) \quad \cdots \quad -\hat{a}_{N_a}(t-N_a-1) \mid \hat{c}_o(t) \quad \cdots \quad \hat{c}_{N_c}(t-N_c)\right]' \end{aligned} \tag{7.123}$$

with $\underline{K}(t)$ the gain or weighting vector and the symbol defining the "best" (minimum error variance) estimate at the specified time. There are also many variations and forms of this basic recursion [19], but here we limit our application to the recursive prediction error method (*RPEM*) based on a local Gauss-Newton optimization method (see Chapter 4 for details).

| | |
|---|---|
| Criterion: | $\mathcal{J}(\underline{\theta}(t)) = E\{e^2(t)\}$ |
| Models: | |
| Signal: | $s(t) = -\sum_{i=1}^{N_a} a_i\, y(t-i)$ |
| Measurement: | $y(t) = s(t) + n(t)$ |
| Noise: | $n(t) = \sum_{i=0}^{N_c} c_i e(t-i)$ |
| Initial conditions: | $\underline{\Theta}(0)$ |
| Algorithm: | $\Theta(t) = \Theta(t-1) + \mu K(t) e(t)$ |
| Quality: | $P(t) = (I - K(t)\phi'(t))P(t-1)/\lambda(t)$ |

***Example 7.9*** For this application we synthesize the response of a ballistic rocket that has been launched and measured with a radar tracking system as it reenters the earth's atmosphere (see Sec. 9.1 for details). The vehicle dynamic motion, assumed stable, is characterized by it precession frequency and the small angular frequencies called nutations that it undergoes during reentry. We also synthesize some events that are simulated as simple phase changes in the temporal response. For our simulation we assumed a precession frequency of 0.235 Hz and nutation frequencies ranging between $1.6 \pm 0.24$ Hz, $2.4 \pm 0.24$ Hz, $3.2 \pm 0.24$ Hz, and $4.0 \pm 0.24$ Hz. In the spectrum these nutation frequencies appear in the side bands, since the signal model is essentially an amplitude modulated signal. So for these data we extract the precession frequency and the eight sidebands. We also synthesized three events at 550, 590, and 640 seconds. We simulated the data with additive zero-mean, gaussian noise with a variance of *SNR*= 20 dB for 1024 samples and processed the data using the *ARMA* model given by *ARMA*(35, 25).

We ran the instantaneous spectrogram estimator using the *RPEM* over the enhanced measurements, and the results are shown in Figure 7.15 below. Here we see that the spectrogram image clearly displays the dominant precession spectral peak (0.235 Hz) as it temporally evolves as well as the nutational side band frequencies. Note with this high *SNR* the synthesized "phase changes" are also clearly observable as indicated by the arrows. The measurement is time aligned and shown below the spectrogram as well. This example demonstrates the effectiveness of the adaptive *ARMA*-model spectrogram MBP on synthetic data.

There are many applications of adaptive signal processing ranging from adaptive identification, deconvolution, array beamforming, to adaptive digital filter design ([1]–[6]). This concludes the section on applications of adaptive signal processing. Next we consider a case study—the estimation of a plasma pulse from noisy measurements.

## 7.7 CASE STUDY: PLASMA PULSE ESTIMATION USING *MBP*

In this section we consider the design of a noise canceling filter to extract a plasma pulse from contaminated measurement data [29]. First we discuss the background of the problem and then the design of the processor.

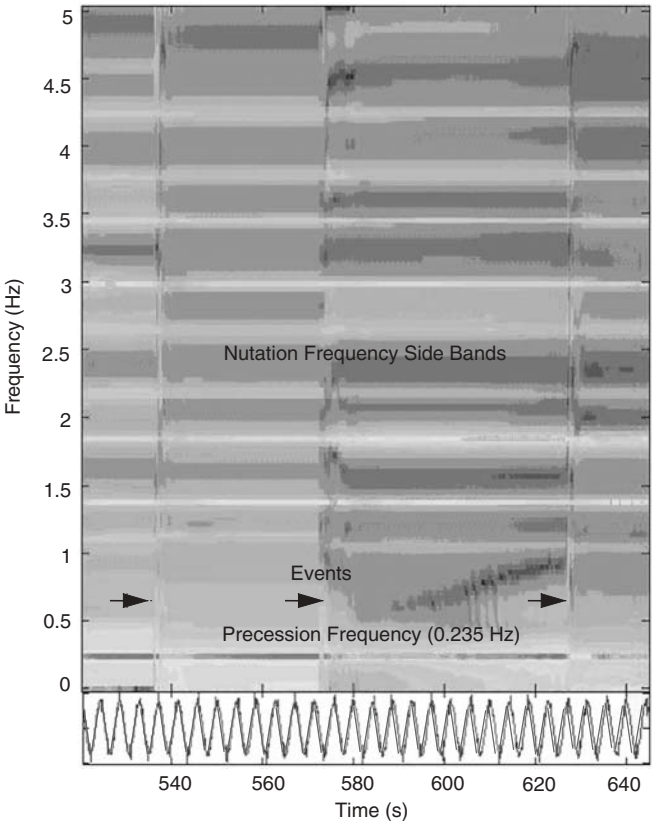



**Figure 7.15.** Adaptive ARMA spectrogram estimation for event detection: RPEM algorithm for ARMA(35, 25). Events detected are shown by the arrows.

Controlled *fusion* of heavy isotopes of hydrogen (deuterium and tritium) would enable virtually limitless energy and therefore provide a solution to the dwindling supply of conventional energy sources. Fusion reactions occur when ions of the hydrogen isotopes, heated to sufficient temperatures, collide and overcome the electrical forces of separation. The basic requirement for controlled fusion is to heat a plasma (or ionized gas) to high temperatures, in excess of $10^8$ degrees, and *confine* it for times long enough that a significant number of fusion events occur. One method of accomplishing this is call *magnetic confinement*. The magnetic confinement method used at Lawrence Livermore National Laboratory (LLNL) is the Tandem Mirror Experiment—Upgrade (TMX-U) experiment.

A parameter of significant importance to magnetically confined plasma is the diamagnetism of the plasma. The diamagnetism is a measure of energy density stored in hot particles that is used to determine the efficiency of the applied magnetic fields. In the TMX-U, a single-turn loop transformer—the so-called diamagnetic loop (DML)—is used as the sensor for the plasma diamagnetism. The DML sensor

is subjected to various noise sources that make the plasma estimation problem difficult. Variations of the magnetic field used to contain the plasma are present because of feedback circuits and ripple currents in the main power system. In many cases the signal that is used to determine the plasma diamagnetism is so badly corrupted with coherent frequency noise (ripple) that the plasma perturbation due to diamagnetism is not even visible. When the signals are approaching the noise level, or when the feedback control system has introduced a trend to the data, a sophisticated technique must be used for the processing of the measured signals. It must incorporate trend removal with the capability of removing the coherent noise without affecting the frequency content of the plasma perturbation itself. We chose to use the noise-canceling processor of the previous section.

First we analyze the acquired diamagnetic loop (DML) sensor measurements and show how the data can be processed to retain the essential information required for postexperimental analysis. The measured DML data is analog (anti-alias) filtered and digitized at a 25 KHz sampling rate (40 $\mu$s sample interval).

A typical experiment generates a transient signal (plasma) that is recorded for approximately 650 ms. Preprocessed data (decimated, etc.) and the frequency spectrum are shown in Figure 7.16 along with an expanded section of the transient pulse and noise. We note that the raw data are contaminated with a sinusoidal drift, linear

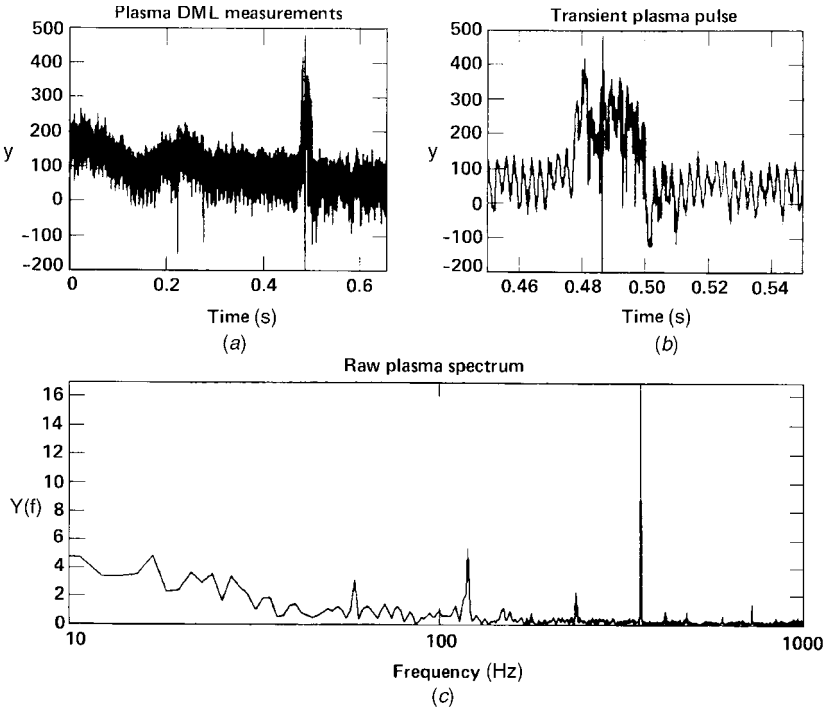



**Figure 7.16.** Preprocessed diamagnetic loop measurement data and spectrum. (*a*) Raw data. (*b*) Zoomed pulse. (*c*) Fourier spectrum.

trend, and random noise as well as sinusoidal disturbances at harmonics of 60 Hz, the largest at 360 Hz caused by the feedback circuits and ripple currents in the main power system. The pulse is also contaminated by these disturbances. We also note that some of the plasma information appears as high-energy spikes—(pulses) *riding* on the slower plasma buildup pulse.

A processor must be developed to eliminate these disturbances, yet preserve all of the essential features of the transient plasma pulse and associated energy spikes. This application is ideally suited for noise canceling, since the signal and noise should not be correlated. This condition is satisfied by the DML measurement data, since the onset of the measurement consists only of the disturbances (trend and sinusoids), and the signal is available at the time of the transient plasma pulse.

During the operation of the TMX experiment a "shot" (injection of a plasma into the reactor) terminates after a few seconds, during this time data are collected and displayed so that the experimenter can adjust process parameters and criteria and perform another shot within a five-minute time period. So we see that even though the processor need not be online, it still must function in a real-time environment. Clearly, postexperimental analysis creates no restrictions on the processor design and alloted computational time. So we analyzed the performance of the processor to function for both real-time and postexperimental modes of operation. We studied the performance of the processor by varying its order **N**. The real-time processor must perform reasonably well enough to enable the experimenter to make the necessary decisions regarding the selection of process parameters for the next shot.

After some preliminary runs of the processor over various data sets we decided to use $N = 512$ weights for the postexperimental design, since it produced excellent results. Using the postexperimental design as a standard, we then evaluated various designs for weights in the range of $8 < N < 512$. Before we discuss the comparisons, let us consider the heuristic operation of the processor. The crucial step in the design of the canceler is the estimation of the optimal noise filter $\hat{h}$, which is required to produce the minimum variance estimate of the noise, $\hat{n}$. In essence we expect the filter to match the corresponding noise spectrum in magnitude and phase. This means that we expect the optimal filter to pass the spectral peaks of the noise and attenuate any signal information not contained in the reference. These results are confirmed as shown by the performance of the 512-weight filter shown in Figure 7.17*a*. Here we see that the filter passband enable most of the noise resonances to pass while signal energy is attenuated. The real-time design is shown below in Figure 7.17*b*. We see that the 64-weight filter still passes much of the noise energy but does not spectrally match the noise as well as the 512-weight filter, since there are fewer weights. These results are again confirmed in Figure 7.18 where the estimated and actual noise spectra are shown. Again we see that the 512-weight produces a much better spectral match, than the 64-weight design due to its increased resolution. Note that the highest energy noise spectral peaks were matched by both processors reasonably well thereby eliminating these disturbances in the canceling operation. Intermediate designs for the real-time processor fall in between these results where selecting higher number of weights resulted in better processor performance.

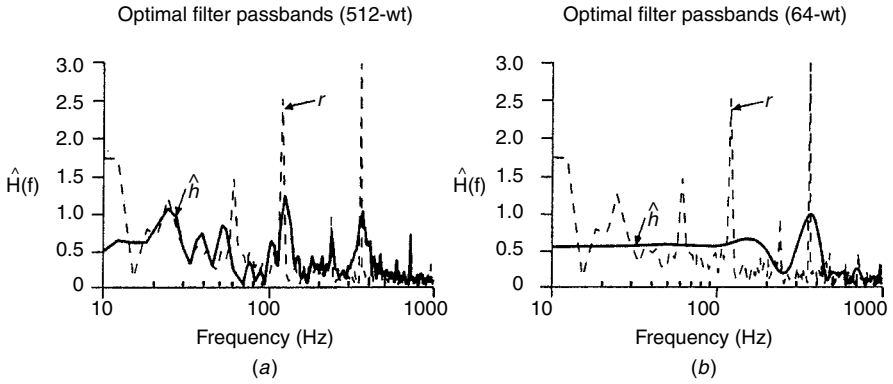



**Figure 7.17.** Optimal noise filter and noise spectra: *(a)* Post filter, and *(b)* real-time filter.

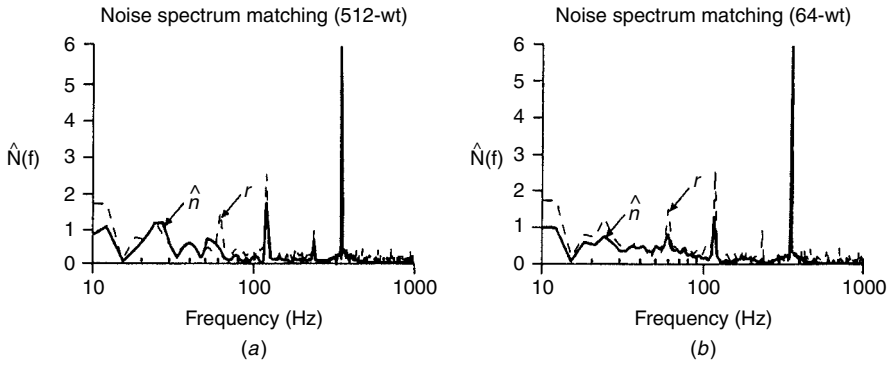



**Figure 7.18.** Optimal noise spectral matching: *(a)* Post filter, and *(b)* real-time filter.

The noise canceler algorithm was constructed using various commands in *MATLAB* [16]. Both the postexperimental and real-time designs were run on the data set described in Figure 7.16 and the results of the 512-weight design is shown in Figure 7.19. Here we see the raw and processed data and corresponding spectra. A closer examination of the estimated transient pulse shows that not only have the disturbances been removed, but that the integrity of the pulse has been maintained and all of the high-frequency energy spikes have been preserved. Note that the 512-weight processor has clearly eliminated the trends and sinusoidal disturbances (spectrum) and retained the transient plasma information. The real-time (64-weight) processor performs satisfactorily, but does not attenuate the disturbances as well as the postexperimental processor as discussed previously. Once these disturbances have been removed, the resulting signal can be processed further to remove the random noise and provide an estimate of the stored energy buildup in the machine. This completes the case study.

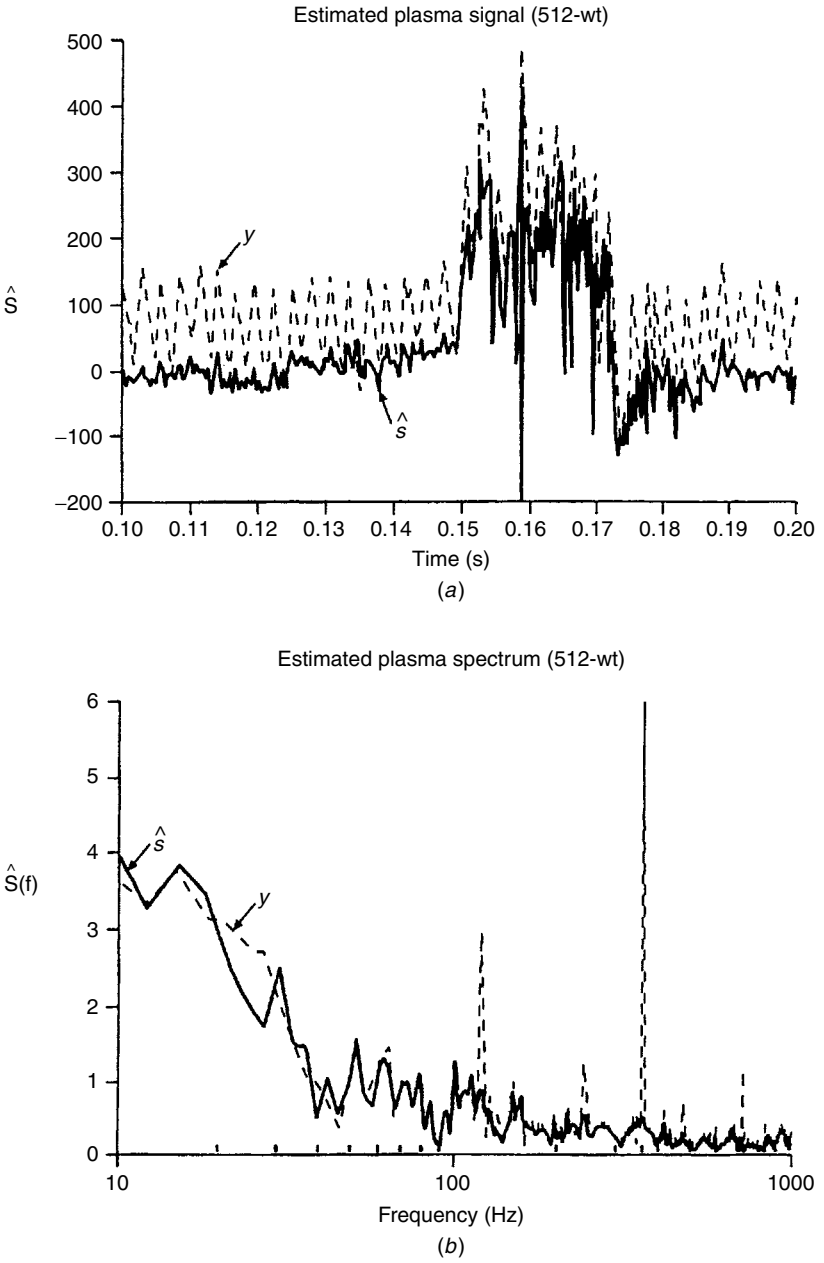



**Figure 7.19.** Postexperiment noise canceler design: *(a)* Plasma pulse, and *(b)* Spectra.

## 7.8 SUMMARY

In this chapter we introduced idea of the adaptive signal processing method and discussed various gradient-based adaption algorithms: stochastic gradient, Newton, and the instantaneous gradient. We then applied this approach to all-zero adaptive filter designs and investigated the stochastic and instantaneous gradient or least-mean-squared (*LMS*) approaches as well as the Newton-like recursive least squares (*RLS*) design. Next we investigated the pole-zero adaptive filter designs and showed how the *RPEM* algorithms of the previous chapter can be utilized to solve the adaptive problem. In this connection then we developed an *LMS*, *NLMS*, *RNLMS* algorithms. All-pole and all-zero adaptive lattice filters were developed and an alternative approach to the block processor of Chapter 4. We investigated various applications of the adaptive algorithms and investigated noise canceling filter designs and its variants, and the adaptive predictor and line enhancer. Finally we discussed a case study involving the design of noise cancelling filters for the extraction of a transient pulse.

## *MATLAB* NOTES

*MATLAB* can be used to develop the adaptive signal processors discussed in this chapter. Most of the algorithms are part of the *SYSTEM IDENTIFICATION TOOLBOX* that provides a variety of the adaption algorithms along with accompanying analysis tools that can be used to evaluate, display and compare processor performance. The family of adaptive commands take the general form, (**r***) where the wildcard (*) is followed by the model set selected.

The *ARMAX* (**rarmax**) family is represented along with its special cases *ARX* (**rarx**) and the general prediction error method (**rpem**) in which many of the general multiple input−single output adaptive structures can be processed. Special model sets such as the *Box-Jenkins* (**rbj**) formulations and the all-zeros or *FIR* forms using the output-error structures (**roe**) discussed in this chapter [30]. Finally pseudolinear regression models (**rplr**) that encompass the extended least squares (*ELS*) and *HARF* algorithms are also available.

All of these implementations enable the user to select the accompanying adaption algorithms along with their inherent parameters (e.g., forgetting factor). Besides the prediction error and *RLS* (linear regressions) algorithms already mentioned, the standard gradient-based *LMS* and *NLMS* choices are also available with their associated variants depending on the model set selected.

## REFERENCES

1. B. Widrow and S. Stearns, *Adaptive Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1984.
2. S. Haykin, *Adaptive Filter Theory*, 3rd Ed., Englewood Cliffs; NJ: Prentice-Hall, 1996.
3. R. Monzingo and T. Miller, *Introduction to Adaptive Arrays*, New York: Wiley, 1980.
4. G. Goodwin and K. Sin, *Adaptive Filtering, Prediction and Control*, Englewood Cliffs, NJ: Prentice-Hall, 1984.

5. L. Ljung and T. Soderstrom, *Theory and Practice of Recursive Identification*, Cambridge: MIT Press, 1983.
6. P. Eykhoff, *System Identification—Parameter and State Estimation*, New York: Wiley, 1974.
7. D. Luenberger, *Optimization by Vector Space Methods*, New York: Wiley, 1969.
8. D. Luenberger, *Introduction to Linear and Nonlinear Programming*, Reading, MA: Addison-Wesley, 1973.
9. P. Gill, W. Murray, and M. Wright, *Practical Optimization*, New York: Academic Press, 1983.
10. A. Sayed, *Fundamentals of Adaptive Filtering* Hoboken, NJ: Wiley, 2003.
11. G. Noble, *Applied Linear Algebra*, Englewood Cliffs, NJ: Prentice-Hall, 1967.
12. S. Orfanidis, *Optimum Signal Processing*, New York: Macmillan, 1985.
13. R. Bitmead and B. Anderson, "Adaptive frequency sampling filters," *IEEE Trans. Circ. Syst.*, **28**, 524–535 1981.
14. M. Hayes, *Statistical Digital Signal Processing and Modeling*, New York: Wiley, 1996.
15. R. Bitmead and B. Anderson, "Performance of adaptive estimation algorithms in dependent random environments," *IEEE Trans. Auto. Contr.*, **25**, 788–794 1980.
16. Mathworks, *MATLAB User's Manual*, Boston: Mathworks, 1990.
17. B. Friedlander, "System identification techniques for adaptive signal processing," *Circ. Syst. Signal Process.*, **1**, 3–41 1982.
18. L. Griffiths, "A continuously adaptive filter implemented as a lattice structure," *Proc. ICASSP*, 1977.
19. J. Candy, *Signal Processing: The Modern Approach*, New York: McGraw-Hill, 1988.
20. S. Kay, *Modern Spectral Estimation*, Englewood Cliffs, NJ: Prentice-Hall, 1988.
21. C. Cowen and P. Grant, *Adaptive Filters*, Englewood Cliffs, NJ: Prentice-Hall, 1985.
22. L. Griffiths "Rapid measurement of digital instantaneous frequency," *IEEE Trans. Acoust. Speech Signal Process.*, **23**, 207–222 1975.
23. L. Griffiths "An adaptive lattice structure for noise-cancelling applications," *Proc. ICASSP*, 1978.
24. M. Morf and D. Lee, "Recursive least-squares ladder forms for fast parameter tracking," *Proc. IEEE CDC Conf.*, 1979.
25. D. Lee, M. Morf, and B. Friedlander, "Recursive least-squares ladder estimation algorithms," *IEEE Trans. Circuits Syst.*, **28**, 467–481 1981.
26. B. Widrow, J. Glover, J. McCool, J. Kaunitz, C. Williams, R. Hearn, J. Zeidler, E. Dong, and R. Goodlin, "Adaptive noise canceling: Principles and applications," *Proc. IEEE*, **63**, (12), 1692–1716 1975.
27. J. Zeidler, "Performance analysis of LMS adaptive prediction filters," *Proc. IEEE*, **78**, (12), 1781–1806 1990.
28. P. Clarkson, *Optimal and Adaptive Signal Processing*, Boca Raton, FL: CRC Press, 1993.
29. J. Candy, T. Casper, and R. Kane, "Plasma estimation: A noise-canceling application," *Automatica*, **22**, 223–229 1986.
30. L. Ljung, *System Identification: Theory for the User*, Englewood Cliffs, NJ: Prentice-Hall, 1987.

## PROBLEMS

**7.1** Suppose that we are given a set of measurements $\{y(1),\ y(2)\}$and asked to design a two-weighted adaptive signal estimator with $\underline{w}' = [w(1)\ w(2)]$ such that

$$\hat{s}(t) = \sum_{i=1}^{2} w(i)y(i)$$

**(a)** Find the optimal Wiener estimator.

**(b)** Find the gradient estimator.

**(c)** Find the Newton estimator.

**7.2** We are asked to design and analyze the performance of an all-zero processor for a desired signal given by an $AR$ model

$$s_d(t) = \frac{3}{4}s_d(t-1) + n(t)$$

and measurement given by

$$x(t) = \frac{1}{8}s_d(t) + v(t)$$

for $n,\ v$ zero-mean, white noise with variance 0.1 and 1 respectively. Design *both* a single-weight and two-weight processor using the following:

**(a)** Optimal Wiener filter

**(b)** Stochastic gradient processor

**(c)** Stochastic Newton processor

**(d)** *LMS* processor

**(e)** *RLS* processor

**7.3** Suppose for the previous problem that the measurement is given by

$$x(t) = s_d(t-1) + s_d(t-2) + v(t)$$

then repeat the designs.

**7.4** We are given a two-weight, adaptive signal processor characterized by

$$\underline{h}(i+1) = h(i) - \triangle\frac{\partial J(\underline{h})}{\partial h}$$

where $e(t) = s_d(t) - \underline{h}'\underline{X}$.

**(a)** Derive the stochastic gradient, Newton, and *LMS* solutions; explicitly show the entries of the associated matrices.

**(b)** Suppose $R_{xx}(k) = \alpha^{|k|}$ and $R_{s_dx}(k) = \beta^{|k|}$; repeat part *(a)*. What are reasonable step-sizes for convergence? Calculate the misadjustment for a selected step-size.

**(c)** Repeat part *(a)* with $\alpha = 0.5$ and $\beta = 0.8$.

**(d)** Simulate this process, and apply the adaptive algorithms. Are the results as predicted?

**7.5** We are asked to compare the performance of two processors each evolving from the output covariance matrix

$$R_{xx} = \begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & 1 \end{bmatrix}$$

The cross-covariance vectors are given by

$$\gamma^1_{s_dx} = \begin{bmatrix} 1 \\ \frac{1}{2} \end{bmatrix}, \quad \gamma^2_{s_dx} = \begin{bmatrix} 2 \\ \frac{1}{4} \end{bmatrix}$$

**(a)** Find the optimal weight vectors, $h^i_{\text{opt}}$, $i = 1, 2$.

**(b)** Calculate the minimum mean-squared error, $J_{\min}$, $i = 1, 2$. Which estimator can be expected to perform better?

**(c)** Find expressions for each learning curve in terms of $J^i_{\min}$.

**(d)** Assume you are going to implement a stochastic gradient algorithm. What is a reasonable step-size for stability? For speed?

**(e)** Perform the eigenvalue-eigenvector transformation and sketch the error surface.

**7.6** Suppose that we have a desired signal characterized by a second-order*AR* process, that is,

$$s_d(t) + \alpha_1 s_d(t-1) + \alpha_2 s_d(t-2) = e(t)$$

with $e(t)$ zero-mean, white of variance $R_{ee}$. Assume that we are to design a two-weight, all-pole stochastic gradient processor characterized by

$$\underline{\hat{s}}_d(t) = \underline{h}'\underline{X} = [h_1 \; h_2] \begin{bmatrix} x(t) \\ x(t-1) \end{bmatrix}$$

**(a)** In terms of the *AR* process parameters calculate the covariance matrix, $R_{xx}$.

**(b)** Determine the eigenvalue spread of $R_{xx}$ and the corresponding eigenvectors (normalized to unit length).

**(c)** Calculate the minimum mean-squared error and sketch a plot as a function of eigenvalue spread. What is the overall conclusion?

**7.7** Calculate the optimal all-zero Wiener solutions for the following process statistics and find $\underline{h}_{opt}$, $J_{min}$ and obtain the corresponding eigensolutions:

**(a)** $R_{xx} = \begin{bmatrix} 6 & 3 \\ 3 & 6 \end{bmatrix}$, $R_{s_d x} = \begin{bmatrix} 21 \\ 24 \end{bmatrix}$, $R_{s_d s_d} = 126$

**(b)** $R_{xx} = \begin{bmatrix} 1 & 1/2 \\ 1/2 & 1 \end{bmatrix}$, $R_{s_d x} = \begin{bmatrix} 1/4 \\ 1/2 \end{bmatrix}$, $R_{s_d s_d} = 64$

**7.8** An unknown plant can be "identified" using the stochastic gradient algorithm by using the connections shown below

**(a)** Develop the stationary solution to this problem.

**(b)** Develop the stochastic gradient and *LMS* iterators.

**(c)** Suppose that the "unknown" model is given by an *FIR* model

$$y(t) = (1 + 1.8q^{-1} + 0.9q^{-2})x(t)$$

Assume that $x, n$ are white with $R_{xx} = 1/12$, $R_{nn} = 1/12$. Simulate this system and obtain the optimal stationary and *LMS* solutions. Calculate the minimum mean-squared error in this case. Compare the solutions.

**(d)** *FIR* filters can also be designed in this manner by specifying the unknown system coefficients and allowing the adaptive algorithm to learn the response. Start with a second-order Butterworth low-pass response ($f_c = 10$ Hz) and excited the filter with pseudorandom noise. Design a 16, 32, and 64 weight filter and "teach" the adaptive filter. Discuss your results.

**7.9** The unknown input can be estimated or "deconvolved" adaptively

**(a)** Develop the stationary solution to this problem.

**(b)** Develop the stochastic gradient and *LMS* iterators.

**(c)** Suppose we use *ARMAX*(2, 1, 1, 1) model and design an adaptive deconvolver using the *LMS* algorithm. Choose weights and compare with the stationary solution.

**7.10** Suppose that we are asked to develop a single-weight noise canceler with signal

$$y = s + n$$
$$n = Fr$$

with $r$ the reference noise source.

**(a)** Show that the optimal weight $h = F$ and that complete noise cancelation occurs with this choice.

**(b)** Suppose that the noise model is

$$n = Fr + v$$

with $v$ random white. What is the optimal weight now? Does complete cancelation still occur?

**(c)** Suppose that the signal model is

$$y = s + n + v$$

Are the results different than in *(b)*?

**(d)** Suppose that the noise model also contain signal information that is

$$\hat{n} = F\ r + \triangle\ s$$

Calculate the output of the canceler $e(t)$ and show that when $\triangle = 0$, the results are identical to *(a)*.

**7.11** In the adaptive *LMS* noise canceler the *complex* algorithm is given by

$$\underline{h}(i+1) = \underline{h}(i) + \triangle e(t)\underline{x}^*(t)$$

Suppose that $x(t) = Ae^{j\omega_0 t}$, obtain a scalar weight update equation, and using $Z$-transforms, show that the transfer function is that of a "notch filter" that can be controlled by the step-size; that is, a pole at $z = e^{j\omega_0}(1 - \triangle|A|^2(M+1))$.

**7.12** Simulate a 10 Hz sinusoidal signal contaminated with random noise of variance $R_{nn} = 0.01$. Now assume that it is related to a reference noise signal, that is, $AR(2)$

$$r(t) = 1.5r(t-1) + 0.1562r(t-2) = n(t)$$

Design a stationary and adaptive solution to this problem.

**7.13** Suppose that we are asked to develop the optimal $d$-step prediction solution for a stationary signal. Using an all-zero filter

$$\underline{r}(t+d) = \underline{h}'\underline{y}$$

**(a)** Derive the optimal estimate of Eqs. (7.6) through (7.13).

**(b)** Derive the adaptive stochastic gradient and corresponding *LMS* solutions to this problem.

**7.14** An antenna array can be considered a "linear combiner" as shown below

**(a)** Derive the optimal Wiener "stationary" solution to this problem.

**(b)** Derive the adaptive stochastic gradient and *LMS* solutions.

**(c)** Suppose that this represents a two-element antenna array, and we are asked to design an adaptive nulling scheme ($s_d(t) = 0$). What are the optimal weights for this problem?

**7.15** An adaptive predictor can be used as a "self-tuning filter." Suppose that we would like to extract a periodic signal in broadband noise using the predictor shown below.

**(a)** Discuss the operation of this device and an appropriate choice of $d$, the prediction distance.

**(b)** Simulate a 50 Hz sinusoid in bandpass Gaussian noise (cutoffs: 10, 90 Hz) of variance 0.01. Based on estimated covariances select $d$ and obtain in the adaptive ($LMS$) solution.

# 8

# ADAPTIVE STATE-SPACE MODEL-BASED PROCESSORS

## 8.1 STATE-SPACE ADAPTION ALGORITHMS

In this section we briefly introduce the idea of adaptive model-based processing (*AMBP*) using the state-space model. Based on Chapter 7, it seems intuitively obvious that replacing the model set and specializing the corresponding adaption algorithm should be a relatively straightforward task. Unfortunately, because of the inherent complexity of the underlying multiple input–multiple output (*MIMO*) Gauss-Markov model as well as the complexity of the corresponding vector-matrix adaption algorithms, the simplicity of the adaptive structure tends to get lost in the details.

Here we start with the basic adaptive processor structure introduced in Chapter 7 and shown in Figure 8.1*a*. Selecting the state-space model set with some slight modifications ($s \rightarrow x$ or $y$) to account for the Gauss-Markov model and the variety of estimates available for the underlying *MBP* (state, measurement and innovations). It is clear from Figure 8.1*b* that the adaptive algorithm structure easily accommodates the state-space and *MBP* structure. Examining the figure more closely, we see that when we produce a filtered measurement, we have the innovations sequence as our (prediction) error which is to be minimized by the adaption algorithm (parameter estimator) until the underlying statistical properties of the innovations are satisfied.

With this structure in mind, we can now describe the various types of "adaptive" model-based processing schemes discussed in the literature. The different methods of adaptive state-space processors can be categorized as Bayesian, maximum like-

*Model-Based Signal Processing*, by James V. Candy

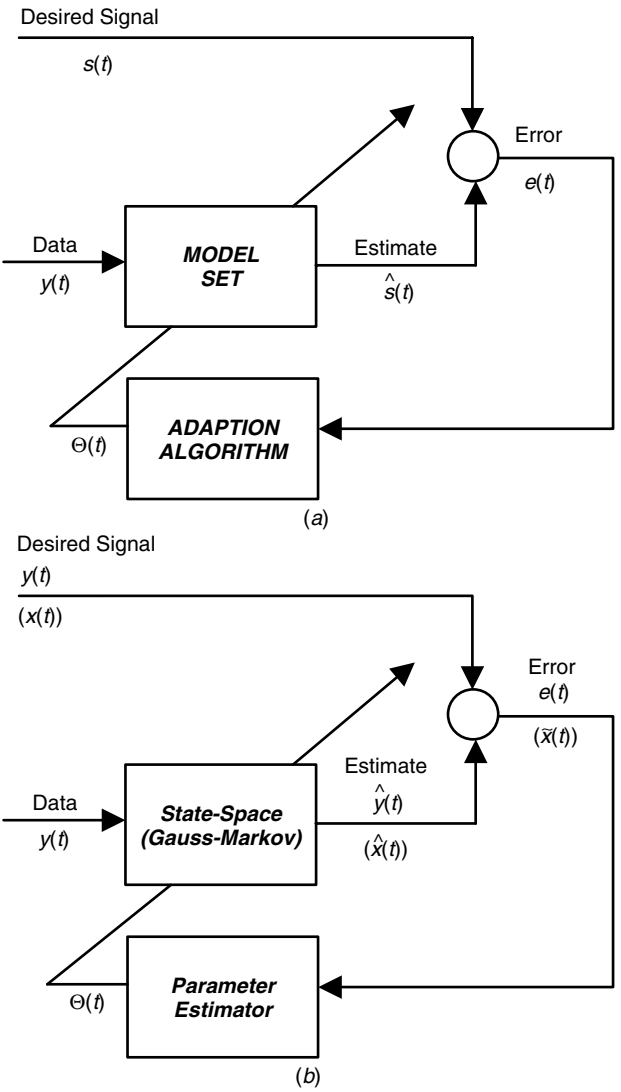



**Figure 8.1.** Adaptive processor structure: (*a*) Conventional adaptive processor. (*b*) State-space adaptive processor: measurement or state estimation.

lihood, correlation, and covariance-based. The Bayesian and maximum likelihood approaches have been discussed previously in Chapters 5 and 6 for the linear and nonlinear *MBP*. When a parameter vector is "augmented" with the state into the linear state-space model, the *joint* state and parameter estimation problem becomes nonlinear and can be solved leading to both Bayesian and likelihood solutions ([1], [2]). Correlation and covariance methods typically evolve from the original idea of avoiding divergence in the *MBP*. Recall from Chapter 5 (Section 5.6)

that divergence of the *MBP* occurs when the gain becomes small and ignores the measurement data. In this case the processor estimates "diverge" from the truth giving completely erroneous results. The primary remedy under these conditions is to increase the gain through the noise covariance matrices that leads to the covariance-based methods of *AMBP*.

Perhaps it is more appropriate to distinguish the *AMBP* from the fact that they form a class of "parametrically" adaptive schemes in which the parameters are embedded in the processor model. In the conventional adaptive processing methods of Chapter 7, the embedded models are typically "black-box" or at most "gray-box" models with no apparent relationship to the underlying physical phenomenology generating the data. The state-space representation can clearly fall into this class especially when *MIMO* data are available. However, one of the major advantages of the state-space approach is the ability to use the models developed from the physics in which the parameters have true physical meaning. In this situation a more popular terminology is called *physics-based* processing rather than the more general model-based. We will discuss physics-based processing in the next chapter when we examine a variety of case studies using these schemes. In any case we will call the *AMBP parametrically adaptive*. If our parameter set is constrained to be only adjusting the noise covariances, then it still can be considered a special case of this approach.

The following sections will discuss the development of adaptive state-space processors based on linear time-invariant models concentrating on the innovations model first and then proceeding to the covariance-based approaches. These sections are followed by the nonlinear state-space models, perhaps the most versatile class of all. First, we develop the general linear structure in the next section.

## 8.2 ADAPTIVE LINEAR STATE-SPACE *MBP*

In this section we discuss the development of adaptive linear state-space models based on the recursive prediction error (*RPE*) approach. We introduce the underlying generic representation of the adaptive processor for this case and then develop special linear structures employing this approach.

Recursive prediction error algorithms are based on a structure that can abstractly be cast into models that are characterized by filtering input/output data through a linear processor [1]. The generic state-space processor we have used throughout is represented by the *recursive prediction error model*, which can be cast, in general, as

$$\begin{aligned} \overline{\mathcal{X}}_\theta(t+1) &= \overline{\mathcal{A}}(\Theta)\overline{\mathcal{X}}_\theta(t) + \overline{\mathcal{K}}(\Theta)y(t) \\ \hat{\overline{\mathcal{Y}}}_\theta(t) &= \overline{\mathcal{C}}(\Theta)\overline{\mathcal{X}}_\theta(t) \end{aligned} \tag{8.1}$$

where $\overline{\mathcal{A}}$, $\overline{\mathcal{K}}$, and $\overline{\mathcal{C}}$ are matrix functions of $\Theta$, the unknown parameter vector. We have assumed that the predictor is in state-space form, whereas in Chapter 4 we started with the *ARMAX* model set.

The *RPE* algorithms are derived from a weighted quadratic prediction error criterion (see Section 4.5 for details) that evolves by minimizing the weighted sum-squared error given by

$$\min_{\theta} \quad \mathcal{J}(\Theta) = \frac{1}{2}\sum_{k=1}^{t} e'_{\theta}(k)\Lambda^{-1}(\Theta)e_{\theta}(k) \tag{8.2}$$

for the prediction error or innovations, $e_\theta \in R^{N_y \times 1}$ and the weighting matrix, $\Lambda(\Theta) \in R^{N_y \times N_y}$. The algorithm is developed by performing a second-order Taylor series expansion about the parameter estimate, $\hat{\Theta}$, and applying the Gauss-Newton assumptions (e.g., $\hat{\Theta} \rightarrow \Theta_{\text{true}}$) leading to the basic *parameter recursion*

$$\hat{\Theta}(t+1) = \hat{\Theta}(t) - \left[\nabla_{\theta\theta}\mathcal{J}(\hat{\Theta}(t))\right]^{-1}\nabla_{\theta}\mathcal{J}(\hat{\Theta}(t)) \tag{8.3}$$

where $\nabla_\theta \mathcal{J}(\Theta)$ is the $N_\theta \times 1$ *gradient vector* and $\nabla_{\theta\theta}\mathcal{J}(\Theta)$ is the $N_\theta \times N_\theta$ *Hessian matrix*. As part of the *RPE* algorithm, we must perform these operations on the predictor; therefore we define the *gradient matrix* as before by

$$\Psi_\theta(t) := \nabla_\theta \hat{\overline{\mathcal{Y}}}'_\theta(t|t-1) = \left[\psi_{\theta_1}(t)\ldots\psi_{\theta_{N_\theta}}(t)\right]'$$
$$\text{for} \quad \Psi_\theta \in \mathcal{R}^{N_\theta \times N_y}; \quad \psi_\theta \in \mathcal{R}^{N_y \times 1} \tag{8.4}$$

Following the *RPE* approach, a gradient filter must also be developed that incorporates the $N_y$-columns of the gradient matrix, $\psi_{\theta_i}(t)$. Therefore the following matrices are introduced to incorporate the required gradients of this filter as

$$F(\Theta, \overline{\mathcal{X}}_\theta, y) := \nabla_\theta[\overline{\mathcal{A}}(\Theta)\overline{\mathcal{X}}_\theta(t) + \overline{\mathcal{K}}(\Theta)y(t)] \qquad \text{for} \quad F \in \mathcal{R}^{N_x \times N_\theta}$$
$$H(\Theta, \overline{\mathcal{X}}_\theta) := \nabla_\theta[\overline{\mathcal{C}}(\Theta)\overline{\mathcal{X}}_\theta(t)] \qquad \text{for} \quad H \in \mathcal{R}^{N_y \times N_\theta} \tag{8.5}$$

Here the $j$th column of $F$, $f_j(\Theta_i)$, and that of $H$, $h_j(\Theta_i)$, are simply the derivatives with respect to the $i$th parameter such that

$$f_j(\Theta_i) := \left[\frac{\partial}{\partial\theta_i}\overline{\mathcal{A}}(\Theta)\right]\overline{\mathcal{X}}_\theta(t) + \left[\frac{\partial}{\partial\theta_i}\overline{\mathcal{K}}(\Theta)\right]y(t)$$
$$h_j(\Theta_i) := \left[\frac{\partial}{\partial\theta_i}\overline{\mathcal{C}}(\Theta)\right]\overline{\mathcal{X}}_\theta(t) \qquad \text{for } i = 1, \ldots, N_\theta \tag{8.6}$$

Thus the corresponding *matrix* gradient filter will be of the form

$$\overline{\mathcal{G}}_\theta(t+1) = \overline{\mathcal{A}}(\Theta)\overline{\mathcal{G}}_\theta(t) + F(\Theta, \overline{\mathcal{X}}_\theta, y)$$
$$\Psi_\theta(t) = \overline{\mathcal{C}}(\Theta)\overline{\mathcal{G}}_\theta(t) + H(\Theta, \overline{\mathcal{X}}_\theta) \tag{8.7}$$

for $\overline{\mathcal{G}}_\theta \in \mathcal{R}^{N_x \times N_\theta}$ with row vector $g_{\theta_i} \in \mathcal{R}^{N_x \times 1}$. The matrix gradient filter can be transformed to the conventional state-space representation by defining the

augmented state and measurement vectors as

$$\mathcal{X}_\theta(t) := \begin{bmatrix} \overline{\mathcal{X}}_\theta(t) \\ -- \\ g_{\theta_1}(t) \\ \vdots \\ g_{\theta_{N_\theta}}(t) \end{bmatrix} \quad \text{and} \quad \hat{\mathcal{Y}}_\theta(t) := \begin{bmatrix} \hat{\overline{\mathcal{Y}}}_\theta(t) \\ -- \\ \psi_{\theta_1}(t) \\ \vdots \\ \psi_{\theta_{N_\theta}}(t) \end{bmatrix}$$

for $\overline{\mathcal{X}}_\theta \in \mathcal{R}^{(N_\theta+1)N_x \times 1}$ and $\hat{\mathcal{Y}}_\theta \in \mathcal{R}^{(N_\theta+1)N_y \times 1}$, leading to the final generic, augmented, linear *prediction error model* as

$$\begin{aligned} \mathcal{X}_\theta(t+1) &= \mathcal{A}(\Theta)\mathcal{X}_\theta(t) + \mathcal{K}(\Theta)y(t) \\ \hat{\mathcal{Y}}_\theta(t) &= \mathcal{C}(\Theta)\mathcal{X}_\theta(t) \end{aligned} \tag{8.8}$$

for $\mathcal{A}(\Theta) \in \mathcal{R}^{(N_\theta+1)N_x \times (N_\theta+1)N_x}$ a block diagonal matrix, diag$[\mathcal{A}_1(\Theta) \ldots \mathcal{A}_{N_\theta+1}(\Theta)]$, $\mathcal{K}(\Theta) \in \mathcal{R}^{(N_\theta+1)N_x \times N_y}$, and $\mathcal{C}(\Theta) \in \mathcal{R}^{(N_\theta+1)N_y \times (N_\theta+1)N_x}$. Here both $\mathcal{K}$ and $\mathcal{C}$ contain the column vectors, $f_j(\theta_i)$ and $h_j(\theta_i)$, respectively. Their explicit relationship is not critical. The important point is that a gradient filter can be constructed and placed into the generic prediction state-space form of Eq. (8.8).

Now that we have the augmented linear model, we can develop the general *RPE* algorithm for linear state-space models. Following Ljung [2], we start with the weighted quadratic cost function

$$\mathcal{J}(\Theta) = \tfrac{1}{2}E\{e'_\theta(t)\Lambda^{-1}(t)e_\theta(t)\} \qquad \text{for } e_\theta(t) := y(t) - \hat{\overline{\mathcal{Y}}}_\theta(t) \tag{8.9}$$

where $\Lambda(t)$ is the prediction error variance with $e_\theta \in R^{N_y \times 1}$ and $\Lambda \in R^{N_y \times N_y}$ as before. Thus we would like to minimize the prediction error with respect to the parameter vector, $\Theta$, $\min_\Theta \mathcal{J}(\Theta)$. We use the optimization approach starting with the parameter recursion of Eq. (8.3) to estimate both the gradient and Hessian.

Differentiating Eq. (8.9) using the chain rule, we obtain the gradient

$$\begin{aligned} \nabla_\theta \mathcal{J}(\Theta) &= \nabla_\theta e'_\theta(t)\Lambda^{-1}(t)e_\theta(t) = \nabla_\theta \left(y(t) - \hat{\overline{\mathcal{Y}}}_\theta(t)\right)' \Lambda^{-1}(t)e_\theta(t) \\ &= -\Psi_\theta(t)\Lambda^{-1}(t)e_\theta(t) \end{aligned} \tag{8.10}$$

and insert the gradient matrix from the definition of Eq. (8.4).

Defining the Hessian matrix as $R(t) := \nabla_{\theta\theta}\mathcal{J}(\Theta)$ and substituting for the gradient in Eq. (8.3), we obtain the parameter vector recursion

$$\hat{\Theta}(t+1) = \hat{\Theta}(t) + \gamma(t)\,[R(t)]^{-1}\,\Psi_\theta(t)\Lambda^{-1}(t)e_\theta(t) \tag{8.11}$$

where $\gamma(t)$ is an $N_\theta$-weighting vector sequence. Using our generic prediction state-space model of Eq. (8.8), we can now obtain $\Psi_\theta$ and $e_\theta(t)$. However, this does give us a recursive form, but it can be transformed under the following approximations to

yield the desired result (see [2] for details): $\mathcal{X}_\theta(t) \approx \mathcal{X}(t)$, $\hat{\mathcal{Y}}_\theta(t) \approx \hat{\mathcal{Y}}(t)$, $\Psi_\theta(t) \approx \psi(t)$ and $e_\theta(t) \approx e(t)$. Therefore we obtain the required generic model structure for the *RPE* approach

$$\begin{aligned} \mathcal{X}(t+1) &= \mathcal{A}(\Theta)\mathcal{X}(t) + \mathcal{K}(\Theta)y(t) \\ \hat{\mathcal{Y}}(t) &= \mathcal{C}(\Theta)\mathcal{X}(t) \\ e(t) &= y(t) - \hat{\mathcal{Y}}(t) \end{aligned} \tag{8.12}$$

Substituting these parameters into the parameter vector relation above, we obtain the following recursion:

$$\hat{\Theta}(t+1) = \hat{\Theta}(t) + \gamma(t)R^{-1}(t)\psi(t)\Lambda^{-1}(t)e(t) \tag{8.13}$$

Next we must develop a recursion for the Hessian to achieve a completely recursive algorithm. Under the Gauss-Newton approximation, $\hat{\Theta} \approx \Theta_{\text{true}}$, it has been shown [2] that the Hessian can be approximated by a weighted sample variance estimator of the form

$$R(t) = \frac{1}{t}\sum_{k=1}^{t} \gamma(k)\psi(k)\Lambda^{-1}(k)\psi'(k) \tag{8.14}$$

which is easily placed in recursive form to give

$$R(t) = R(t-1) + \gamma(t)\left[\psi(t)\Lambda^{-1}(t)\psi'(t) - R(t-1)\right] \tag{8.15}$$

Following the same Gauss-Newton arguments, the prediction error covariance can be placed in recursive form as well

$$\Lambda(t) = \Lambda(t-1) + \gamma(t)\left[e(t)e'(t) - \Lambda(t-1)\right] \tag{8.16}$$

**Table 8.1. Generic Linear State-Space *RPE* Algorithm**

| | |
|---|---|
| Prediction error | |
| $e(t) = y(t) - \hat{\mathcal{Y}}(t)$ | (Prediction error) |
| $\Lambda(t) = \Lambda(t-1) + \gamma(t)\left[e(t)e'(t) - \Lambda(t-1)\right]$ | (Prediction error covariance) |
| $R(t) = R(t-1) + \gamma(t)\left[\psi(t)\Lambda^{-1}(t)\psi'(t) - R(t-1)\right]$ | (Hessian) |
| Parameter estimation | |
| $\hat{\Theta}(t+1) = \hat{\Theta}(t) + \gamma(t)R^{-1}(t)\psi(t)\Lambda^{-1}(t)e(t)$ | (Parameter update) |
| Prediction | |
| $\mathcal{X}(t+1) = \mathcal{A}(\Theta)\mathcal{X}(t) + \mathcal{K}(\Theta)y(t)$ | (State) |
| $\hat{\mathcal{Y}}(t) = \mathcal{C}(\Theta)\mathcal{X}(t)$ | (Measurement/gradient) |
| for $\mathcal{X} = [\overline{\mathcal{X}}_\theta \mid g_\theta]'$ and $\hat{\mathcal{Y}} = [\hat{\overline{\mathcal{Y}}}_\theta \mid \psi_\theta]'$ | |
| Initial conditions | |
| $\hat{\mathcal{X}}(0), \quad \Theta(0), \quad \Lambda(0), \quad \tilde{R}(0)$ | |

This completes the brief description of the *RPE* algorithm for *generic* linear state-space models which we summarize in Table 8.1 (for more details see Ljung [2]). Next we apply the approach to the innovations model.

## 8.3 ADAPTIVE INNOVATIONS STATE-SPACE *MBP*

In this section we discuss the development of adaptive innovations state-space model representation for the linear time invariant case based on the recursive prediction error (*RPE*) approach. Here we are given the set of input–output data, and we would like to adaptively estimate the gain directly. We can consider this processor *parametrically adaptive*, since we are actually planning to estimate the parameters of the gain matrix. The approach we take closely follows Ljung [2] specifically using the *RPE* algorithm of Chapter 4 and the generic state-space model of the previous section.

We generalize our problem to:

> **GIVEN** a set of input/output data, $\{u(t), y(t)\}, t = 1, \ldots, N$. **FIND** the "best" (minimum error variance) set of the unknown parameters, $\{\Theta_i\}, i = 1, \ldots, N_\theta$, that characterize the innovations model of Eq. (8.12).

Before we begin this development let us investigate the properties of the innovations model more closely.

### 8.3.1 Innovations Model

As mentioned previously in the introduction, our task is to develop a parametrically adaptive version of the linear state-space model. In this subsection we concentrate on the innovations model discussed briefly in Chapters 2 and 5 (see Sections 2.10 and 5.11).

In our usual model development, we start with the Gauss-Markov representation given by the model set $\Sigma_{GM} := \{A, B, C, D, R_{ww}, R_{vv}, R_{wv}, x(0), P(0)\}$ and a variety of state-space problems are defined within this framework (e.g., state estimation). The same problems can be defined in terms of this model. Therefore the model set for the innovations model is defined as $\Sigma_{INV} := \{A, B, C, D, K, R_{ee}, x(0), P(0)\}$. The only problem here is that we are usually "not" usually given, $\Sigma_{INV}$, but $\Sigma_{GM}$. From the equivalence of these models, it is easy to show that if we are given $\Sigma_{GM}$ and we want to obtain $\Sigma_{INV}$, then we must develop the relations between the parameters of the model set. The equivalent solution is given by the set of relations called the *Kalman-Szego-Popov* (*KSP*) equations [3]. The *KSP* equations can be solved iteratively ([4], [5]) to obtain $(K, R_{ee})$ by directly implementing the innovations model for the *time-invariant* case. Of course, another approach is simply to execute the usual *MBP* algorithm of Table 5.1 or the steady-state *MBP* of Table 5.9, which is equivalent to the *KSP* equations.

We start this development by investigating the solution to the *multivariable* version of the spectral factorization theorem for *time-invariant* systems of Chapter 2 (see Eq. 2.88), which proves the existence of a spectral factor. In the multivariable case, the factor evolves from the generalized Kalman-Szego-Popov lemma [3], which establishes the existence of a *stochastic realization* defined by the conventional set of matrices: $\Sigma = \{A, C, R_{ww}, R_{wv}, R_{vv}\}$ such that the *PSD* is

$$S_{yy}(z) = \left[C(zI - A)^{-1} \mid I_{N_y}\right] \left[\begin{array}{c|c} R_{ww} & R_{wv} \\ \hline R'_{wv} & R_{vv} \end{array}\right] \left[\begin{array}{c} (z^{-1}I - A')^{-1}C' \\ \hline I_{N_y} \end{array}\right] \quad (8.17)$$

admitting a factorization as

$$S_{yy}(z) = \left[C(zI - A)^{-1} \mid I_{N_y}\right] \left[\begin{array}{c} Q \\ \hline R \end{array}\right] \left[Q' \mid R'\right] \left[\begin{array}{c} (z^{-1}I - A')^{-1}C' \\ \hline I_{N_y} \end{array}\right]$$

$$= \left[C(zI - A)^{-1}Q + R\right]\left[Q'(z^{-1}I - A')^{-1}C' + R'\right] = \mathcal{T}_y(z)\mathcal{T}'_y(z^{-1}) \quad (8.18)$$

Corresponding to the spectral factor, we can define a stochastic realization, $\Sigma_{KSP} := \{A, B, C, D\}$ based on the *KSP* lemma by using the equivalence of the spectral factor to the *sum decomposition* of Eq. (2.82), that is,

$$S_{yy}(z) = S^{+}_{yy}(z) + S^{-}_{yy}(z^{-1}) - C_o = C(zI - A)^{-1}B + D + D' + B'(z^{-1}I - A')^{-1}C' \quad (8.19)$$

for $C_o$, the output covariance matrix at lag zero. Here $\Sigma_{KSP}$ evolves from the conventional multivariable Gauss-Markov representation with correlated noise sources given by

$$\begin{aligned} x(t) &= Ax(t-1) + Bu(t-1) + w(t-1) \\ y(t) &= Cx(t) + Du(t) + v(t) \end{aligned} \quad (8.20)$$

where $w \sim \mathcal{N}(0, R_{ww})$, $v \sim \mathcal{N}(0, R_{vv})$ and $R_{wv}$.

It can be shown ([3], [4], [6]) that if $\Sigma_{KSP}$ is a stochastic realization, then there exists a positive definite, symmetric matrix $\mathcal{P}$ such that the following set of *KSP* equations are satisfied:

$$\begin{aligned} \mathcal{P} - A\mathcal{P}A' &= R_{ww} \\ B - A\mathcal{P}C' &= R_{wv} \\ D + D' - C\mathcal{P}C' &= R_{vv} \end{aligned} \quad (8.21)$$

The proof follows by equating like terms in the sum decomposition and spectral factors from the *KSP* lemma. It can also be shown that any stochastic realization

given by $\Sigma_{KSP}$ may not satisfy the *KSP* equations uniquely; therefore we turn to the innovations model which does provide a *unique* solution.

The equivalent *innovations model* for this case is given by

$$\begin{aligned}\hat{x}(t) &= A\hat{x}(t-1) + Ke(t-1)\\ y(t) &= C\hat{x}(t) + e(t)\end{aligned}$$

where $e \sim \mathcal{N}(0, R_{ee})$, $K$ is the optimal $N_x \times N_y$ gain or weighting matrix of the *MBP* given in Table 5.1 with corresponding estimated state covariance, $\hat{\mathcal{P}} := \text{cov}(\hat{x}(t))$.

Taking the $Z$-transform of the innovations model enables us to realize the spectral factorization as developed in Chapter 5 (see Eq. 5.140)

$$\begin{aligned}S_{yy}(z) = \left[C(zI - A)^{-1} \mid I_{N_y}\right] &\begin{bmatrix} KR_{ee}K' & | & KR_{ee} \\ -- & --- & -- \\ (KR_{ee})' & | & R_{ee} \end{bmatrix}\\ &\times \begin{bmatrix} (z^{-1}I - A')^{-1}C' \\ --- \\ I_{N_y} \end{bmatrix}\end{aligned} \tag{8.22}$$

which admits the unique spectral factorization (Wiener solution)

$$\begin{aligned}S_{yy}(z) &= \left[C(zI - A)^{-1} \mid I_{N_y}\right] \begin{bmatrix} KR_{ee}^{1/2} \\ -- \\ R_{ee}^{1/2} \end{bmatrix}\\ &\quad \times \left[R_{ee}^{T/2}K' \mid R_{ee}^{T/2}\right] \begin{bmatrix} (z^{-1}I - A')^{-1}C' \\ -- \\ I_{N_y} \end{bmatrix}\\ &= \left[C(zI - A)^{-1}KR_{ee}^{1/2} + R_{ee}^{1/2}\right]\left[R_{ee}^{T/2}K'(z^{-1}I - A')^{-1}C' + R_{ee}^{1/2}\right]\\ &= \mathcal{T}_e(z)\mathcal{T}_e'(z^{-1})\end{aligned} \tag{8.23}$$

Therefore we can define a *unique* set of noise covariance matrices represented by the minimal number of parameters ($N_xN_y$) to characterize $K$ and ($\frac{1}{2}N_y(N_y+1)$) to characterize $R_{ee}$. Thus we define the following covariances:

$$R_{ww}^e := KR_{ee}K', \quad R_{wv}^e := KR_{ee}, \quad \text{and} \quad R_{vv}^e := R_{ee} \tag{8.24}$$

Substituting these covariance relations and the estimated state covariance, $\hat{\mathcal{P}}$, into the *KSP* equations gives the solution using the innovations model as

$$\begin{aligned}\hat{P} - A\hat{P}A' &= KR_{ee}K'\\ B - AC' &= KR_{ee}\\ D + D' - C\hat{P}C' &= R_{ee}\end{aligned} \tag{8.25}$$

**Table 8.2. Iterative *KSP* Algorithm**

Innovations covariance

$$R_{ee}(i) = D + D' - C\hat{\mathcal{P}}(i)C'$$

Gain

$$K(i) = \left(B - A\hat{\mathcal{P}}(i)C'\right) R_{ee}^{-1}(i)$$

Estimated covariance

$$\hat{\mathcal{P}}(i+1) = A\hat{\mathcal{P}}(i)A' + K(i)R_{ee}(i)K'(i)$$

Initial conditions

$$\hat{\mathcal{P}}(0)$$

Stopping rule

$$\left|I - \hat{\mathcal{P}}(i) \times \hat{\mathcal{P}}^{-1}(i+1)\right| \leq \epsilon \qquad \text{for } \epsilon \ll 1$$

Solving these equations iteratively: $R_{ee}(i) \to R_{ee}$, $K(i) \to K$, and $\hat{\mathcal{P}}(i) \to \mathcal{P}$ yields the *KSP* algorithm using the innovations model summarized in Table 8.2. Consider the following example.

***Example 8.1*** Consider the tuned $RC$-circuit problem of Section 5.5. We would like to extract the innovations model from the original Gauss-Markov representation given by

$$x(t) = 0.97x(t-1) + u(t-1) + w(t-1)$$
$$y(t) = 2x(t) + v(t)$$

with $R_{ww} = 10^{-4}$, $R_{vv} = 4$, $x(0) = 2.5$, and $P(0) = 10^{-6}$. We can iteratively solve the *KSP* equations directly to obtain

$$K = 8.23 \times 10^{-4}$$
$$R_{ee} = 6.6 \times 10^{-3}$$
$$\hat{P} = 1.65 \times 10^{-3}$$

Therefore, the resulting innovations model is

$$x(t) = 0.97x(t-1) + u(t-1) + (8.23 \times 10^{-4})e(t-1)$$
$$y(t) = 2x(t) + e(t)$$

with $e \sim \mathcal{N}(0, 6.6 \times 10^{-3})$. Summarizing in the model-based framework, we have the following:

| | |
|---|---|
| Criterion: | $J = \text{trace } \tilde{P}(t\|t)$ |
| Models: | |
| Signal: | $x(t) = 0.97x(t-1) + u(t-1) + w(t-1)$ |
| Measurement: | $y(t) = 2x(t) + v(t)$ |
| Noise: | $w \sim \mathcal{N}(0, 10^{-4})$ and $v \sim \mathcal{N}(0, 4)$ |
| Initial conditions: | $\hat{x}(0\|0) = 2.5$ and $\tilde{P}(0\|0) = 1 \times 10^{-6}$ |
| Algorithm: | $\hat{K}$ from *KSP* Table 8.2 |
| Quality: | $\hat{P}$ from *KSP* Table 8.2 |

This completes the example.

Before we close this subsection, it is meaningful to place the linear (parametrically) adaptive state-space processors in perspective relative to the conventional (correlated) Gauss-Markov representation. Following Ljung [2], we are given the linear Gauss-Markov model with *unknown* parameters, $\Theta$. Therefore we now have (including the deterministic input)

$$\begin{aligned} x_\theta(t) &= A(\Theta)x_\theta(t-1) + B(\Theta)u(t-1) + w(t-1) \\ y(t) &= C(\Theta)x_\theta(t) + v(t) \end{aligned} \tag{8.26}$$

where $w \sim \mathcal{N}(0, R_{ww}(\Theta))$, $v \sim \mathcal{N}(0, R_{vv}(\Theta))$ with $R_{wv}(\Theta)$, and initial conditions $x_\theta(0)$ and $P_\theta(0)$. Here we see that the unknown parameters can enter the matrices in an arbitrary manner. The solution to this problem for fixed $\Theta$ is simply the *time varying MBP* in prediction form (see Table 5.7) parameterized by $\Theta$, that is,

$$\begin{aligned} \hat{x}_\theta(t+1|t) &= [A(\Theta) - K_\theta(t)C(\Theta)]\,\hat{x}_\theta(t|t-1) + B(\Theta)u(t) + K_\theta(t)y(t) \\ \hat{y}_\theta(t) &= C(\Theta)\hat{x}_\theta(t|t-1) \quad \text{for} \quad \hat{x}_\theta(0) \end{aligned} \tag{8.27}$$

with $K_\theta(t)$ the *time-varying* gain calculated directly from the model parameters as

$$\begin{aligned} K_\theta(t) &= \left[A(\Theta)\tilde{P}_\theta(t|t-1)C'(\Theta) + R_{wv}(\Theta)\right] R_{ee}^{-1}(\Theta) \\ \tilde{P}_\theta(t|t) &= A(\Theta)\tilde{P}_\theta(t|t-1)A'(\Theta) + R_{ww}(\Theta) - K_\theta(t)R_{ee}(\Theta)K_\theta'(t) \\ R_{ee}(\Theta) &= C(\Theta)\tilde{P}_\theta(t|t-1)C'(\Theta) + R_{vv}(\Theta) \quad \text{with} \quad \tilde{P}_\theta(0|0) \end{aligned} \tag{8.28}$$

As discussed in Section 5.8, if the system is time invariant, that is, $\tilde{P}_\theta(t) \to \tilde{P}(\Theta)$, $K_\theta(t) \to K(\Theta)$ as $t \to \infty$, then we have the *time-invariant predictor* given by

$$\begin{aligned} \hat{x}_\theta(t+1|t) &= [A(\Theta) - K(\Theta)C(\Theta)]\,\hat{x}_\theta(t|t-1) + B(\Theta)u(t) + K(\Theta)y(t) \\ \hat{y}_\theta(t) &= C(\Theta)\hat{x}_\theta(t|t-1) \end{aligned} \tag{8.29}$$

The corresponding *time invariant* gain and covariance equations evolve as before

$$\begin{aligned} K(\Theta) &= \left[A(\Theta)\tilde{P}(\Theta)C'(\Theta) + R_{wv}(\Theta)\right] R_{ee}^{-1}(\Theta) \\ \tilde{P}(\Theta) &= A(\Theta)\tilde{P}(\Theta)A'(\Theta) + R_{ww}(\Theta) - K(\Theta)R_{ee}(\Theta)K'(\Theta) \\ R_{ee}(\Theta) &= C(\Theta)\tilde{P}(\Theta)C'(\Theta) + R_{vv}(\Theta) \quad \text{with} \quad \tilde{P}(\Theta_o) \end{aligned} \tag{8.30}$$

Since the noise covariances *only* enter the predictor directly through the gain and innovations covariance matrices, $K(\Theta)$, $R_{ee}(\Theta)$, it is more pragmatic to parameterize them (fewer parameters) rather than $R_{ww}(\Theta)$, $R_{wv}(\Theta)$, $R_{vv}(\Theta)$ as demonstrated earlier in developing the *KSP* equations. This is easily accomplished with the predictor model of Eq. (8.27) directly. If we collect gain terms, then the *innovations model* again evolves as

$$\begin{aligned} \hat{x}(t+1|t) &= A(\Theta)\hat{x}(t|t-1) + B(\Theta)u(t) + K(\Theta)e(t) \\ y(t) &= C(\Theta)\hat{x}(t|t-1) + e(t) \\ e(t) &= y(t) - \hat{y}_\theta(t|t-1) \end{aligned} \tag{8.31}$$

which can directly be related to the Gauss-Markov representation using the unique forms of the covariance matrices defined earlier as

$$\begin{aligned} R_{ww}^{e}(\Theta) &= K(\Theta)R_{ee}(\Theta)K'(\Theta) \\ R_{wv}^{e}(\Theta) &= K(\Theta)R_{ee}(\Theta) \\ R_{vv}^{e}(\Theta) &= R_{ee}(\Theta) \end{aligned} \tag{8.32}$$

Thus the innovations model represents a unique characterization of the Gauss-Markov model and is a canonical form (see [7]) containing the fewest parameters for this representation. We have shown how to obtain the innovations model parameters, $\Sigma_{INV}$ from the $\Sigma_{GM}$ model using the iterative solution to the *KSP* equations or the steady-state *MBP* (see Chapter 5). Next we investigate adaptive solutions using the innovations model.

### 8.3.2 RPE Approach Using the Innovations Model

In this subsection we apply the generic *RPE* algorithm of Section 8.2 to the *time invariant* innovations model. However, we must further understand the underlying properties and evolution of the innovations. Since we are developing the parametrically adaptive *RPE* algorithm, we must first transform the basic innovations structure to the equivalent prediction form. Recall that the conventional *predictor* has the inherent structure

$$\begin{aligned} \hat{x}_\theta(t+1|t) &= A(\Theta)\hat{x}_\theta(t|t-1) + B(\Theta)u(t) + K(\Theta)e_\theta(t) \\ \hat{y}_\theta(t|t-1) &= C(\Theta)\hat{x}_\theta(t|t-1) \end{aligned} \tag{8.33}$$

Substituting for the innovations (see Eq. 8.31) and collecting like terms, we obtain the required *prediction error model* directly from the innovations representation, that is,

$$\begin{aligned}\hat{x}_\theta(t+1|t) &= [A(\Theta) - K(\Theta)C(\Theta)]\,\hat{x}_\theta(t|t-1) + B(\Theta)u(t) + K(\Theta)y(t)\\ \hat{y}_\theta(t|t-1) &= C(\Theta)\hat{x}_\theta(t|t-1)\end{aligned} \tag{8.34}$$

Comparing this model to the final generic state-space representation of Eq. (8.12), we see that the following mappings occur:

$$\begin{aligned}\mathcal{A}(\Theta) &\rightarrow A(\Theta) - K(\Theta)C(\Theta)\\ \mathcal{B}(\Theta) &\rightarrow [B(\Theta), K(\Theta)]\\ \mathcal{C}(\Theta) &\rightarrow C(\Theta)\end{aligned} \tag{8.35}$$

Therefore we only require the expression for the gradient matrix in terms of the innovations model to develop a recursive algorithm. From the generic development in Section 8.2 the $N_\theta \times N_y$*-gradient matrix* in terms of our innovations model is given by

$$\begin{aligned}\Psi_\theta(t) &= \nabla_\theta e_\theta(t) = \nabla_\theta \hat{y}'_\theta(t|t-1) = \nabla_\theta\,[C(\Theta)\hat{x}_\theta(t|t-1)]'\\ \Psi_\theta &\in \mathcal{R}^{N_\theta \times N_y}\end{aligned} \tag{8.36}$$

Now we discuss the key expressions, since understanding the details of this development is critical to understanding the algorithm itself. We start with the simple matrix expansion of the gradient matrix (with operations developed componentwise to avoid any confusion using the simple scalar differentiation).[1] Therefore

$$\begin{aligned}\Psi_\theta(t) &= \nabla_\theta\,[C(\Theta)\hat{x}_\theta(t|t-1)]'\\ &= \begin{bmatrix}\dfrac{\partial}{\partial\theta_1}\\ \vdots\\ \dfrac{\partial}{\partial\theta_{N_\theta}}\end{bmatrix}\Big[\mathbf{c}'_1(\theta)\hat{x}_\theta(t|t-1)\ldots\mathbf{c}'_{N_y}(\theta)\hat{x}_\theta(t|t-1)\Big]\\ &= \begin{bmatrix}\dfrac{\partial}{\partial\theta_1}[\mathbf{c}'_1(\theta)\hat{x}_\theta(t|t-1)] & \cdots & \dfrac{\partial}{\partial\theta_1}[\mathbf{c}'_{N_y}(\theta)\hat{x}_\theta(t|t-1)]\\ \vdots & \cdots & \vdots\\ \dfrac{\partial}{\partial\theta_{N_\theta}}[\mathbf{c}'_1(\theta)\hat{x}_\theta(t|t-1)] & \cdots & \dfrac{\partial}{\partial\theta_{N_\theta}}[\mathbf{c}'_{N_y}(\theta)\hat{x}_\theta(t|t-1)]\end{bmatrix}\end{aligned}$$

[1]Recall product rule as $\frac{d}{d\theta}(xy) = x\frac{dy}{d\theta} + y\frac{dx}{d\theta}$.

for $\mathbf{c}_i' \in \mathcal{R}^{1\times N_x}$ and $\hat{x}_\theta \in \mathcal{R}^{N_x\times 1}$. From the product rule we have

$$\Psi_\theta(t) = \begin{bmatrix} \frac{\partial \mathbf{c}_1'(\theta)}{\partial\theta_1}\hat{x}_\theta(t|t-1) + \mathbf{c}_1'(\theta)\frac{\partial \hat{x}_\theta(t|t-1)}{\partial\theta_1} & \cdots & \frac{\partial \mathbf{c}_{N_y}'(\theta)}{\partial\theta_1}\hat{x}_\theta(t|t-1) + \mathbf{c}_{N_y}'(\theta)\frac{\partial \hat{x}_\theta(t|t-1)}{\partial\theta_1} \\ \vdots & \cdots & \vdots \\ \frac{\partial \mathbf{c}_1'(\theta)}{\partial\theta_{N_\theta}}\hat{x}_\theta(t|t-1) + \mathbf{c}_1'(\theta)\frac{\partial \hat{x}_\theta(t|t-1)}{\partial\theta_{N_\theta}} & \cdots & \frac{\partial \mathbf{c}_{N_y}'(\theta)}{\partial\theta_{N_\theta}}\hat{x}_\theta(t|t-1) + \mathbf{c}_{N_y}'(\theta)\frac{\partial \hat{x}_\theta(t|t-1)}{\partial\theta_{N_\theta}} \end{bmatrix}$$

which can now be written in succinct notation as

$$\Psi_\theta(t) = \nabla_\theta\left[C(\Theta)\hat{x}_\theta(t|t-1)\right]' = \left[\nabla_\theta C(\theta)\right]\hat{x}_\theta(t|t-1) + C(\theta)\left[\nabla_\theta\hat{x}_\theta(t|t-1)\right] \tag{8.37}$$

Define the following terms, which will ease the notational burden

$$\mathcal{W}_\theta(t) := \nabla_\theta\hat{x}_\theta(t|t-1) \quad \text{and} \quad \mathcal{Z}_\theta(\hat{x}) := \nabla_\theta\left[C(\Theta)\right]\hat{x}_\theta(t|t-1) \tag{8.38}$$

for $\mathcal{W}_\theta \in \mathcal{R}^{N_x\times N_\theta}$, the *state predictor gradient weighting matrix* and $\mathcal{Z}_\theta \in \mathcal{R}^{N_y\times N_\theta}$, the *measurement gradient weighting matrix*. Thus, in terms of these definitions, we have

$$\Psi_\theta'(t) = C(\Theta)\mathcal{W}_\theta(t) + \mathcal{Z}_\theta(\hat{x}) \tag{8.39}$$

The corresponding innovation is

$$e_\theta(t) = y(t) - \hat{y}_\theta(t|t-1) = y(t) - C(\Theta)\hat{x}_\theta(t|t-1) \tag{8.40}$$

Therefore

$$\nabla_\theta e_\theta(t) = \nabla_\theta\left[y(t) - C(\Theta)\hat{x}_\theta(t|t-1)\right] = -\Psi_\theta'(t) \tag{8.41}$$

Next we must develop an expression for $\mathcal{W}_\theta$ by differentiating the predictor of Eq. (8.33), that is,

$$\mathcal{W}_\theta(t+1) = \nabla_\theta\hat{x}_\theta(t+1|t) = \nabla_\theta\left[A(\Theta)\hat{x}_\theta(t|t-1) + B(\Theta)u(t) + K(\Theta)e_\theta(t)\right] \tag{8.42}$$

We will again proceed with a componentwise construction of these matrices as above, to illuminate the final results.

We begin by defining these matrices with the $j$th-component of the predicted state as $\hat{x}_j(t|t-1;\theta)$ to obtain

$$\mathcal{W}_\theta'(t) = \begin{bmatrix} \frac{\partial \hat{x}_1(t|t-1;\theta)}{\partial\theta_1} & \cdots & \frac{\partial \hat{x}_{N_x}(t|t-1;\theta)}{\partial\theta_1} \\ \vdots & \cdots & \vdots \\ \frac{\partial \hat{x}_1(t|t-1;\theta)}{\partial\theta_{N_\theta}} & \cdots & \frac{\partial \hat{x}_{N_x}(t|t-1;\theta)}{\partial\theta_{N_\theta}} \end{bmatrix}$$

$$\mathcal{Z}'_\theta(t) = \begin{bmatrix} \dfrac{\partial \mathbf{c}'_1(\theta)}{\partial \theta_1}\hat{x}_\theta(t|t-1) & \cdots & \dfrac{\partial \mathbf{c}'_{N_y}(\theta)}{\partial \theta_1}\hat{x}_\theta(t|t-1) \\ \vdots & \cdots & \vdots \\ \dfrac{\partial \mathbf{c}'_1(\theta)}{\partial \theta_{N_\theta}}\hat{x}_\theta(t|t-1) & \cdots & \dfrac{\partial \mathbf{c}'_{N_y}(\theta)}{\partial \theta_{N_\theta}}\hat{x}_\theta(t|t-1) \end{bmatrix}$$

We again approach the derivation using componentwise derivatives to reach our desired result. Therefore we differentiate the predictor as

$$\begin{aligned} \mathcal{W}_{\theta_i}(t+1) &:= \frac{\partial}{\partial \theta_i}\hat{x}_\theta(t+1|t) = \frac{\partial}{\partial \theta_i}\left[A(\theta)\hat{x}_\theta(t|t-1) + B(\theta)u(t) + K(\theta)e_\theta(t)\right] \\ &= \frac{\partial}{\partial \theta_i}\left[A(\theta)\hat{x}_\theta(t|t-1) + B(\theta)u(t) + K(\theta)y(t)\right. \\ &\qquad \left. - K(\theta)C(\theta)\hat{x}_\theta(t|t-1)\right] \end{aligned} \tag{8.43}$$

Now performing the individual differentials using the product rule as before, we have

$$\begin{aligned} \mathcal{W}_{\theta_i}(t+1) = &\left[\frac{\partial A(\theta)}{\partial \theta_i}\right]\hat{x}_\theta(t|t-1) + A(\theta)\left[\frac{\partial \hat{x}_\theta(t|t-1)}{\partial \theta_i}\right] + \left[\frac{\partial B(\theta)}{\partial \theta_i}\right]u(t) \\ &+ \left[\frac{\partial K(\theta)}{\partial \theta_i}\right]y(t) - \left[\frac{\partial K(\theta)}{\partial \theta_i}\right]C(\theta)\hat{x}_\theta(t|t-1) - K(\theta) \\ &\times \left(\left[\frac{\partial \mathbf{c}'_i(\theta)}{\partial \theta_i}\right]\hat{x}_\theta(t|t-1) + \mathbf{c}'_i(\theta)\left[\frac{\partial \hat{x}_\theta(t|t-1)}{\partial \theta_i}\right]\right) \end{aligned} \tag{8.44}$$

Grouping like terms and using our definitions for $\mathcal{W}_\theta$ and $e_\theta$, we obtain

$$\begin{aligned} \mathcal{W}_{\theta_i}(t+1) = &\left[A(\theta) - K(\theta)C(\theta)\right]\mathcal{W}_{\theta_i}(t) - K(\theta)\left(\left[\frac{\partial \mathbf{c}'_i(\theta)}{\partial \theta_i}\right]\hat{x}_\theta(t|t-1)\right) \\ &+ \left(\left[\frac{\partial A(\theta)}{\partial \theta_i}\right]\hat{x}_\theta(t|t-1) + \left[\frac{\partial B(\theta)}{\partial \theta_i}\right]u(t) + \left[\frac{\partial K(\theta)}{\partial \theta_i}\right]e_\theta(t)\right) \end{aligned} \tag{8.45}$$

Define the $N_x \times N_\theta$-matrix,

$$\mathcal{U}_\theta(\hat{x}_\theta, u, e_\theta) := \left[\nabla_\theta A(\Theta)\right]\hat{x}_\theta(t|t-1) + \left[\nabla_\theta B(\Theta)\right]u(t) + \left[\nabla_\theta K(\Theta)\right]e_\theta(t) \tag{8.46}$$

Substituting componentwise into Eq. (8.45) for each of the definitions ($\mathcal{W}_{\theta_i}$, $\mathcal{Z}_{\theta_i}$, $\mathcal{U}_{\theta_i}$), we obtain

$$\mathcal{W}_{\theta_i}(t+1) = \left[A(\theta) - K(\theta)C(\theta)\right]\mathcal{W}_{\theta_i}(t) + \mathcal{U}_{\theta_i}(\hat{x}_\theta, u, e_\theta) - K(\theta)\mathcal{Z}_{\theta_i}(\hat{x}_\theta(t|t-1)) \tag{8.47}$$

Now expanding over $i = 1, \ldots, N_\theta$, we obtain the final *recursion* for the state predictor gradient weighting matrix as

$$\mathcal{W}_\theta(t+1) = [A(\Theta) - K(\Theta)C(\Theta)]\,\mathcal{W}_\theta(t) + \mathcal{U}_\theta\,(\hat{x}_\theta(t|t-1), u, e_\theta) - K(\Theta)\mathcal{Z}_\theta\,(\hat{x}_\theta(t|t-1)) \tag{8.48}$$

which completes the *RPE* algorithm applied to the innovations model. We summarize the algorithm in Table 8.3 using the simplified notation for clarity. That is, we use the Gauss-Newton approximations as before: $x_\theta(t) \approx x(t)$, $\hat{y}_\theta(t) \approx \hat{y}(t)$, $\Psi_\theta(t) \approx \psi(t)$ and $e_\theta(t) \approx e(t)$. We now apply this algorithm to the *RC*-circuit problem.

***Example 8.2*** Consider the *RC*-circuit problem of Section 5.5 with unknown parameters, $[A, C]$; that is, the underlying Gauss-Markov model has the form

$$x(t) = \Theta_1 x(t-1) + u(t-1) + w(t-1)$$
$$y(t) = \Theta_2 x(t) + v(t)$$

The corresponding innovations model is therefore

$$\hat{x}(t+1|t) = \Theta_1 \hat{x}(t|t-1) + u(t) + \Theta_3 e(t)$$
$$\hat{y}(t|t-1) = \Theta_2 \hat{x}(t|t-1)$$

We seek the adaptive processor that provides joint estimates of both the unknown state and parameters. We applied the *RPE* algorithm available in *MATLAB* [29] to the simulated data with $R_{ww} = 10^{-4}$, $R_{vv} = 4$, $x(0) = 2.5$, and $P(0) = 10^{-6}$ to obtain the adaptive solution. We show the noisy measurement data (noisy line), true measurement (dashed line) and predicted (filtered) measurement. The processor is able to produce a good estimate of the true measurement tracking it closely. The corresponding innovations sequence is shown in Figure 8.2*b* with 5.5% of the samples exceeding the predicted bounds. The performance is validated by investigating the statistics of the innovations sequence. The zero-mean/whiteness test ($0.04 < 0.17/2.35\%$ out) indicates a converged processor as shown in Figure 8.2*c*.

Thus we see that the adaptive innovations processor is capable estimating the unknown parameters (5% initial parametric error) with the following final parameter estimates:

$$\Theta_1 = 0.970 \pm 0.003 \qquad \text{(Process matrix)}$$
$$\Theta_2 = 1.987 \pm 0.140 \qquad \text{(Measurement matrix)}$$
$$\Theta_3 = -0.015 \pm 0.011 \qquad \text{(Gain matrix)}$$
$$\hat{R}_{ee} = 3.371$$

**Table 8.3. Innovations-Based *RPE* Algorithm**

Prediction error

$$e(t) = y(t) - \hat{y}(t|t-1) \quad \text{(Prediction error)}$$
$$\Lambda(t) = \Lambda(t-1) + \gamma(t)\left[e(t)e'(t) - \Lambda(t-1)\right] \quad \text{(Prediction error covariance)}$$
$$R(t) = R(t-1) + \gamma(t)\left[\psi(t)\Lambda^{-1}(t)\psi'(t) - R(t-1)\right] \quad \text{(Hessian)}$$

Parameter estimation

$$\hat{\Theta}(t+1) = \hat{\Theta}(t) + \gamma(t)R^{-1}(t)\psi(t)\Lambda^{-1}(t)e(t) \quad \text{(Parameter update)}$$

State prediction

$$\hat{x}(t+1|t) = A(\Theta)\hat{x}(t|t-1) + B(\Theta)u(t) + K(\Theta)e(t) \quad \text{(State)}$$
$$\hat{y}(t|t-1) = C(\Theta)\hat{x}(t|t-1) \quad \text{(Measurement)}$$

Gradient prediction

$$\mathcal{W}(t) = [A(\Theta) - K(\Theta)C(\Theta)]\,\mathcal{W}(t) + \mathcal{U}(\hat{x}, u, e) - K(\Theta)\mathcal{Z}(\hat{x}) \quad \text{(Weight)}$$
$$\psi'(t) = C(\Theta)\mathcal{W}(t) + \mathcal{Z}(\hat{x}) \quad \text{(Gradient)}$$

$$\text{for } \mathcal{W}(t) := \nabla_\theta \hat{x}(t|t-1), \quad \mathcal{Z}(\hat{x}) := \nabla_\theta\,[C(\Theta)]\,\hat{x}(t|t-1) \quad \text{and}$$

$$\mathcal{U}(\hat{x}, u, e) := [\nabla_\theta A(\Theta)]\,\hat{x}(t|t-1) + [\nabla_\theta B(\Theta)]\,u(t) + [\nabla_\theta K(\Theta)]\,e(t)$$

Initial conditions

$$\hat{x}(0|0), \quad \Theta(0), \quad \Lambda(0), \quad \tilde{R}(0)$$

Summarizing, we have the following:

Criterion: $J = \text{trace } \tilde{P}(t|t)$
Models:
Signal: $x(t) = \Theta_1 x(t-1) + u(t-1) + \Theta_3 e(t-1)$
Measurement: $y(t) = \Theta_2 x(t) + v(t)$
Noise: $w \sim \mathcal{N}(0, 10^{-4})$ and $v \sim \mathcal{N}(0, 4)$
Initial conditions: $\hat{x}(0|0) = 2.5$ and $\tilde{P}(0|0) = 1 \times 10^{-6}$
Algorithm: $\hat{\Theta}(t+1) = \hat{\Theta}(t) + \gamma(t)R^{-1}(t)\psi(t)\Lambda^{-1}(t)e(t)$
Quality: $\Lambda(t) = \Lambda(t-1) + \gamma(t)\left[e(t)e'(t) - \Lambda(t-1)\right]$

This completes the example.

It is possible to return to the general linear Gauss-Markov representation with correlated process and measurement noise to develop the *RPE* adaptive algorithm. However, it is quite complicated, and therefore we refer the interested reader to Appendix 3.B of Ljung [2]. In practice, the innovations model is the usual approach taken for the linear adaptive state-space processors primarily because of the *KSP* relations showing the unique relationship between gain and innovations covariance to the process and noise covariances of the Gauss-Markov model. This completes

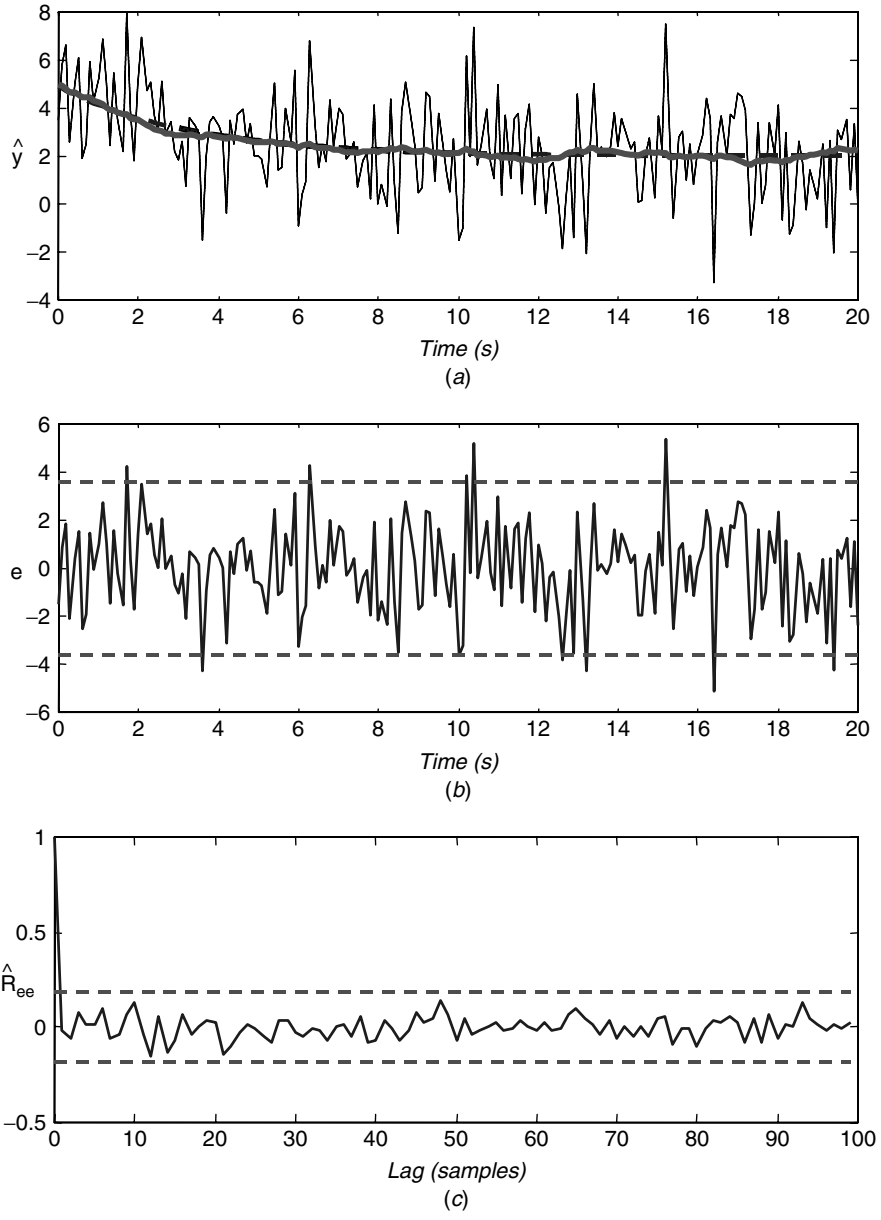



**Figure 8.2.** Adaptive innovations state-space *MBP*: *RC*-circuit problem. (*a*) Noisy (*solid-line*), true (*thick dashed*) and predicted (*solid*) measurements. (*b*) Innovations sequence (5.5% out). (*c*) Zero-mean/whiteness test ($0.04 < 0.17/2.35\%$ out).

the section on adaptive innovations models. Next we investigate the covariance approach to linear adaptive state-space processors.

## 8.4 ADAPTIVE COVARIANCE STATE-SPACE *MBP*

In this section we investigate adaptive covariance-based *MBP* in state-space form. Perhaps these algorithms were the earliest of the adaptive schemes. They were primarily aimed at preventing the divergence of the *MBP*. *Divergence* occurs when the gain becomes too small and the processor "ignores" the incoming data during the update phase. In this case the estimates are erroneous and must be corrected. The remedy for this condition is to increase the gain. Since the gain is (simply) proportional to the noise statistics,

$$\mathcal{K} \propto \frac{R_{ww}}{R_{vv}} \tag{8.49}$$

it can be increased enabling the incoming data to be processed and correcting the updated estimates. It is clear from this relation that increasing the gain can be accomplished by adjusting the noise covariances, that is, increasing $R_{ww}$ or decreasing $R_{vv}$. Performing these adjustments adaptively leads to the covariance forms of the *AMBP*. Mehra [8] discusses the various approaches to covariance-based processing, but some of the earlier work originated by Jazwinski [9].

With this motivation in mind, we develop the adaptive approach of Jazwinski [9]. The main idea is based on the properties of the optimal "tuned" *MBP* of Chapter 5. We know that the optimality condition is based on the statistical properties of the innovations sequence, that is, they must be zero-mean and white. Therefore this processor will adaptively adjust the process noise covariance statistics ($R_{ww}$) to maintain the whiteness of the innovations sequence. The criterion is based on adjusting the process noise covariance amplitude, which in turn modifies the predicted error covariance and therefore increases the processor gain. The requirement imposed is that the innovations predicted by the *MBP* be consistent with their underlying statistics. The covariance-based algorithm adaptively estimates $\hat{R}_{ww}$, producing the most probable innovation sequences. Short innovation sequences are used in order to prevent the adaption algorithm from reaching steady state. This feature is desirable, since $R_{ww}$ is usually used to account for process model errors. Here we concentrate on the process noise covariance scheme, since the measurement noise is usually much easier to estimate in practice and can be fixed to a reasonable value.

Recall from Chapter 5 that the innovations sequence is given by

$$e(t) = y(t) - \hat{y}(t|t-1) = y(t) - C(t)\hat{x}(t|t-1) = C(t)\tilde{x}(t|t-1) + v(t) \tag{8.50}$$

and the corresponding innovations covariance follows directly as

$$R_{ee}(t) = C(t)\tilde{P}(t|t-1)C'(t) + R_{vv}(t) \tag{8.51}$$

Also recall from Table 5.1 that the predicted error covariance is

$$\tilde{P}(t|t-1) = A(t-1)\tilde{P}(t-1|t-1)A'(t-1) + R_{ww}(t-1) \tag{8.52}$$

Now substituting this equation into Eq. (8.51), we obtain

$$\begin{aligned} R_{ee}(t) = {} & C(t)A(t-1)\tilde{P}(t-1|t-1)A'(t-1)C'(t) \\ & + C(t)R_{ww}(t-1)C'(t) + R_{vv}(t) \end{aligned} \tag{8.53}$$

The basic adaptive problem is to find the estimated process noise covariance, $\hat{R}_{ww}$ that produces the *most probable* innovation. To investigate this problem, let us assume that the process covariance matrix is diagonal with equal variances, $\theta$, $R_{ww} = \theta\mathbf{I}$. Then we seek the solution to the problem

$$\max_{\theta} \quad \Pr\,(e(t)) \tag{8.54}$$

for $\Pr(\cdot)$ the probability mass of the innovation. Jazwinski has shown [9] that the maximizing $\theta$ is determined by satisfying the statistical *consistency condition*

$$e^2(t) \approx E\{e^2(t)\} \tag{8.55}$$

Substituting the innovations covariance of Eq. (8.53) for $E\{e^2(t)\}$ and solving, we obtain

$$\begin{aligned} & e^2(t) - E\{e^2(t)\} = e^2(t) \\ & - \Big(C(t)A(t-1)\tilde{P}(t-1|t-1)A'(t-1)C'(t) + \theta C(t)C'(t) + R_{vv}(t)\Big) = 0 \end{aligned} \tag{8.56}$$

or

$$\theta C(t)C'(t) = e^2(t) - E\{e^2(t)|\theta \equiv 0\} \tag{8.57}$$

where we have used the notation $E\{e^2(t)|\theta \equiv 0\}$ to signify that the process noise term in Eq. (8.53) is removed (or null). Clearly, if

$$e^2(t) < E\{e^2(t)|\theta \equiv 0\} \tag{8.58}$$

no process noise input is required. Thus, assuming a scalar measurement ($N_y = 1$, $C \rightarrow \underline{C}$) and solving for $\theta$ in 8.57, we obtain the adaptive noise estimator proposed by Jazwinski as

$$\hat{\theta} = \begin{cases} \dfrac{e^2(t) - E\{e^2(t)|\theta \equiv 0\}}{\underline{C}(t)\underline{C}'(t)}, & > 0 \\ 0, & \text{otherwise} \end{cases} \tag{8.59}$$

This processor is adaptive because the innovations can be monitored using $R_{ee}(t)$. As long as they lie within the $\pm 1.96\sqrt{R_{ee}(t)}$ confidence limits, the process noise covariance is null. However, when they exceed these limits, the process noise covariance is increased, increasing $\tilde{P}(t|t-1)$ and therefore increasing the gain as discussed earlier. It can also be shown that to improve the noise covariance estimates, sample statistics can be used in place of the raw innovations (see [9] for details).[2]

In order to check the statistical consistency of the processor, we compare the squared innovation to its predicted covariance over a fixed interval of data, say $M$:

$$e^2(t+\ell) = R_{ee}(t+\ell) \qquad \text{for} \quad \ell = 1, \ldots, M \tag{8.60}$$

This relation invokes consistency. Note that the resulting *AMBP* will have an inherent time lag of $M$-samples, since $t+M$ measurements must be processed before the actual prediction to time $t+1$ can be made.

This approach can be extended to the *MIMO* case assuming that $R_{ww}$ is diagonal. In this problem, we have the *consistency condition*

$$e^2(t+\ell) = E\{e^2(t+\ell)|R_{ww} \equiv 0\} + \sum_{k=1}^{N_w} \alpha(\ell, k) R_{ww}(k,k) \tag{8.61}$$

where

$$E\{e^2(t+\ell)|R_{ww} \equiv 0\} = C(t)A(t-1)\tilde{P}(t-1|t-1)A'(t-1)C'(t) + R_{vv}(t) \tag{8.62}$$

and

$$\alpha(\ell, k) = \sum_{\ell=1}^{N_w} C(t+\ell)A(t+\ell)A'(t+\ell)C'(t+\ell) \tag{8.63}$$

Defining the *vector*

$$\underline{\xi} := [e^2(t+1) - E\{e^2(t+1)|R_{ww} \equiv 0\}, \ldots, e^2(t+M) - E\{e^2(t+M)|R_{ww} \equiv 0\}]$$

and collecting diagonals of $R_{ww}$ gives

$$\underline{R}'_{ww} := [R_{ww}(1,1), \ldots, R_{ww}(N_w, N_w)]$$

[2]Mehra ([8], [10], [11]) shows that the following recursive (stochastic approximation) estimator can be used as an alternative

$$\hat{R}_{ww}(k+1, k+1) = \hat{R}_{ww}(k) + \frac{1}{k-1}\left(\hat{R}_{ww}(k+1,k) - \hat{R}_{ww}(k)\right)$$

where $k$ is the window length or batch number and $\hat{R}_{ww}(k+1,k)$ is the current estimate based on the $k+1$ batch. The problem with this approach is that the number of free parameters available in $R_{ww}$ is limited. Therefore the *KSP* equations should be used to estimate the *gain* directly ($R^e_{ww} = K R_{ee} K'$) as discussed in Section 8.2.

The *consistency condition* over $M$ measurements becomes

$$\mathcal{A}\underline{R}_{ww} = \underline{\xi} \tag{8.64}$$

for $\mathcal{A}$ the matrix constructed from the elements of Eq. (8.63), that is, $A = [\alpha(\ell, k)]$; $\ell = 1, \ldots, N_w$; $k = 1, \ldots, M$.

Jazwinski [9] has shown that the maximization of the joint probability mass, $\Pr(e(t+1), \ldots, e(t+M))$ leads to the vector solution of process noise covariance diagonals given by

$$\underline{R}_{ww} = \left(\mathcal{A}'\mathcal{A}\right)^{-1}\mathcal{A}'\underline{\xi} \tag{8.65}$$

which gives the vector adaptive solution

$$\underline{\hat{R}}_{ww} = \begin{cases} \underline{R}_{ww}, & > 0 \\ 0, & \text{otherwise} \end{cases} \tag{8.66}$$

subject to the *constraint* that

$$\hat{R}_{ww}(k,k) < 0 \quad \rightarrow \quad \hat{R}_{ww}(k,k) = 0 \tag{8.67}$$

Let us examine the performance of a process noise estimator on the *RC*-circuit problem. We choose to use the innovations model approach and the *RPE* method to construct the adaptive processor.

***Example 8.3*** Again, we choose the *RC*-circuit problem to demonstrate the performance of the *RPE* adaptive state-space *MBP*. Here we assume that all of the circuit parameters are known except the process noise covariance. We develop an adaptive scheme to estimate the noise covariance as the unknown parameter. In this case we use the innovations formulation as before and start with the steady-state gain, $K_{ss} = 8.23 \times 10^{-4}$. We choose the starting value of $R_{ww} = 1.5 \times 10^{-4}$—a 50% error. The processor is able to still produce reasonable estimates of the predicted measurements while tracking the true (noise-free) trajectory as shown in Figure 8.3*a*. The performance is quite good as seen by the apparent whiteness of the innovations sequence (6% out) and corresponding zero-mean/whiteness test results ($0.16 < 0.17/1.56\%$ out) shown in Figure 8.3*b* and 8.3*c*. The noise covariance is estimated in 7 intervals along with the corresponding gain estimates shown in Figure 8.4*a* and *b*. We see that the process noise covariance increases along with the gain. The final values are $\hat{R}_{ww} = 2.3 \times 10^{-3}$ and $\hat{K} = 1.33 \times 10^{-2}$. We can see that the gain has increased from its steady-state value corresponding to the increase in estimated process noise producing the desired effect. We summarize this example in the model-based framework as follows:

Criterion: $J = \text{trace } \tilde{P}(t|t)$

Models:

Signal: $x(t) = 0.97x(t-1) + u(t-1) + w(t-1)$

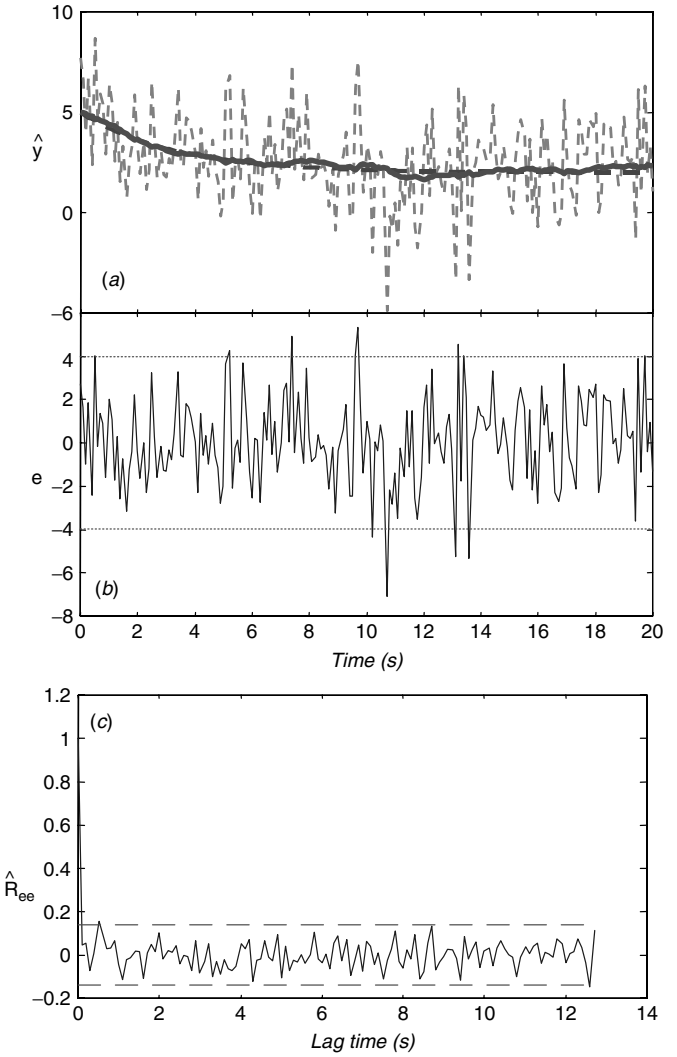



**Figure 8.3.** Adaptive innovations state-space *MBP* for process noise estimation: *RC*-circuit problem. (*a*) Noisy (*dashed*), true (*thick dashed*), and predicted (*solid*) measurements. (*b*) Innovations sequence (6% out). (*c*) Zero-mean/whiteness test ($0.16 < 0.17/1.56\%$ out).

| | |
|---|---|
| Measurement: | $y(t) = 2x(t) + v(t)$ |
| Noise: | $w \sim \mathcal{N}(0, \hat{R}_{ww})$ and $v \sim \mathcal{N}(0, 4)$ |
| Initial conditions: | $\hat{x}(0\|0) = 2.5$ and $\tilde{P}(0\|0) = 1 \times 10^{-6}$ |
| Algorithm: | $\hat{\Theta}(t+1) = \hat{\Theta}(t) + \gamma(t) R^{-1}(t) \psi(t) \Lambda^{-1}(t) e(t)$ |
| Quality: | $\Lambda(t) = \Lambda(t-1) + \gamma(t) \left[ e(t) e'(t) - \Lambda(t-1) \right]$ |

This completes the example.

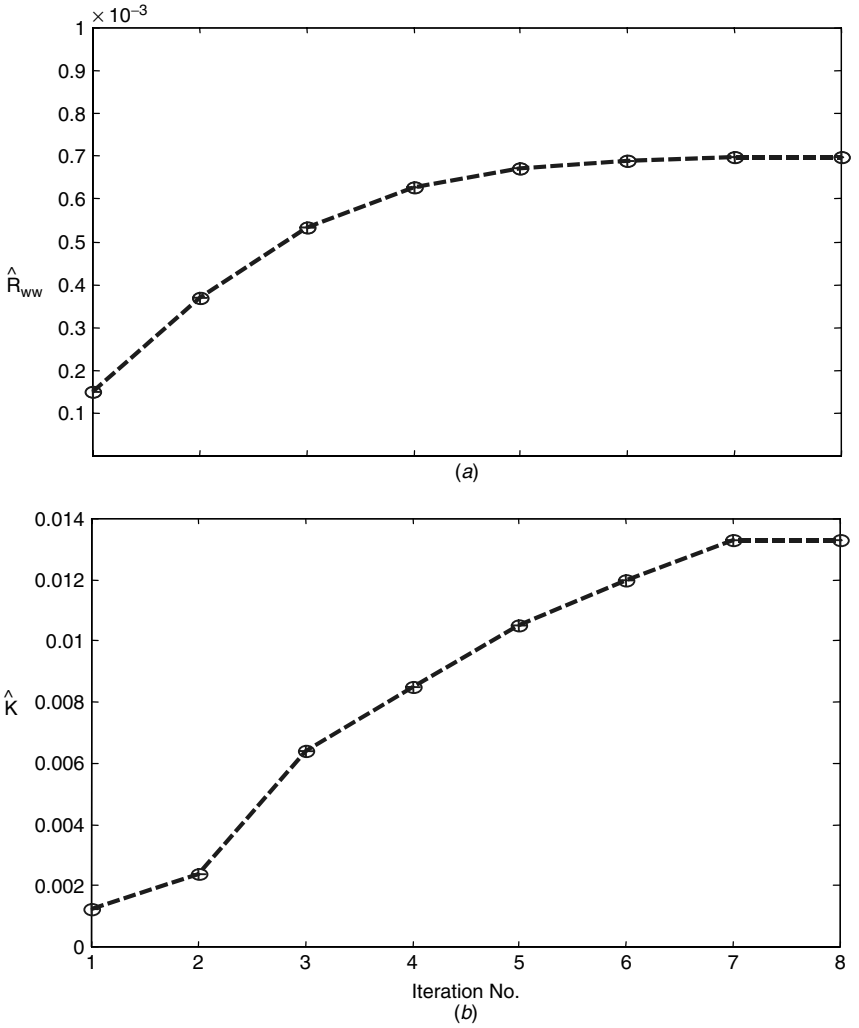



**Figure 8.4.** Adaptive process noise covariance *MBP* for *RC*-circuit problem. (*a*) Estimated process noise. (*b*) Estimated gain.

## 8.5 ADAPTIVE NONLINEAR STATE-SPACE *MBP*

In this section we develop the parametrically adaptive model-based processor for nonlinear state-space systems. The *AMBP* is a *joint estimator/processor*, since it estimates *both* the states as well as the unknown model parameters. It is parametrically adaptive, since it adjusts or "adapts" the model parameters at each time step. A simple diagram of the processor is shown in Figure 8.5 demonstrating the interaction between the state and parametric estimates coupled through the innovations

sequence. In general, this processor can be considered to be a form of identifier, since system identification is typically concerned with the estimation of a model and its associated parameters from noisy measurement data. Usually the model structure is predefined (as in our case), and then a parameter estimator is developed to "fit" parameters according to some error criterion. After completion or during this estimation, the quality of the estimates must be evaluated to decide if the processor performance is satisfactory or equivalently the model adequately represents the data. There are various types (criteria) of identifiers employing many different model (usually linear) structures ([12], [13], [14]). In this section we are primarily concerned with joint estimation in which the models and parameters are usually nonlinear. We will concentrate on developing a parameter estimator capable of on-line operations with nonlinear dynamics.

From our previous discussion in Chapter 6, it is clear that the extended model-based processor *XMBP* (Kalman filter) can satisfy these constraints nicely, so we begin our analysis of the *XMBP* as a parametric estimator closely following the roadmap of Ljung [2] for the linear problem discussed in Section 8.2. The general nonlinear parameter estimator structure can be derived directly from the *XMBP* algorithm in Table 6.3.

To develop the internal structure of the *AMBP*, we start with the *XMBP* equations, augment them with the unknown parameters and then investigate the resulting algorithm. We first define the composite state-vector consisting of the original states, $x(t)$, and the unknown "augmented" parameters represented by $\theta(t)$, that is,

$$\mathcal{X}(t) := \begin{bmatrix} x(t) \\ --- \\ \theta(t) \end{bmatrix} \tag{8.68}$$

where $t$ is the time index, $\mathcal{X} \in R^{(N_x+N_\theta)\times 1}$ and $x \in R^{N_x\times 1}$, $\theta \in R^{N_\theta\times 1}$. Substituting this augmented state vector into the *XMBP* relations of Table 6.3, the following matrix partitions evolve naturally:

$$\tilde{\mathcal{P}}(t|t-1) := \begin{bmatrix} \tilde{P}_{xx}(t|t-1) & | & \tilde{P}_{x\theta}(t|t-1) \\ - & - & - \\ \tilde{P}_{\theta x}(t|t-1) & | & \tilde{P}_{\theta\theta}(t|t-1) \end{bmatrix}, \quad \tilde{P}_{\theta x}(t|t-1) = \tilde{P}'_{x\theta}(t|t-1) \tag{8.69}$$

where $\tilde{\mathcal{P}} \in R^{(N_x+N_\theta)\times(N_x+N_\theta)}$, $\tilde{P}_{xx} \in R^{N_x\times N_x}$, $\tilde{P}_{\theta\theta} \in R^{N_\theta\times N_\theta}$, and $\tilde{P}_{x\theta} \in R^{N_x\times N_\theta}$. This also leads to the partitioning of the gain

$$\mathcal{K}(t) := \begin{bmatrix} K_x(t) \\ --- \\ K_\theta(t) \end{bmatrix} \tag{8.70}$$

for $\mathcal{K} \in R^{(N_x+N_\theta)\times N_y}$, $K_x \in R^{N_x\times N_y}$ and $K_\theta \in R^{N_\theta\times N_y}$.

We must also partition the state and measurement predictions as equations, that is,[3]

$$\mathcal{X}(t|t-1) = \begin{bmatrix} \hat{x}(t|t-1) \\ --- \\ \hat{\theta}(t|t-1) \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{a}[\hat{x}(t-1|t-1), \hat{\theta}(t-1|t-1)] + \mathbf{b}[\hat{x}(t-1|t-1), \hat{\theta}(t-1|t-1), u(t-1)] \\ --- \\ \hat{\theta}(t-1|t-1) \end{bmatrix} \tag{8.71}$$

where the corresponding predicted measurement equation becomes

$$\hat{y}(t|t-1) = \mathbf{c}[\hat{x}(t|t-1), \hat{\theta}(t|t-1)] \tag{8.72}$$

Next we consider the predicted error covariance

$$\begin{aligned} \tilde{P}(t|t-1) = {} & A[\hat{x}(t|t-1), \hat{\theta}(t|t-1)]\tilde{P}(t|t-1) \\ & \times A'[\hat{x}(t|t-1), \hat{\theta}(t|t-1)] + R_{ww}(t-1) \end{aligned} \tag{8.73}$$

which can be written in partitioned form using Eq. (8.69) and the following jacobian process matrix[4]

$$A[\hat{x}, \hat{\theta}] = \begin{bmatrix} A_x[\hat{x}(t|t-1), \hat{\theta}(t|t-1)] & | & A_\theta[\hat{x}(t|t-1), \hat{\theta}(t|t-1)] \\ - & - & - \\ O & | & I_{N_\theta} \end{bmatrix} \tag{8.74}$$

where

$$\begin{aligned} A_x[\hat{x}, \hat{\theta}] &:= \frac{\partial a[\hat{x}, \hat{\theta}]}{\partial x} + \frac{\partial b[\hat{x}, \hat{\theta}]}{\partial x} \\ A_\theta[\hat{x}, \hat{\theta}] &:= \frac{\partial a[\hat{x}, \hat{\theta}]}{\partial \theta} + \frac{\partial b[\hat{x}, \hat{\theta}]}{\partial \theta} \end{aligned} \tag{8.75}$$

with $A \in R^{(N_x+N_\theta)\times(N_x+N_\theta)}$, $A_x \in R^{N_x \times N_x}$, and $A_\theta \in R^{N_x \times N_\theta}$.

Using these partitions, we can develop the *AMBP* directly from the *XMBP* processor in this joint state and "parameter estimation" form. Substituting Eq. (8.74) into the *XMBP* prediction covariance relation of Table 6.3 and using the partition

[3]Note that we have "implicitly" assumed that the parameters can be considered *piecewise constant*, $\Theta(t) = \Theta(t-1)$ or follow a random walk if we add process noise to this representation in the Gauss-Markov sense. However, if we *do* have process dynamics with linear or nonlinear models characterizing the parameters, then they will replace the random walk and the associated jacobians, and so forth, will change from this representation. The *XMBP* can still be applied in this case as well.

[4]Here is where the underlying random walk model enters the structure. The lower block rows of $A_x$ could be replaced by $[A_{\theta x}[\hat{x}, \hat{\theta}] \mid A_{\theta\theta}[\hat{x}, \hat{\theta}]]$, which enables a linear or nonlinear dynamic model to be embedded directly.

defined in Eq. (8.69), we obtain (suppressing the $\hat{x}$, $\hat{\theta}$, time index $t$ notation for simplicity)

$$\tilde{\mathcal{P}}(t|t-1) = \left[\begin{array}{c|c} A_x & A_\theta \\ \hline 0 & I_{N_\theta} \end{array}\right] \tilde{\mathcal{P}}(t-1|t-1) \left[\begin{array}{c|c} A_x & A_\theta \\ \hline 0 & I_{N_\theta} \end{array}\right]' + \left[\begin{array}{c|c} R_{w_x w_x} & 0 \\ \hline 0 & R_{w_\theta w_\theta} \end{array}\right] \tag{8.76}$$

Expanding these equations, we obtain the following set of predicted covariance relations:

$$\tilde{\mathcal{P}}(t|t-1) = \left[\begin{array}{c|c} A_x \tilde{P}_{xx} A_x' + A_\theta \tilde{P}_{\theta x} A_x' + A_x \tilde{P}_{x\theta} A_\theta' + A_\theta \tilde{P}_{\theta\theta} A_\theta' + R_{w_x w_x} & A_x \tilde{P}_{x\theta} + A_\theta \tilde{P}_{\theta\theta} \\ \hline \tilde{P}_{\theta x} A_x' + \tilde{P}_{\theta\theta} A_\theta' & \tilde{P}_{\theta\theta} + R_{w_\theta w_\theta} \end{array}\right] \tag{8.77}$$

The innovations covariance follows from the *XMBP* as

$$R_{ee}(t) = C[\hat{x}, \hat{\theta}] \tilde{\mathcal{P}}(t|t-1) C'[\hat{x}, \hat{\theta}] + R_{vv}(t) \tag{8.78}$$

Now we must use the partitions of $\tilde{\mathcal{P}}$ above along with the measurement jacobian

$$C[\hat{x}, \hat{\theta}] = \left[C_x[\hat{x}(t|t-1), \hat{\theta}(t|t-1)] \;\middle|\; C_\theta[\hat{x}(t|t-1), \hat{\theta}(t|t-1)]\right] \tag{8.79}$$

where

$$\begin{aligned} C_x[\hat{x}, \hat{\theta}] &:= \frac{\partial c[\hat{x}, \hat{\theta}]}{\partial x} \\ C_\theta[\hat{x}, \hat{\theta}] &:= \frac{\partial c[\hat{x}, \hat{\theta}]}{\partial \theta} \end{aligned} \tag{8.80}$$

with $C \in R^{N_y \times (N_x + N_\theta)}$, $C_x \in R^{N_y \times N_x}$, and $C_\theta \in R^{N_y \times N_\theta}$.

The corresponding innovations covariance follows from Eqs. (8.78) and (8.79) as

$$R_{ee}(t) = [C_x \;|\; C_\theta] \left[\begin{array}{c|c} \tilde{P}_{xx} & \tilde{P}_{x\theta} \\ \hline \tilde{P}_{\theta x} & \tilde{P}_{\theta\theta} \end{array}\right] \left[\begin{array}{c} C_x' \\ \hline C_\theta' \end{array}\right] + R_{vv}(t) \tag{8.81}$$

or after expanding it, as

$$R_{ee}(t) = C_x \tilde{P}_{xx} C_x' + C_\theta \tilde{P}_{\theta x} C_x' + C_x \tilde{P}_{x\theta} C_\theta' + C_\theta \tilde{P}_{\theta\theta} C_\theta' + R_{vv} \tag{8.82}$$

$R_{ee} \in R^{N_y \times N_y}$. The gain of the *XMBP* in Table 6.3 is calculated from these partitioned expressions as

$$\mathcal{K}(t) = \begin{bmatrix} K_x(t) \\ - - - \\ K_\theta(t) \end{bmatrix} = \begin{bmatrix} \tilde{P}_{xx} & | & \tilde{P}_{x\theta} \\ - & - & - \\ \tilde{P}_{\theta x} & | & \tilde{P}_{\theta\theta} \end{bmatrix} \begin{bmatrix} C_x \\ - - - \\ C_\theta \end{bmatrix}' R_{ee}^{-1}(t) \tag{8.83}$$

or

$$\mathcal{K}(t) = \begin{bmatrix} K_x(t) \\ - - - \\ K_\theta(t) \end{bmatrix} = \begin{bmatrix} (\tilde{P}_{xx}C_x' + \tilde{P}_{x\theta}C_\theta')R_{ee}^{-1}(t) \\ - - - \\ (\tilde{P}_{\theta x}C_x' + \tilde{P}_{\theta\theta}C_\theta')R_{ee}^{-1}(t) \end{bmatrix} \tag{8.84}$$

where $K \in^{(N_x+N_\theta)\times N_y}$, $K_x \in R^{N_x \times N_y}$, and $K_\theta \in R^{N_\theta \times N_y}$. With the gain determined, the corrected state/parameter estimates follow easily, since the innovations remain unchanged, that is,

$$e(t) = y(t) - \hat{y}(t|t-1) = y(t) - c[\hat{x}(t|t-1), \hat{\theta}(t|t-1)] \tag{8.85}$$

Therefore partitioning the corrected state equations, we have

$$\hat{\mathcal{X}}(t|t) = \begin{bmatrix} \hat{x}(t|t) \\ - - - \\ \hat{\theta}(t|t) \end{bmatrix} = \begin{bmatrix} \hat{x}(t|t-1) \\ - - - \\ \hat{\theta}(t|t-1) \end{bmatrix} + \begin{bmatrix} K_x(t)e(t) \\ - - - \\ K_\theta(t)e(t) \end{bmatrix} \tag{8.86}$$

Finally the corrected covariance expression is easily derived from the following partitions

$$\tilde{\mathcal{P}}(t|t) = \begin{bmatrix} \tilde{P}_{xx} & | & \tilde{P}_{x\theta} \\ - & - & - \\ \tilde{P}_{\theta x} & | & \tilde{P}_{\theta\theta} \end{bmatrix} - \begin{bmatrix} K_x(t) \\ - - - \\ K_\theta(t) \end{bmatrix} [C_x \quad | \quad C_\theta] \begin{bmatrix} \tilde{P}_{xx} & | & \tilde{P}_{x\theta} \\ - & - & - \\ \tilde{P}_{\theta x} & | & \tilde{P}_{\theta\theta} \end{bmatrix} \tag{8.87}$$

Performing the indicated multiplications leads to the final expression

$$\tilde{\mathcal{P}}(t|t) = \begin{bmatrix} \tilde{P}_{xx} - K_x C_x \tilde{P}_{xx} - K_x C_\theta \tilde{P}_{\theta x} & | & \tilde{P}_{x\theta} - K_x C_x \tilde{P}_{x\theta} - K_x C_\theta \tilde{P}_{\theta\theta} \\ - - - & - - - & - - - \\ \tilde{P}_{\theta x} - K_\theta C_x \tilde{P}_{xx} - K_\theta C_\theta \tilde{P}_{\theta x} & | & \tilde{P}_{\theta\theta} - K_\theta C_x \tilde{P}_{x\theta} - K_\theta C_\theta \tilde{P}_{\theta\theta} \end{bmatrix} \tag{8.88}$$

We summarize the parametrically adaptive model-based processor in predictor-corrector form in Table 8.4. We note that this algorithm is *not* implemented in this fashion, it is implemented in the numerically stable U-D-factorized form as in *SSPACK_PC* [15]. Here we are just interested in the overall internal structure of the algorithm and the decomposition that evolves. The simplified structure of the *XMBP* parameter estimator is shown in Figure 8.5. Here we see the basic

**Table 8.4. *AMBP* Algorithm**

*Predictor/corrector form*

Prediction

$$\hat{x}(t|t-1) = a[\hat{x}(t-1|t-1), \hat{\theta}(t-1|t-1)] + b[u(t-1)] \quad \text{(State)}$$

$$\hat{\theta}(t|t-1) = \hat{\theta}(t-1|t-1) \quad \text{(Parameter)}$$

$$\tilde{P}_{xx}(t|t-1) = A_x[\hat{x},\hat{\theta}]\tilde{P}_{xx}(t-1|t-1)A_x'[\hat{x},\hat{\theta}] + A_\theta[\hat{x},\hat{\theta},u]\tilde{P}_{\theta x}(t-1|t-1)A_x'[\hat{x},\hat{\theta}] + A_x[\hat{x},\hat{\theta}]\tilde{P}_{x\theta}(t-1|t-1)A_\theta'[\hat{x},\hat{\theta},u] + A_\theta[\hat{x},\hat{\theta}\tilde{P}_{\theta\theta}(t-1|t-1)A_\theta'[\hat{x},\hat{\theta},u] + R_{w_x w_x}(t-1) \quad \text{(State covariance)}$$

$$\tilde{P}_{\theta\theta}(t|t-1) = \tilde{P}_{\theta\theta}(t-1|t-1) + R_{w_\theta w_\theta}(t-1) \quad \text{(Parameter covariance)}$$

$$\tilde{P}_{x\theta}(t|t-1) = A_x[\hat{x},\hat{\theta}]\tilde{P}_{x\theta}(t-1|t-1) + A_\theta[\hat{x},\hat{\theta},u]\tilde{P}_{\theta\theta}(t-1|t-1) \quad \text{(Cross-covariance)}$$

Innovation

$$e(t) = y(t) - \hat{y}(t|t-1) = y(t) - c[\hat{x}(t|t-1), \hat{\theta}(t|t-1)] \quad \text{(Innovation)}$$

$$R_{ee}(t) = C_x[\hat{x},\hat{\theta}]\tilde{P}_{xx}(t|t-1)C_x'[\hat{x},\hat{\theta}] + C_\theta[\hat{x},\hat{\theta},u]\tilde{P}_{\theta x}(t|t-1)C_x'[\hat{x},\hat{\theta}] + C_x[\hat{x},\hat{\theta}]\tilde{P}_{x\theta}(t|t-1)C_\theta'[\hat{x},\hat{\theta},u] + C_\theta[\hat{x},\hat{\theta},u]\tilde{P}_{\theta\theta}(t|t-1)C_\theta'[\hat{x},\hat{\theta},u] + R_v(t) \quad \text{(Innovation covariance)}$$

Gain

$$K_x(t) = \left(\tilde{P}_{xx}(t|t-1)C_x'[\hat{x},\hat{\theta}] + \tilde{P}_{x\theta}(t|t-1)C_\theta'[\hat{x},\hat{\theta},u]\right) R_{ee}^{-1}(t) \quad \text{(State gain)}$$

$$K_\theta(t) = \left(\tilde{P}_{\theta x}(t|t-1)C_x'[\hat{x},\hat{\theta}] + \tilde{P}_{\theta\theta}(t|t-1)C_\theta'[\hat{x},\hat{\theta},u]\right) R_{ee}^{-1}(t) \quad \text{(Parameter gain)}$$

Correction

$$\hat{x}(t|t) = \hat{x}(t|t-1) + K_x(t)e(t) \quad \text{(State)}$$

$$\hat{\theta}(t|t) = \hat{\theta}(t|t-1) + K_\theta(t)e(t) \quad \text{(Parameter)}$$

$$\tilde{P}_{xx}(t|t) = \tilde{P}_{xx}(t|t-1) - K_x(t)C_x[\hat{x},\hat{\theta}]\tilde{P}_{xx}(t|t-1) - K_x(t)C_\theta[\hat{x},\hat{\theta},u]\tilde{P}_{\theta x}(t|t-1) \quad \text{(State covariance)}$$

$$\tilde{P}_{\theta\theta}(t|t) = \tilde{P}_{\theta\theta}(t|t-1) - K_\theta(t)C_\theta[\hat{x},\hat{\theta},u]\tilde{P}_{x\theta}(t|t-1) - K_\theta(t)C_\theta[\hat{x},\hat{\theta},u]\tilde{P}_{\theta\theta}(t|t-1) \quad \text{(Parameter covariance)}$$

$$\tilde{P}_{x\theta}(t|t) = \tilde{P}_{x\theta}(t|t-1) - K_x(t)C_x[\hat{x},\hat{\theta}]\tilde{P}_{x\theta}(t|t-1) - K_x(t)C_\theta[\hat{x},\hat{\theta},u]\tilde{P}_{\theta\theta}(t|t-1) \quad \text{(Cross-covariance)}$$

Initial conditions

$$\hat{x}(0|0) \quad \tilde{P}(0|0)$$

Jacobians

$$A[\hat{x},\theta] := \frac{\partial}{\partial x}A[x,\theta]\Big|_{\substack{x=\hat{x}(t|t-1)\\ \theta=\hat{\theta}(t|t-1)}} \qquad C[\hat{x},\theta] := \frac{\partial}{\partial x}C[x,\theta]\Big|_{\substack{x=\hat{x}(t|t-1)\\ \theta=\hat{\theta}(t|t-1)}}$$

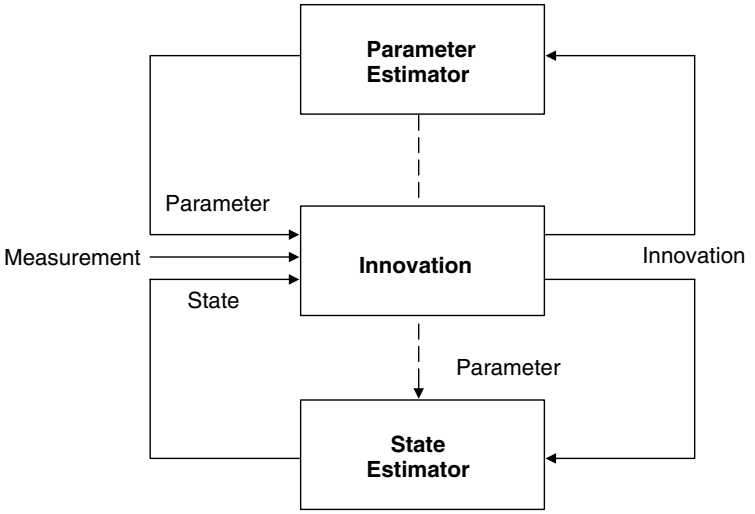



**Figure 8.5.** Nonlinear parametrically adaptive (*AMBP*): Simplified processor structure.

structure of the *AMBP* which consists of two distinct, yet coupled processors: a parameter estimator and a state estimator (filter). The parameter estimator provides estimates that are corrected by the corresponding innovations during each recursion. These estimates are then provided to the state estimator in order to update the model parameters used in the estimator. After both state and parameters estimates are calculated, a new measurement is processed and the procedure continues. A more detailed diagram of the model-based identifier obtained from Table 8.4 is shown in Figure 8.6. This completes the development of the generic *AMBP*.

It is important to realize that besides its numerical implementation, the *AMBP* is simply the *XMBP* with an augmented state vector, thereby implicitly creating the partitions developed above. The implementation of these decomposed equations directly is **not** necessary—just augment the state with the unknown parameters and the *AMBP* evolves naturally from the standard *XMBP* algorithm of Table 6.3. The *AMBP* of Table 8.4 indicates where to locate the partitions. That is, suppose that we would like to extract the submatrix, $\tilde{P}_{\theta\theta}$, but the *XMBP* only provides the overall $(N_x + N_\theta)$ error covariance matrix, $\tilde{P}$. However, locating the lower $N_\theta \times N_\theta$ submatrix of $\tilde{P}$ enables us to extract $\tilde{P}_{\theta\theta}$ directly.

Next let us reconsider the nonlinear system example given in Chapter 6 and investigate the performance of our adaptive *MBP* processor.

***Example 8.4*** Recall the discrete nonlinear trajectory estimation problem of Chapter 6 given by

$$x(t) = (1 - 0.05\triangle T)x(t-1) + 0.04x^2(t-1) + w(t-1)$$

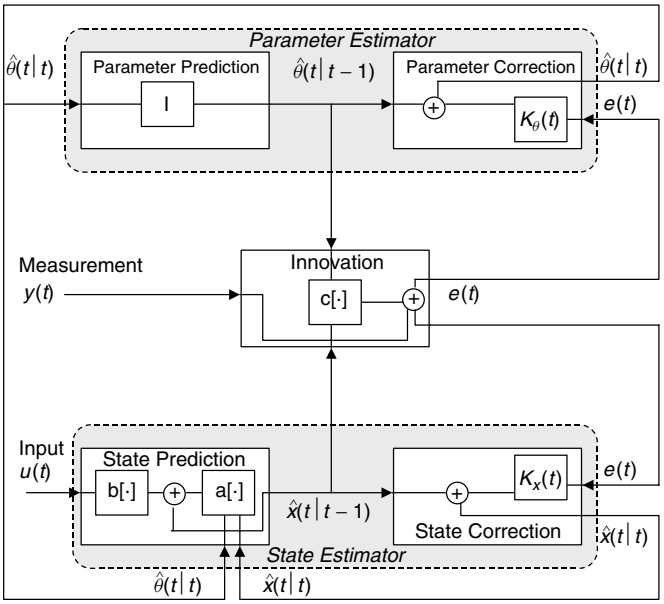



**Figure 8.6.** Nonlinear parametrically adaptive (*AMBP*): Detailed processor structure.

with corresponding measurement model

$$y(t) = x^2(t) + x^3(t) + v(t)$$

where $v(t) \sim \mathcal{N}(0, 0.09)$, $x(0) = 2.0$, $P(0) = 0.01$, $\triangle T = 0.01$ s, and $R_{ww} = 0$.

Here we generalize the problem to the case where the coefficients of the process are unknown leading to the *AMBP* solution. Therefore the process equations for this problem become

$$x(t) = (1 + \theta_1 \triangle T)x(t-1) + \theta_2 x^2(t-1) + w(t-1)$$

with the identical measurement and covariances as before. The true parameters are

$$\Theta_{\text{true}} = [-0.05 \quad 0.04\ ]'$$

The *AMBP* can be applied to this problem by defining the parameter vector as

$$\Theta(t) = \Theta(t-1) \qquad \text{(Constant)}$$

and augmenting it to form the new state vector $\mathcal{X} = [x'\ \theta_1\ \theta_2]'$. Therefore the process model becomes

$$\mathcal{X}(t) = \begin{bmatrix} (1 + \theta_1(t-1)\triangle T)x(t-1) + \theta_2(t-1)\triangle T x^2(t-1) + w(t-1) \\ \theta_1(t-1) \\ \theta_2(t-1) \end{bmatrix}$$

$$y(t) = x^2(t) + x^3(t) + v(t)$$

To implement the *AMBP*, we have the required jacobians

$$A[\mathcal{X}(t-1)] = \begin{bmatrix} [1+\theta_1(t-1)\triangle T + 2\triangle T\theta_2(t-1)x(t-1)] & \triangle T x(t-1) & \triangle T x^2(t-1) \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$C[\mathcal{X}(t-1)] = \begin{bmatrix} 2x(t-1) + 3x^2(t-1) & 0 & 0 \end{bmatrix}$$

Using *SSPACK_PC* [15], we apply the *AMBP* to solve this problem for 1500 samples with $\triangle T = 0.01$ s. Initially we used the starting parameters:

$$\tilde{P}(0|0) = \text{diag}\,[100 \quad 100 \quad 100] \quad \text{and} \quad \hat{x}(0|0) = [1.8 \quad 0 \quad 0]'$$

The results of the *AMBP* run are shown in Figure 8.7. We see the estimated state and parameter estimates in 8.7*b* and 8.7*c*. After a short transient (25 samples), the state estimate begins tracking the true state as evidenced by the innovations sequence in Figure 8.7*c*. The parameter estimates slowly converge to their true values as evidenced by the plots and insets of the figure. The final estimates are

$$\hat{\theta}_1 = -0.0504 \pm 0.0009$$
$$\hat{\theta}_2 = \quad 0.04015 \pm 0.0004$$

Part of the problem for the slow convergence results stems from the lack of sensitivity of the measurement, or equivalently, innovations to parameter variations in this problem. This is implied from the zero-mean/whiteness tests shown in Figure 8.7*c*. The innovations are statistically white but indicate a slight bias ($0.087 > 0.023$). Note the slight downward trend in both the innovations sequence, and it normalized correlation function. The bias is created by the slow parameter convergence. The filtered measurement is also shown in Figure 8.7*c* as well. This completes the *AMBP* example. We summarize the results as follows:

| | |
|---|---|
| Criterion: | $J = \text{trace}\ \tilde{P}(t\|t)$ |
| Models: | |
| Signal: | $x(t) = (1+\theta_1(t-1)\triangle T)x(t-1) + \theta_2(t-1)$ $\triangle T x^2(t-1) + w(t-1)$ |
| Measurement: | $y(t) = x^2(t) + x^3(t) + v(t)$ |
| Noise: | $w \sim \mathcal{N}(0, 10^{-6})$ and $v \sim \mathcal{N}(0, 0.09)$ |
| Initial conditions: | $x(0\|0) = [1.8\ 0\ 0]'$ and $\tilde{P}(0\|0) = \text{diag}[100\ 100\ 100]$ |
| Algorithm: | |
| | $\hat{x}(t\|t) = \hat{x}(t\|t-1) + K_x(t)e(t)$ |
| | $\hat{\theta}(t\|t) = \hat{\theta}(t\|t-1) + K_\theta(t)e(t)$ |
| Quality: | $\tilde{P}(t\|t)$ |

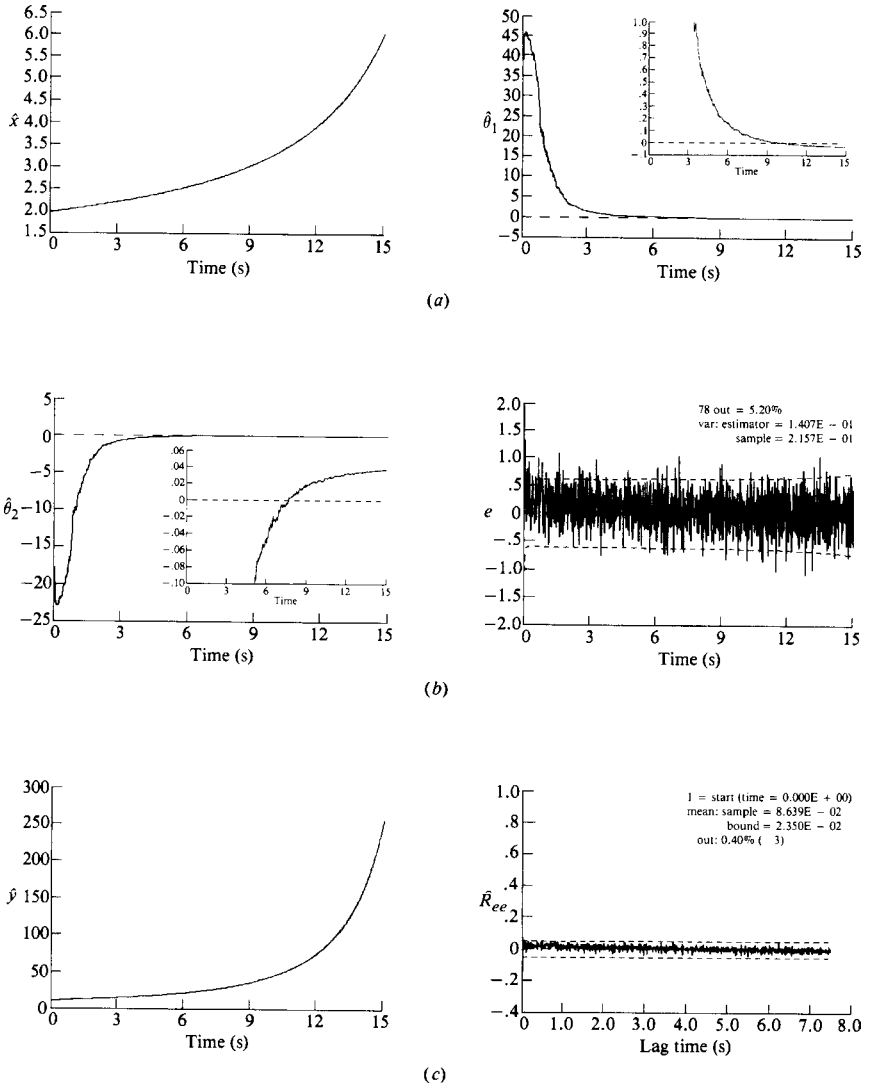



**Figure 8.7.** MBP simulation. (*a*) Estimated state and parameter 1. (*b*) Estimated parameter 2 and innovation (5.2% out). (*c*) Filtered measurement and zero-mean/whiteness test (0.087 > 0.023 and 0% out).

As pointed out by Ljung ([2], [14]), it is important to realize that the *XMBP* is suboptimal as a parameter estimator as compared to the *RPE* method based on the Gauss-Newton (stochastic) descent algorithm. Comparing the processors in this context, we see that if the gradient term

$$[\nabla_\theta K(\Theta)]\, e(t)$$

is incorporated into the *XMBP* (add this term to $A_\theta$), its convergence will be improved approaching the performance of the *RPE* algorithm (see Ljung [14] for details).

We also note in passing that the nonlinear *MBP* in the form developed in Chapter 6 as well as the parametrically adaptive *AMBP* are all heavily employed as *neural networks*. For more details of this important application, see Haykin [16]. Next we consider the development of this processor for an ocean acoustic problem. Here the *AMBP* incorporates an ocean acoustic propagation model into its internal structure for adaptive processing.

## 8.6 CASE STUDY: *AMBP* FOR OCEAN ACOUSTIC SOUND SPEED INVERSION

In this section we discuss a case study developing a *MBP* to invert or estimate the sound speed profile (SSP) from noisy hydrophone pressure-field measurements. The resulting *MBP* is based on the state-space representation of the normal-mode propagation model used to represent the shallow ocean environment. For a detailed development of this model, we refer the interested reader to Chapter 9 where we develop physics-based applications. More specifically the normal-mode representation in ocean acoustics is developed in Section 9.5 in model-based localization. For this study we use data obtained from the well-known Hudson Canyon experiment [17], a noisy shallow water ocean environment. A parametrically adaptive processor is designed to solve this problem. In this application note that the *MBP* is recursing in "space" along the array, not time as in other applications.

In ocean acoustics we are usually concerned with an environmental model of the ocean and how it effects the propagation of sound through this noisy, hostile medium. The problem of estimating the environmental parameters characterizing the ocean medium is called the *ocean tomography* or equivalently the *environmental inversion* problem and has long been a concern because of its detrimental effect on various detection/localization schemes ([18]–[27]).

In this section we concentrate on the design of an *AMBP* to solve the environmental inversion problem while jointly estimating the underlying signals—which we term *model-based inversion*. That is, the *AMBP* is designed to estimate the sound speed profile (SSP) as well as enhance the corresponding modal/pressure-field signals [23]. We briefly review the state-space propagator of the normal-mode model for the Hudson Canyon experiment [24]. Subsequently, we design an *AMBP* to estimate the SSP from noisy experimental hydrophone measurements for this shallow water, ocean acoustic problem and evaluate its performance.

### 8.6.1 State-Space Forward Propagator

We discuss the development of the propagator for the Hudson Canyon experiment performed in 1988 in the Atlantic. Excellent agreement was determined between the

model and measured data indicating a well-known, well-documented shallow water experiment with bottom interaction and yielding ideal data sets for investigating the applicability of a *AMBP* to measured ocean acoustic data [17]. A fixed vertical hydrophone array consisting of 24 phones spaced 2.5 m apart extending from the seafloor up to a depth of about 14 m below the surface was used. Temperature and SSP measurements were made at regular intervals, and the data were collected under carefully controlled conditions in the ocean environment. The normalized horizontal wave number spectrum for a 50 Hz temporal frequency is dominated by five modes occurring at wave numbers between 0.14 to 0.21 $\text{m}^{-1}$ with relative amplitudes increasing with increased wave number. A SNAP [25] simulation was performed and the results agree quite closely, indicating a well-understood ocean environment.

In order to construct the state-space propagator, we require the set of parameters which were obtained from the experimental measurements and processing (wave number spectra). The horizontal wave number spectra were estimated using synthetic aperture processing [26].

For this study we assume a horizontally-stratified ocean of depth $h$ with a *known* source position $(r_s, z_s)$ and assume that the acoustic energy from a point source can be modeled as a trapped wave governed by the wave equation. We obtain the solution to this equation following the approach of Clay [27] and leading to the normal-mode equations.

We will restrict our development to the range independent case. Therefore a solution of the range only function is given by a Hankel function and the *depth* relation is an eigen-equation in $z$ with

$$\frac{d^2}{dz^2}\phi_m(z) + \kappa_z(m)\phi_m(z) = 0, \qquad m = 1, \ldots, M \tag{8.89}$$

whose eigensolutions $\{\phi_m(z)\}$ are the *modal functions* and $\kappa_z$ is the wavenumber in the $z$-direction. These solutions depend on the sound speed profile, $c(z)$, and the boundary conditions at the surface and bottom as well as the corresponding *dispersion* relation

$$\kappa^2 = \frac{\omega^2}{c^2(z)} = \kappa_r^2(m) + \kappa_z^2(m), \qquad m = 1, \ldots, M \tag{8.90}$$

where $\kappa_r, \kappa_z$ are the respective wave numbers in the $r$ and $z$ directions with $c$ the depth-dependent sound speed profile and $\omega$ the harmonic source frequency.

The normal-mode solutions can easily be placed in state-space form, and we refer the interested reader to [28] for the detailed theory. Our approach for the Hudson Canyon experiment is to characterize the model by a Gauss-Markov representation that includes the second-order statistics. The measurement noise can represent the near-field acoustic noise field, flow noise on the hydrophone and electronic noise. The modal noise can represent SSP errors, distant shipping noise, errors in the boundary conditions, sea state effects, and ocean inhomogeneities. By assuming that the horizontal range of the source $r_s$ is known a priori, we can use the Hankel function $H_o(\kappa_r(m)r_s)$ which is the source range solution. Therefore we reduce

the state-space model to that of "depth only" (above). The corresponding Gauss-Markov model for this problem is

$$\frac{d}{dz}\mathbf{x}(z) = A(z)\mathbf{x}(z) + \mathbf{w}(z) \tag{8.91}$$

with the pressure field measurement model given by

$$p(z) = \mathbf{C}'(r_s, z)\mathbf{x}(z) + \mathbf{v}(z) \tag{8.92}$$

where

$$\mathbf{C}'(r_s, z) = [\beta_1(r_s, z) \quad 0| \cdots |\beta_M(r_s, z) \quad 0] \tag{8.93}$$

with

$$\beta_i(r_s, z) = \frac{q\phi_i(z_s)}{\int_0^h \phi_i^2(z)dz} H_o(k_r(i)r_s)$$

The random noise vectors $\underline{w}$ and $\underline{v}$ are assumed gaussian and zero-mean, with respective covariance matrices $R_{ww}$ and $R_{vv}$.

Since our array spatially samples the pressure-field, we choose to discretize the differential state equations using a finite (first) difference approach. That is, for the $m$th mode at the $\ell$th sensor we use

$$\frac{d}{dz}\underline{x}_m(z_\ell) \approx \frac{\underline{x}_m(z_{\ell+1}) - \underline{x}_m(z_\ell)}{\triangle z_\ell} \tag{8.94}$$

where $\triangle z_\ell = z_{\ell+1} - z_\ell$. Therefore we have

$$\underline{x}_m(z_{\ell+1}) = [I + \triangle z_\ell A_m(z_\ell)] \ \underline{x}_m(z_\ell) + \triangle z_\ell \underline{w}(z_\ell)$$

We can now write the discrete-time propagator matrix by combining the terms in the brackets to obtain

$$\underline{x}_m(z_{\ell+1}) = \begin{bmatrix} 1 & | & \triangle z_\ell \\ - - - & - & - \\ \left(\kappa_r^2(m) - \frac{\omega^2}{c(z_\ell)}\right)\triangle z_\ell & | & 1 \end{bmatrix} \underline{x}_m(z_\ell) + \triangle z_\ell \underline{w}(z_\ell) \tag{8.95}$$

Assuming we have a vertical line sensor array to measure the pressure field, the measurement model for the $m$th mode becomes

$$p_m(r_s, z_\ell) = \underline{C}'_m(r_s, z_\ell)\underline{x}_m(z_\ell) + v_m(z_\ell) \tag{8.96}$$

The vertical wave numbers are functions of the sound speed profile through the dispersion relationship that can be further analyzed through knowledge of the SSP. Since the processor will be sequential, it is recursing over depth (space). We

would like it to improve the estimation of the SSP "in-between" the hydrophone sensors. With a state-space processor we can employ two spatial increments in $z$ simultaneously: one for the measurement system $\triangle z_\ell$ and one for the modal state space $\triangle z_j = \triangle z_\ell / N_\triangle$, where $N_\triangle$ is an integer. In order to propagate the states (modes) at $\triangle z_j$ and measurements at $\triangle z_\ell$, we must have values of the SSP at each $\triangle z_j$ (subinterval) as well. Therefore we develop a sound-speed profile model by expanding $c(z)$ in a Taylor series about a nominal depth, say $z_0$, then

$$c(z) \approx \sum_{k=0}^{N} \theta_k(z_o) \frac{(z - z_o)^k}{k!} = \theta_0(z_0) + \theta_1(z_0)(z - z_0) + \cdots + \theta_N(z_0) \frac{(z - z_0)^N}{N!} \tag{8.97}$$

where $\theta_i(z_0) := \frac{\partial^i c(z)}{\partial z^i} \Big|_{z=z_0}, i = 0, \ldots, N$. In this formulation we have a "model" of the SSP of the form

$$c_N(z_\ell, \theta) = \underline{\triangle}'_N(z_\ell) \underline{\theta}(z_\ell) \tag{8.98}$$

where the set of $\{\theta_i(z_\ell)\}$ are only known at a sparse number of depths, $\ell = j, j+1, \ldots, j+N_\theta$. More simply, we have a set of measurements of the SSP measured a priori at specific depths—not necessarily corresponding to all sensor locations $\{z_\ell\}$. Therefore we use these values as initial values to the *AMBP* enabling it to sequentially update the set of parameters $\{\theta_i(z_j)\}$ over the layer $z_{\ell-1} \leq z_j < z_\ell$ until a new value of $\theta_i(z_j)$ becomes available. Then we re-initialize the parameter estimator with this value and continue our SSP estimation until we have recursed through each sensor location. This way we can utilize our measured SSP information in the form of a parameter update and improve the parameter estimates using the processor. Thus we can characterize this SSP representation in an approximate Gauss-Markov model that is nonlinear when we constrain the SSP parameters to the set $\{\theta(z_\ell)\}$, $z_\ell = z_j, \ldots, z_j + N_\theta$, that is,

$$\underline{\theta}_N(z_{\ell+1}) = \begin{cases} \underline{\theta}_N(z_\ell) + \triangle z_j \underline{w}_\theta(z_\ell) & z_\ell < z_j < z_{\ell+1} \\ \underline{\theta}_N(z_\ell) \delta(z_\ell - z_j) & z_\ell = z_j \end{cases} \tag{8.99}$$

where we have $w_\theta \sim \mathcal{N}(0, R_{w_\theta w_\theta})$, $\underline{\theta}_N(z_0) \sim \mathcal{N}(\underline{\theta}_N(0), P_\theta(0))$, and $\underline{\triangle}'_N(z_\ell) = [1 \quad \triangle z_{\ell-1} \ldots \triangle z_{\ell-1}^N / N!]$.

It is this model that we use in our *AMBP* to estimate the sound speed and solve the environmental inversion problem. We can now "augment" this SSP representation into our normal-mode/pressure-field propagation model to obtain an overall system model. The augmented *Gauss-Markov (approximate) model* for $M$-modal functions in the interval $z_\ell < z_j < z_{\ell+1}$ is given by

$$\begin{bmatrix} \underline{x}(z_{\ell+1}) \\ - - - \\ \underline{\theta}_N(z_{\ell+1}) \end{bmatrix} = \begin{bmatrix} A(z_\ell, \theta) & | & 0 \\ - & - & - \\ 0 & | & I_{N+1} \end{bmatrix} \begin{bmatrix} \underline{x}(z_\ell) \\ - - - \\ \underline{\theta}_N(z_\ell) \end{bmatrix} + \begin{bmatrix} \underline{w}(z_\ell) \\ - - - \\ \triangle z_j \underline{w}_\theta(z_\ell) \end{bmatrix}, \quad z_\ell \neq z_j$$

with corresponding measurement model

$$\begin{bmatrix} p(r_s, z_\ell) \\ - - - - \\ c(z_\ell, \theta) \end{bmatrix} = \begin{bmatrix} \underline{C}'(r_s, z_\ell) & | & \underline{0} \\ - & - & - \\ \underline{0} & | & \underline{\triangle}'_N(z_\ell) \end{bmatrix} \begin{bmatrix} \underline{x}(z_\ell) \\ - - - - - \\ \underline{\theta}(z_\ell) \end{bmatrix} + \begin{bmatrix} v_p(z_\ell) \\ - - - - - \\ v_c(z_\ell) \end{bmatrix} \tag{8.100}$$

This model corresponds to the case where *both* acoustics (pressure-field) and sound speed measurements are available at each sensor location.

If we were to further assume that there is not a complete set of SSP measurements but only acoustic pressure-field measurements available at each sensor location, that is, no $c(z_\ell)$ measurements, then the measurement system model consists only of the first row above:

$$p(r_s, z_\ell) = \begin{bmatrix} \underline{C}'(r_s, z_\ell) & | & \underline{0} \end{bmatrix} \begin{bmatrix} \underline{x}(z_\ell) \\ - - - - - \\ \underline{\theta}(z_\ell) \end{bmatrix} + v_p(z_\ell) \tag{8.101}$$

Here we have only a sparse number of sound speed data available, possibly from a past database, and the SSP must be estimated from pressure-field measurements alone—this case corresponds to a tactical situation where it may not be possible to perform current-temperature-density (CTD) measurements in the required time period of engagement. In our formulation, this case implies that we have a "model" of the SSP of the form

$$\hat{c}_N(z_\ell) = \underline{\triangle}'_N(z_\ell)\underline{\theta}(z_\ell) \tag{8.102}$$

where the set of $\{\theta_i(z_\ell)\}$ are only known at a sparse number of depths, $\ell = j, j+1, \ldots, j+N_\theta$. This completes the development of the state-space propagator for the experiment, next we discuss the design of the MBP for the Hudson Canyon experimental data ([23], [24]).

### 8.6.2 Sound-Speed Estimation: *AMBP* Development

In this section we develop a solution to the environmental inversion problem by designing a *AMBP* to estimate the sound speed, online, from noisy hydrophone pressure-field measurements. The *environment inversion problem* can be defined in terms of our previous models as follows:

> **GIVEN** a set of noisy acoustic (pressure-field) measurements $\{p(r_s, z_\ell)\}$ and a set of sound speed parameters $\{\underline{\theta}(z_\ell)\}$. **FIND** the "best" (minimum variance) estimate of the SSP, $\hat{c}(z_\ell)$.

For the Hudson Canyon problem we have a *sparse* set of SSP measurements available at $N_\theta = 9$ depths with complete set of 24 pressure-field measurements. The solution to the inversion problem can be obtained using the "environmentally adaptive" *AMBP*.

To construct the *AMBP*, we use the first two terms ($N = 2$) of the Taylor series expansion of the SSP (piecewise linear) for our model where both $\theta_0$ and $\theta_1$ are space-varying gaussian random functions with specified means and variances, and therefore, through linearity, so is $c(z_\ell)$. Thus our Gauss-Markov model for this problem is given by Eq. (8.100). We will use a spatial sampling interval of $\triangle z_j = \triangle z_\ell/10$ in the state propagation equations as discussed previously. The corresponding *AMBP* estimator evolves from the following functions and jacobians.

The *AMBP* estimator evolves from the algorithm of Table 8.4 where we have that

$$a[x,\theta] = \begin{bmatrix} A(z,\theta) & | & 0 \\ - & - & - \\ 0 & | & I_2 \end{bmatrix}$$

$$c[x,\theta] = \begin{bmatrix} \underline{C}'(r_s, z_\ell) & | & 0 \\ - & - & - \\ 0 & | & \underline{\triangle}'_N(z_\ell) \end{bmatrix}$$

where we have assumed that we have historical or CTD measurements available to extrapolate the sound speed and fit the unknown parameters, $\theta_i(z_o)$. The corresponding system jacobian $A[x,\theta] := \partial a[x]/\partial x$ is given by the following set of relations:

$$a_{m1}[x,\theta] = x_{m1}(z_\ell) + \triangle z_\ell x_{m2}(z_\ell)$$

$$a_{m2}[x,\theta] = \left(\kappa_r^2(m) - \frac{\omega^2}{c_N^2(z_\ell,\theta)}\right)\triangle z_\ell x_{m1}(z_\ell) + x_{m2}(z_\ell)$$

$$a_{2M+1}[x,\theta] = \theta_0(z_\ell)$$

$$a_{2M+2}[x,\theta] = \theta_1(z_\ell) \qquad \text{for } z_{\ell-1} \le z_j \le z_\ell$$

and

$$\frac{\partial a_{m1}[x,\theta]}{\partial x_{m1}} = 1, \quad \frac{\partial a_{m1}[x,\theta]}{\partial x_{m2}} = \triangle z_\ell, \quad \frac{\partial a_{m1}[x,\theta]}{\partial \theta_k} = 0$$

$$\frac{\partial a_{m2}[x,\theta]}{\partial x_{m1}} = \left(\kappa_r^2(m) - \frac{\omega^2}{c_N^2(z_\ell,\theta)}\right)\triangle z_\ell, \quad \frac{\partial a_{m2}[x,\theta]}{\partial x_{m2}} = 1$$

$$\frac{\partial a_{m2}[x,\theta]}{\partial \theta_k} = \frac{2\omega^2 \triangle z_\ell z_\ell^k}{c_N^3(z_\ell,\theta)} x_{m1}(z_\ell), \qquad k = 0, 1$$

$$\frac{\partial a_{2M+1}[x,\theta]}{\partial \theta_0} = 1, \quad \frac{\partial a_{2M+2}[x,\theta]}{\partial \theta_1} = 1$$

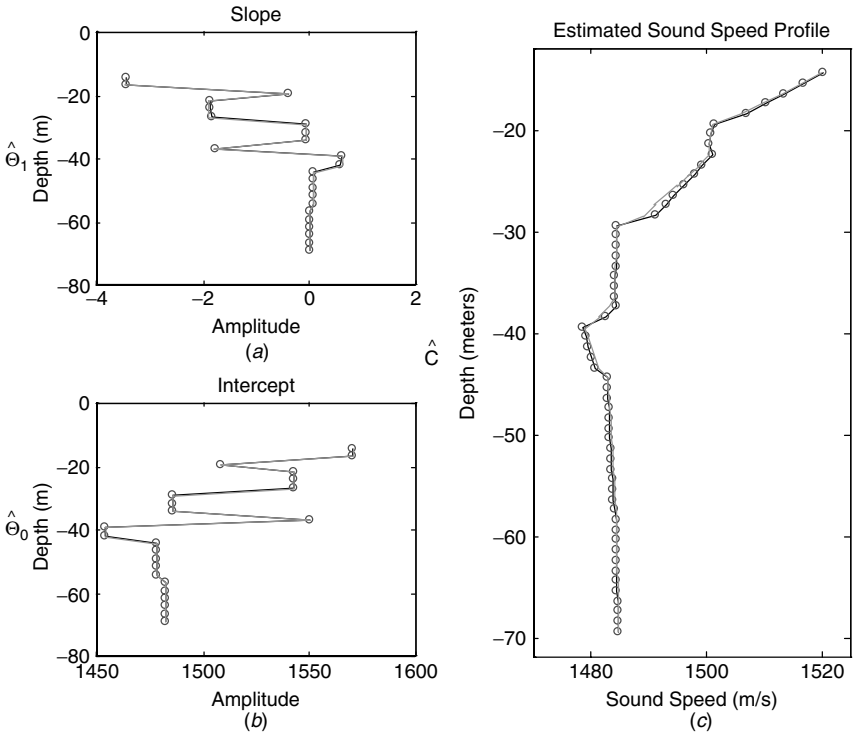



**Figure 8.8.** Model-based inversion: (*a*) Slope estimation. (*b*) Intercept estimation. (*c*) Sound speed estimation.

For the measurement system jacobians we have

$$c_1[x,\theta] = \sum_{i=1}^{M} \beta_i x_{i1} \qquad \text{for } \beta_i(r_s,z) = \frac{q\phi_i(z_s)}{\int_0^h \phi_i^2(z)dz} H_o(k_r(i)r_s)$$

$$c_2[x,\theta] = \sum_{i=0}^{N-1} \triangle z_\ell^i \theta(z_\ell)$$

and

$$\frac{\partial c_1[x,\theta]}{\partial x_{m1}} = \beta_m, \quad \frac{\partial c_1[x,\theta]}{\partial x_{m2}} = 0, \quad \frac{\partial c_1[x,\theta]}{\partial \theta_k} = 0, \quad k = 0,1$$

$$\frac{\partial c_2[x,\theta]}{\partial x_{m1}} = 0, \quad \frac{\partial c_2[x,\theta]}{\partial x_{m2}} = 0, \quad \frac{\partial c_2[x,\theta]}{\partial \theta_k} = z_\ell^k$$

### 8.6.3 Experimental Data Results

We observe the performance of the parametrically adaptive model-based SSP processor. We choose the discrete *AMBP* available in *SSPACK_PC* [15], a third

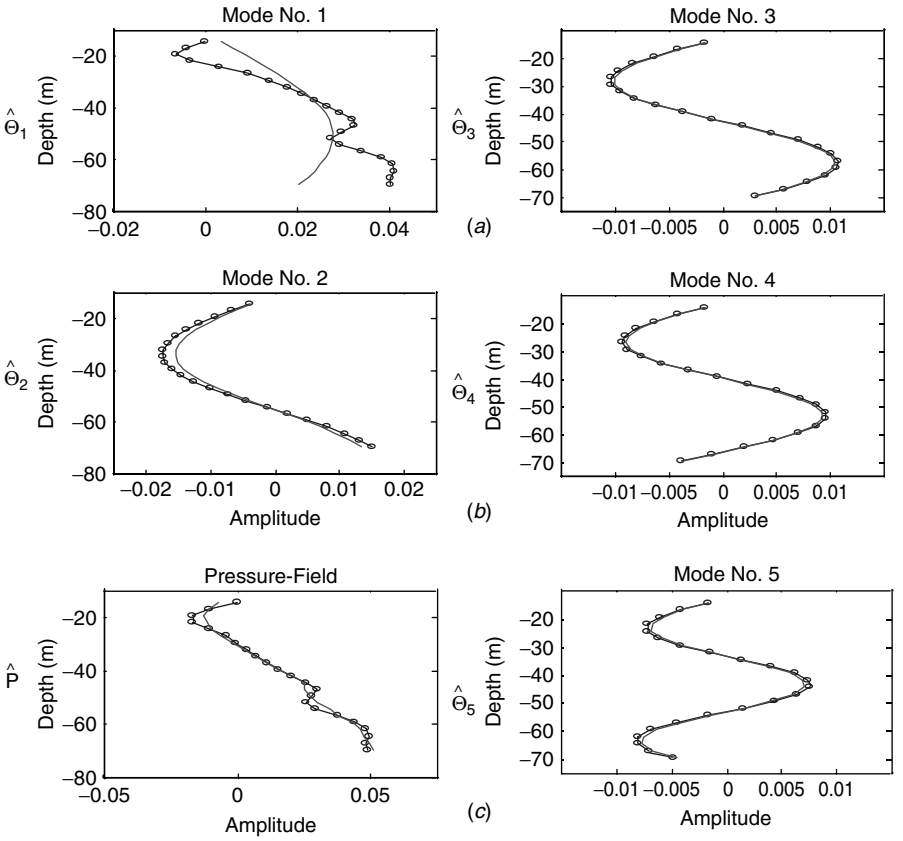



**Figure 8.9.** Model-based enhancement: (a) Estimated modes 1 and 3. (*b*) Estimated modes 2 and 4. (c) Estimated mode 5 and pressure field.

party toolbox in *MATLAB* [29]. Here we use only the acoustic measurements and the 9 sound speed data values $\{c(z_j)\}$, $j = 1, \ldots, 9$ to set hard constraints on the parameter estimator and force it to meet the corresponding $\theta(z_\ell)$ values only when the appropriate depth is reached (see Eq. 8.99). The results of these runs are shown in Figure 8.8. Here we see the estimated SSP parameters and reconstructed SSP. The estimator appears to track the SSP parameters as well as the profile. The standard rms modeling errors (see [23] for details) for the SSP parameters and profile are, respectively, $1.6 \times 10^{-4}$; $2.7 \times 10^{-5}$, and $1.0 \times 10^{-2}$. Since we are using a joint estimator, the enhanced estimates of both modal functions and the pressure field are shown in Figure 8.9 along with the corresponding mean predicted using the SNAP model parameters. Here we see excellent agreement with all of the modes (except mode 1). The standard rms modeling errors for each mode, respectively, are $1.0 \times 10^{-2}$, $1.3 \times 10^{-3}$, $2.3 \times 10^{-4}$, $2.2 \times 10^{-4}$, and $3.5 \times 10^{-4}$. The innovations or residuals are shown in Figure 8.10 where we see that they are zero mean and reasonably white (8.3% out of bounds). The rms standard error for the residuals is

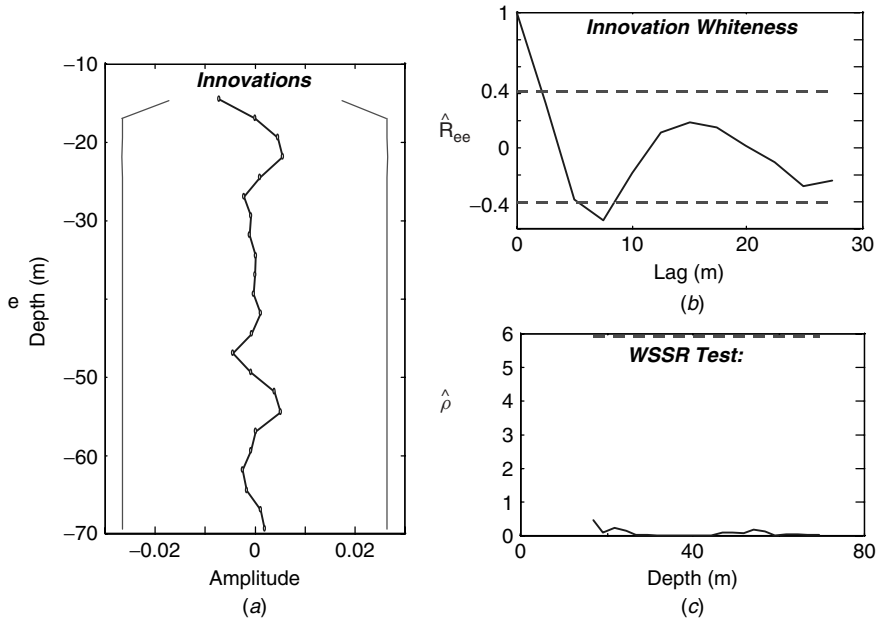



**Figure 8.10.** Model-based residuals: (*a*) Innovation/residual. (*b*) Whiteness test (8.3% out). (*c*) WSSR test (passed).

given by $2.9 \times 10^{-3}$. So we see that the processor is tuned and capable of jointly estimating the SSP and enhancing the modal/pressure field. This completes the case study, and we summarize the results in the model-based framework:

Criterion: $J = \text{trace } \tilde{P}(t|t)$

Models:

Signal:

$$\begin{bmatrix} \underline{x}(z_{\ell+1}) \\ --- \\ \underline{\theta}_N(z_{\ell+1}) \end{bmatrix} = \begin{bmatrix} A(z_\ell, \theta) & | & 0 \\ - & - & - \\ 0 & | & I_{N+1} \end{bmatrix} \begin{bmatrix} \underline{x}(z_\ell) \\ --- \\ \underline{\theta}_N(z_\ell) \end{bmatrix} + \begin{bmatrix} \underline{w}(z_\ell) \\ --- \\ \triangle z_\ell \underline{w}_\theta(z_\ell) \end{bmatrix}, \quad z_\ell \neq z_j$$

$$c(z_\ell, \theta) = \underline{\triangle}'_N \underline{\theta}_N(z_\ell)$$

Measurement:

$$p(r_s, z_\ell) = \begin{bmatrix} \underline{C}'(r_s, z_\ell) & | & \underline{0} \end{bmatrix} \begin{bmatrix} \underline{x}(z_\ell) \\ ---- \\ \underline{\theta}(z_\ell) \end{bmatrix} + v_p(z_\ell)$$

Noise: $w \sim \mathcal{N}(0, R_{ww})$ and $v \sim \mathcal{N}(0, R_{vv})$

Algorithm:

$$\hat{x}(t|t) = \hat{x}(t|t-1) + K_x(t)e(t)$$
$$\hat{\theta}(t|t) = \hat{\theta}(t|t-1) + K_\theta(t)e(t)$$

Quality: $\tilde{P}(t|t)$

We developed an online, adaptive, model-based solution to the environmental inversion problem, that is, a sound speed profile estimation scheme based on coupling the normal-mode propagation model to a functional model of the SSP evolving from Taylor series expansion about the most current sound speed measurement available. The algorithm employed was the parametrically adaptive model-based processor. We showed that the processor performs reasonably well, even for very sparse historical sound speed data.

## 8.7 SUMMARY

In this chapter we developed the adaptive model-based state-space processors, which really evolve from the system identification literature ([1], [2]). After introducing the structure, we developed the generic linear state-space processor enabling us to develop more specific algorithms in the following section using the innovations model. We showed the relationship of the innovations model to the original Gauss-Markov representation and developed an algorithm (*KSP* equations) to perform the transformation. Here a *recursive prediction error approach* led to the final algorithm. Next we investigated covariance state-space algorithms based on the premise that the gain must be adaptively adjusted during the processing using the process noise covariance as the main parameter set of interest. Next, we developed the nonlinear parameter estimator, which is the workhorse for most "physics-based" applications. Here we called the *AMBP* parametrically adaptive, since it essentially solves the joint state and parameter estimation problems. Finally, we investigated the application of the *AMBP* to sound speed estimation in an ocean acoustic application and showed that the processor was able to produce reasonable results even in such a hostile environment.

## *MATLAB* NOTES

*MATLAB* [29] provides commands to design adaptive model-based processors (*AMBP*) for linear time-invariant systems in state-space form using its *Identification Toolbox*. The algorithms are based on the prediction error method (*PEM*) developed in this chapter. The basic state-space structure employed is the innovations model; however, the toolbox also supports "black-box" and arbitrary state-space models parameterizations restricted to time-invariant matrices.

*SSPACK_PC* [15], the third-party *MATLAB* toolbox, is capable of linear and nonlinear time-invariant, time-varying systems using the parametrically adaptive model-based processor of this chapter. With these routines special cases such as the innovations model with completely free parameters is available especially since the *XMBP* and *IX-MBP* are special cases of the recursive prediction error method (*RPEM*) (see: **http//www.techni-soft.net** for more details).

Finally, the REBEL package is also available and written in *MATLAB*. It offers both linear and nonlinear *MBP* as well as the next generation *UMBP* and particle filters (see: (**http://choosh.ece.ogi.edu/rebel**) for more details).

## REFERENCES


1. L. J. Ljung, *System Identification: Theory for the User*, Englewood Cliffs, NJ: Prentice-Hall, 1987.
2. L. J. Ljung and T. Soderstrom, *Theory and Practice of Recursive Identification*, Cambridge: MIT Press, 1983.
3. V. M. Popov, *Hyperstability of Control Systems*, New York: Springer-Verlag, 1973.
4. P. L. Faurre, "Stochastic realization algorithms," in *System Identification: Advances and Case Studies*, R. Mehra and D. Lainiotis eds., New York: Academic Press, 1976.
5. E. Tse and H. Wiennert, "Structure determination and parameter identification for multivariable stochastic linear systems," *IEEE Trans. Auto. Control*, **20**, 603–613, 1975.
6. K. Glover and J. C. Willems, "Parameterizations of Linear Dynamical systems: canonical forms and identifiability," *IEEE Trans. Auto. Control*, **19**, 640–646, 1974.
7. M. Denham, "Canonical forms for identification of multivariable linear systems," *IEEE Trans. Auto. Control*, **19**, 646–656, 1974.
8. R. Mehra, "Approaches to adaptive filtering," *IEEE Trans. Auto. Control*, **17**, 693–698, 1972.
9. A. Jazwinski, *Stochastic Processes and Filtering Theory*, New York: Academic Press, 1970.
10. R. Mehra, "On the identification of variances and adaptive Kalman filtering," *IEEE Trans. Auto. Contr.*, **15**, 175–184, 1970.
11. R. Mehra, "On-line identification of linear dynamic systems with applications to Kalman filtering," *IEEE Trans. Auto. Contr.*, **16**, 12–21, 1971.
12. T. Soderstrom and P. Stoica, *System Identification*, Englewood Cliffs, NJ: Prentice-Hall, 1989.
13. J. P. Norton, *An Introduction to Identification*, New York: Academic Press, 1986.
14. L. Ljung, "Asymptotic behavior of the extended Kalman filter as a parameter estimator for linear systems," *IEEE Trans. Auto. Control*, **24**, 36–50, 1979.
15. J. V. Candy and P. M. Candy, "SSPACK_PC: A model-based signal processing package on personal computers," *DSP Applic.*, **2** (3), 33–42, 1993 (see **http://www.techni-soft.net** for more details).
16. S. Haykin, *Kalman Filtering and Neural Networks*, New York: Wiley, 2001.
17. W. M. Carey, J. Doutt, R. Evans, and L. Dillman, "Shallow water transmission measurements taken on the New Jersey continental shelf," *IEEE J. Oceanic Engr.*, **20** (4), 321–336, 1995.

18. J. V. Candy, *Signal Processing: The Model-Based Approach*. New York: McGraw-Hill, 1986.
19. A. Tolstoy, "Review of matched field processing for environmental inverse problems," *Int. J. Modern Phys.*, **3** (4), 691–708, 1992.
20. A. Tolstoy, *Matched Field Processing for Ocean Acoustics*, Singapore: World Scientific, 1993.
21. C. Feuillade, D. Del Balzo, and M. Rowe, "Environmental mismatch in shallow-water matched-field processing: geoacoustic parameter variability," *J. Acoust. Soc. Am.*, **85** (6), 2354–2364, 1989.
22. R. Hamson, and R. Heitmeyer, "Environmental and system effects on source localization in shallow water by the matched-field processing of a vertical array," *J. Acoust. Soc. Am.*, **86**, 1950–1959, 1989.
23. J. V. Candy and E. J. Sullivan, "Sound velocity profile estimation: A system theoretic approach," *IEEE Trans. Ocean Eng.*, **18** (3), 240–252, 1993.
24. J. V. Candy and E. J. Sullivan, "Model-based environmental inversion: a shallow water application," *J. Acoust. Soc. Am.*, **98**, 1446–1454, 1995.
25. F. B. Jensen and M. C. Ferla, SNAP: the SACLANTCEN normal-mode acoustic propagation model," *SACLANTCEN Report*, **SM-121**, SACLANT Undersea Research Centre, La Spezia, Italy, 1982.
26. N. Yen and W. Carey, "Application of synthetic aperture processing to towed array data," *J. Acoust. Soc. Am.*, **86**, 754–765, 1989.
27. C. S. Clay, and H. Medwin, *Acoustical Oceanography*, New York: Wiley, 1977.
28. J. V. Candy and E. J. Sullivan, "Ocean acoustic signal processing: a model-based approach," *J. Acoust. Soc. Am.*, **92**, 3185–3201, 1992.
29. Mathworks, *MATLAB Users Manual*, Natick, MA: Mathworks, 1990.
30. A. Gelb, *Applied Optimal Estimation*, (Cambridge: MIT Press, 1975).

## PROBLEMS

**8.1** Suppose we are given the following innovations model (in steady state):

$$\hat{x}(t) = a\hat{x}(t-1) + ke(t-1)$$
$$y(t) = c\hat{x}(t) + e(t)$$

where $e(t)$ is the zero-mean, white innovations sequence with covariance, $R_{ee}$.

**(a)** Derive the Wiener solution using the spectral factorization method of Section 8.3.

**(b)** Develop the linear steady-state *MBP* for this model.

**(c)** Develop the parametrically adaptive processor to estimate $k$ and $R_{ee}$.

**8.2** Given the covariance sequence {6.52.5, 0.833, 0.277, 0.0926, 0.0386}, which yields the covariance model: $\{a, b, c, d\} = \{\frac{1}{3}, \frac{5}{4}, 2, \frac{13}{4}\}$

**(a)** Develop the corresponding *KSP* equations, and find $\hat{P}$.

**(b)** Calculate $K$ and $R_{ee}$.

**(c)** Develop the stochastic realization.

**8.3** We are given the following innovations model:

$$\hat{x}(t) = A\hat{x}(t-1) + Ke(t-1)$$
$$y(t) = C\hat{x}(t) + e(t)$$

**(a)** Derive the *RPEM* algorithm based on this model.

**(b)** Let $\{A, C, K, R_{ee}\}$ be scalars; develop the *RPEM* solution to estimate $K$ from noisy data.

**(c)** Let $\{A, C, K, R_{ee}\}$ be scalars, develop the *AMBP* solution to estimate $K$ and compare this algorithm to that of *b* above. Is there any difference? If so, quantify the differences.

**8.4** Using the following *scalar* Gauss-Markov model:

$$x(t) = Ax(t-1) + w(t-1)$$
$$y(t) = C\hat{x}(t) + v(t)$$

with the usual zero-mean, $R_{ww}$ and $R_{vv}$ covariances.

**(a)** Develop the adaptive covariance *MBP* for this problem.

**(b)** Let $\{A, C, K, R_{ee}\}$ be scalars, develop the *AMBP* solution to estimate $A$ from noisy data.

**(c)** Can these algorithms be combined to "tune" the resulting hybrid processor?

**8.5** Suppose that we are given the following structural model:

$$m\ddot{x}(t) + c\dot{x} + kx(t) = p(t) + w(t)$$
$$y(t) = \beta x(t) + v(t)$$

with the usual zero-mean, $R_{ww}$ and $R_{vv}$ covariances.

**(a)** Convert this model to discrete-time using first differences. Using central difference create the discrete Gauss-Markov model. (*Hint*: $\ddot{x}(t) \approx x(t) - 2x(t-1) + x(t-2)/\triangle_t^2$).

**(b)** Suppose that we would like to estimate the spring constant $k$ from noisy displacement measurements, develop the *AMBP* to solve this problem.

**(c)** Transform the discrete Gauss-Markov model to the innovations representation using the *KSP* equations.

**(d)** Solve the parameter estimation problem using the innovations model and the corresponding *RPE* that is, develop the estimator of the spring constant.

**(e)** Suppose that $k$ is known, but the original process noise (discrete model) is unknown. Develop the adaptive covariance estimator to estimate $R_{ww}$ for this problem.

**(f)** Under the same assumptions of $e$ above, develop the *RPE* algorithm to estimate $R_{ww}$.

**8.6** Compare the *RPE* algorithm for the following example using and *ARMAX* model

$$y(t) = -ay(t-1) + bu(t-1) + e(t)$$

with innovations covariance, $R_{ee}$ with the equivalent state-space model of Chapter 2.

**(a)** Write the expressions for the *RPE* in terms of the *ARMAX* model.

**(b)** Write the expressions for the *RPE* in terms of the state-space model.

**(c)** Discuss the similarities and differences in the implementations, if any.

**8.7** Consider tracking a body falling freely through the atmosphere [30]. We assume it is falling down in a straight line towards a radar. The state vector is defined by: $x := [\ z\ \ \dot{z}\ \ \beta\ ]$ where $\beta \sim \mathcal{N}(\mu_\beta, R_{\beta\beta}) = (2000, 2.5 \times 10^5)$ is the ballistic coefficient. The dynamics are defined by the state equations

$$\begin{aligned}
\dot{x}_1(t) &= x_2(t) \\
\dot{x}_2(t) &= \frac{\rho x_2^2(t)}{2x_3(t)} - g \\
\dot{x}_3(t) &= 0 \\
\rho &= \rho_o e^{-\frac{x_1(t)}{k_\rho}}
\end{aligned}$$

where $d$ is the drag deceleration, $g$ is the acceleration of gravity (32.2), $\rho$ is the atmospheric density (with $\rho_o$ $(3.4 \times 10^{-3})$ density at sea level) and $k_\rho$ a decay constant $(2.2 \times 10^4)$. The corresponding measurement is given by

$$y(t) = x_1(t) + v(t)$$

for $v \sim \mathcal{N}(0, R_{vv}) = \mathcal{N}(0, 100)$. Initial values are $x(0) = \mu \sim \mathcal{N}(1065, 500)$, $\dot{x}(0) \sim \mathcal{N}(-6000, 2 \times 10^4)$ and $P(0) = \text{diag}[p_o(1,1), p_o(2,2), p_o(3,3)] = [500,\ 2 \times 10^4,\ 2.5 \times 10^5)$.

**(a)** Is this an *AMBP*? If so, write out the explicit algorithm in terms of the parametrically adaptive algorithm of this chapter.

**(b)** Develop the *XMBP* for this problem and perform the discrete simulation using *MATLAB*.

**(c)** Develop the *LZ-MBP* for this problem and perform the discrete simulation using *MATLAB*.

**8.8** As stated in the chapter, the *XMBP* convergence can be improved by incorporating a gain gradient term in the system jacobian matrices, that is,

$$A_\theta^*[x,\theta] := A_\theta[x,\theta] + [\nabla_\theta K_x(\Theta)]\, e(t) \quad \text{for} \quad \mathcal{K} := [K_x \mid K_\theta]$$

**(a)** By partitioning the original $N_x \times N_\theta$ jacobian matrix, $A_\theta[x,\theta]$, derive the general "elemental" recursion; that is, show that

$$A_\theta^*[i,\ell] = \nabla_{\theta_\ell} a_i[x,\theta] + \sum_{j=1}^{N_y} \nabla_{\theta_\ell} k_x(i,j)\; e_j(t); i = 1,\ldots,$$
$$N_x; \ell = 1,\ldots,N_\theta$$

**(b)** Suppose that we would like to implement this modification, does there exist a numerical solution that could be used? If so, describe it.

**8.9** Parameter estimation can be performed directly when we are given a nonlinear measurement system such that

$$\mathbf{y} = \mathbf{h}(\theta) + \mathbf{v}$$

where $\mathbf{y}, \mathbf{h} \in \mathcal{R}^{N_y \times 1}$ and $\theta \sim \mathcal{N}(\mathbf{m}_\theta, R_{\theta\theta})$, and $\mathbf{v} \sim \mathcal{N}(0, R_{vv})$.

**(a)** From the a posteriori density, $\Pr(\theta|\mathbf{y})$ derive the *MAP* estimator for $\theta$.

**(b)** Expand $\mathbf{y} = \mathbf{h}(\theta)$ in a Taylor series about $\theta_o$ and incorporate the first-order approximation into the *MAP* estimator (approximate).

**(c)** Expand $\mathbf{y} = \mathbf{h}(\theta)$ in a Taylor series about $\theta_o$ and incorporate the second-order approximation into the *MAP* estimator (approximate).

**(d)** Develop and iterated version of both estimators in $(b)$ and $(c)$. How do they compare?

**(e)** Use the parametrically adaptive formulation of this problem, assuming that the measurement model is time-varying. Construct the *AMBP* assuming that $\theta$ is modeled by a random walk. How does this processor compare to the iterated versions?

**8.10** Develop a *second-order XMBP* for the scalar (approximate) Gauss-Markov model given by

$$x(t) = a[x(t-1)] + w(t-1)$$
$$y(t) = c[x(t-1)] + v(t)$$

with $w$, $v$ zero-mean, white gaussian with variances $R_{ww}$ and $R_{vv}$.

**(a)** Perform the appropriate linearizations and now keep the second-order terms in the Taylor series. Let $x^*(t)$ be the reference trajectory and develop the second-order *XMBP* algorithm by letting $x^*(t) \rightarrow \hat{x}(\cdot)$, that is, derive the second-order *XMBP* for this system.

**(b)** Suppose that there is an unknown parameter, $\theta$, in $a[x] \rightarrow a[x;\theta]$, using the second-order *XMBP* developed above construct the second-order *AMBP* for this problem.

**(c)** Compare this processor with the *AMBP*. Discuss the similarities and differences.

**8.11** Suppose that we are asked to solve a detection problem, that is we must "decide" whether a signal is present or not according to the following binary hypothesis test:

$$\begin{aligned}\mathcal{H}_o &: y(t) = v(t) \qquad \text{for} \quad v \sim \mathcal{N}(0, R_{vv}) \\ \mathcal{H}_1 &: y(t) = s(t) + v(t)\end{aligned}$$

The signal is modeled by a Gauss-Markov model

$$s(t) = a[s(t-1)] + w(t-1) \qquad \text{for} \quad w \sim \mathcal{N}(0, R_{ww})$$

**(a)** Calculate the *likelihood-ratio* defined by

$$\mathcal{L}(Y(N)) := \frac{\Pr(Y(N)|\mathcal{H}_1)}{\Pr(Y(N)|\mathcal{H}_o)}$$

where the measurement data set is defined by $Y(N); = \{y(0), y(1), \ldots, y(N)\}$. Calculate the corresponding threshold and construct the detector (binary hypothesis test).

**(b)** Suppose that there is an unknown but deterministic parameter in the signal model, that is,

$$s(t) = a[s(t-1); \theta(t-1)] + w(t-1)$$

Construct the "composite" likelihood ratio for this case. Calculate the corresponding threshold and construct the detector (binary hypothesis test). (*Hint*: Use the *AMBP* to jointly estimate the signal and parameter.)

**(c)** Calculate a sequential form of the likelihood ratio above by letting the batch of measurements, $N \rightarrow t$. Calculate the corresponding threshold and construct the detector (binary hypothesis test). Note there are two thresholds for this type of detector.

**8.12** Angle modulated communications including both frequency modulation (FM) and phase modulation (PM) are basically nonlinear systems from the

model-based perspective. They are characterized by high-bandwidth requirements, and their performance is outstanding in noisy environments. Both can be captured by the *transmitted* measurement model:

$$s(t) = \sqrt{2P}\sin\left[\omega_c t + k_p m(t)\right] \qquad \text{(PM)}$$

or

$$s(t) = \sqrt{2P}\sin\left[\omega_c t + 2\pi k_f \int_{-\infty}^{t} m(\tau)d\tau\right] \qquad \text{(FM)}$$

where $P$ is a constant, $\omega_c$ is the carrier frequency, $k_p$ and $k_f$ are the deviation constants for the respective modulation systems and of course, $m(t)$, is the message model. *Demodulation* to extract the message from the transmission is accomplished by estimating the phase of $s(t)$. For FM, the recovered phase is differentiated and scaled to extract the message, while PM only requires the scaling.

Suppose that the message signal is given by the Gauss-Markov representation

$$m(t) = -\alpha m(t-1) + w(t-1)$$
$$y(t) = s(t) + v(t)$$

with both $w$ and $v$ zero-mean, gaussian with variances, $R_{ww}$ and $R_{vv}$.

**(a)** Construct a receiver for the PM system using the *XMBP* design.

**(b)** Construct an equivalent receiver for the FM system.

**(c)** Assume that the message amplitude parameter $\alpha$ is unknown; construct the *AMBP* receiver for the PM system to jointly estimate the message and parameter.

**(d)** Under the same assumptions as $(c)$, construct the *AMBP* receiver for the FM system to jointly estimate the message and parameter.

**(e)** Compare the receivers for both systems. What are their similarities and differences?

# 9

# APPLIED PHYSICS-BASED PROCESSORS

## 9.1 *MBP* FOR REENTRY VEHICLE TRACKING

The tracking of space vehicles from launch to splash down and extracting critical information about the vehicle dynamics is a problem of high concern whether it be a Space Shuttle launched to deliver a new communications satellite or a reentry vehicle launched to analyze its flight performance. In any case, the basic question of interest is how to extract the vehicle dynamics from noisy and incomplete trajectory information gathered by various independent radar sensors. Primarily this section is aimed at applying model-based signal processing approaches to investigate noisy data sets and extract the desired vehicular information. In particular, we are interested in processing and analyzing tracking data and radar signatures of the ballistic trajectories of reentry vehicles from both simulated and measured data.

Radar can be used as a sensor to measure the desired signatures revealing various details of the vehicular structure. A vehicle in ballistic flight exhibits an angular motion that depends upon its mass properties and deployment conditions. The main parameters of interest are spin and precession rates, momentum attack angle (angle between velocity and angular momentum vectors) and precession half-cone angle as shown in Figure 9.1. Another signature of interest is termed either "nutation", "wobble" or "micro dynamics" referring to a small dynamic motion superimposed on precession by very small mass imbalances [1].

If a tracking radar possesses enough spatial resolution, which is determined by its inherent design parameters, then it can focus energy on the vehicle skin surface and resolve its returns to identify major features. In essence, what we are describing

*Model-Based Signal Processing*, by James V. Candy

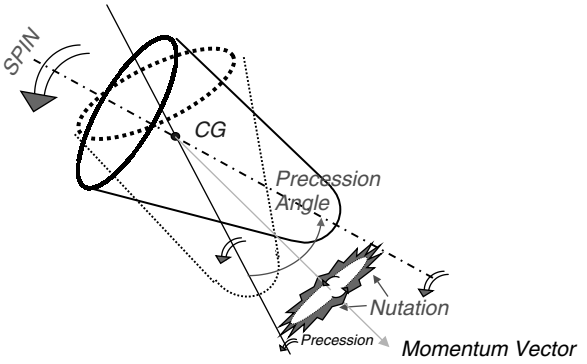



**Figure 9.1.** RV motion dynamics.

is the ability of the radar to image the vehicle or display its major features in order to infer its dynamical motion from its scattered returns. Referring back to the figure, it is clear that the critical parameters are described by vehicular motion relative to its centerline.

In this section we investigate the development of techniques to extract information about reentry vehicle (RV) dynamics from noisy radar cross-sectional (RCS) measurement data. We develop a simplified signal processing model of the radar measurements based on the underlying physics and dynamics. Signal processing techniques are then developed to extract the desired RV dynamical parameters. This suite of techniques includes trajectory motion compensation, environmental transfer function estimation and its inverse, instantaneous spectrogram estimation using parametric autoregressive moving average (ARMA) models, demodulation, and event detection schemes. The overall processing is applied to both synthesized and raw flight data demonstrating performance.

### 9.1.1 RV Simplified Dynamics

The equations of motion for the RV can be described in terms of an axially spinning body [1]. The basic equations of motion follow the conservation of angular momentum principle as found in detailed references [2]. The *precession rate*, is

$$\dot{\psi}(t) = \frac{I_{x|z}}{\cos\theta_c}\omega \tag{9.1}$$

where $\omega$ is the spin rate (deg/s), $\theta_c$ is the precession half-cone angle (deg), and $I_{x|z}$ is the moment of inertia ratio (radians). The corresponding Doppler velocity is given by

$$v_d = r\omega\sin\alpha \tag{9.2}$$

with $r$ the spin radius (radius from spin axis to scatterer; and $\alpha$ the aspect angle of the RV center of gravity (CG) relative to the radar line of site (RLOS).

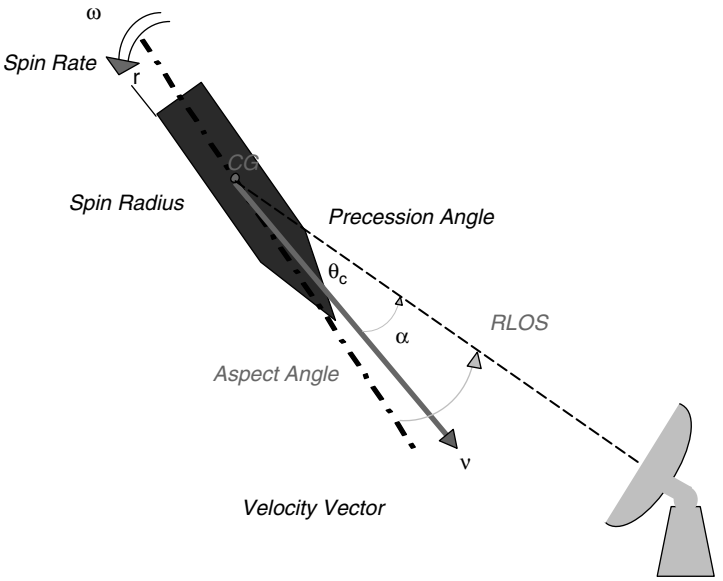



**Figure 9.2.** Typical radar measurement geometry of reentry vehicle.

The aspect angle of the RLOS to the vehicular angular momentum vector is given in terms of the average of the maximum angle, $\alpha_{\max}$, and minimum angle, $\alpha_{\min}$,

$$\alpha = \frac{\alpha_{\max} + \alpha_{\min}}{2} \tag{9.3}$$

with corresponding precession (half-cone) angle obtained as

$$\theta_c = \left| \frac{\alpha_{\max} - \alpha_{\min}}{2} \right| \tag{9.4}$$

We define the corresponding output signal-to-noise ratio (SNR) as the ratio of *target energy*, $\sigma_s^2$, to noise energy $\sigma_v^2$,

$$SNR = \frac{\sigma_s^2}{\sigma_v^2} \tag{9.5}$$

The typical radar geometry defining these terms is shown in Figure 9.2.

Motion solutions of the underlying dynamical equations are used to perform trajectory corrections. A particular vehicle has a characteristic exoatmospheric radar signature depending on its static radiation pattern. The radar cross section is derived from the raw range data by first correcting for the trajectory and then using the RV static pattern (RCS vs. angle) to yield the dynamic RCS data. The

enhanced/corrected RCS data is essentially the "starting" point of this section. We will develop techniques to process this RCS data and extract the parameters of interest. First we develop a signal processing model of a typical launch and analyze its performance. This representation is incorporated into a simulation capable of synthesizing the dominant characteristics of the data measurements. It enables us to "test" the various processing algorithms developed on this known synthetic data. Finally after further analysis, the approach is applied to flight data.

### 9.1.2 Signal Processing Model

We develop a signal processing model of the radar tracking/motion parameter extraction problem. First we cast the problem into a simplified framework and motivate the model. Next we analyze typical properties from a trajectory simulation to motivate a signal representation of the acquired data and then outline the signal processing approach.

The propagation of both the transmitted and reflected waves through the atmosphere (including the ionosphere) can result in significant modification in the electromagnetic (EM) wave parameters. The key atmospheric effects are signal attenuation due to amplitude scintillation, propagation delay, and rotation of the polarized wave. These effects are localized in both time and space. The scintillation and delay are ionospheric effects which can cause pulse distortion or spreading, since the ionosphere acts as a linear filter [3], [4]. The rotation is the effect of the ionosphere on a linear polarized wave producing a rotation in the wave orientation angle. The basic result is that the long propagation path through the dense atmosphere causes significant attenuation and refraction of the EM wave.

Our signal processing model of this propagation is depicted in Figure 9.3. Here we see that the radar transmits an EM wave that propagates through the atmosphere and is distorted by the ionospheric aberration before illuminating the RV. On reception, the RV scatterers or re-transmits the EM energy based on its scattering function back toward the radar receiver where it again is distorted by the ionosphere and attenuated through the atmosphere before being received by the antenna. Naturally radar systems are designed to increase the *SNR* during every possible operation to compensate for these effects. We incorporate (simply) these effects in our signal processing model simulation. Before we investigate the properties of the RV that we must extract, let us examine a typical trajectory obtained from our simulator.

Consider the following signal processing representation of the RV response received at the radar. Suppose we model the dynamics of the RV signal as [2]

$$s(t) = A_p \sin(2\pi f_p t + \phi_p)\left[1 + \sum_{k=1}^{N_v} A_v(k)\sin(2\pi f_v(k)t + \phi_v(k))\right]$$
$$z(t) = s(t) + c(t) + v(t) \tag{9.6}$$

where $z(t)$ is the measured radar return (RCS); $s(t)$ is the RV signal model including precession and nutation $v$; $c(t)$ is the trajectory (trend); $v(t)$ is gaussian random

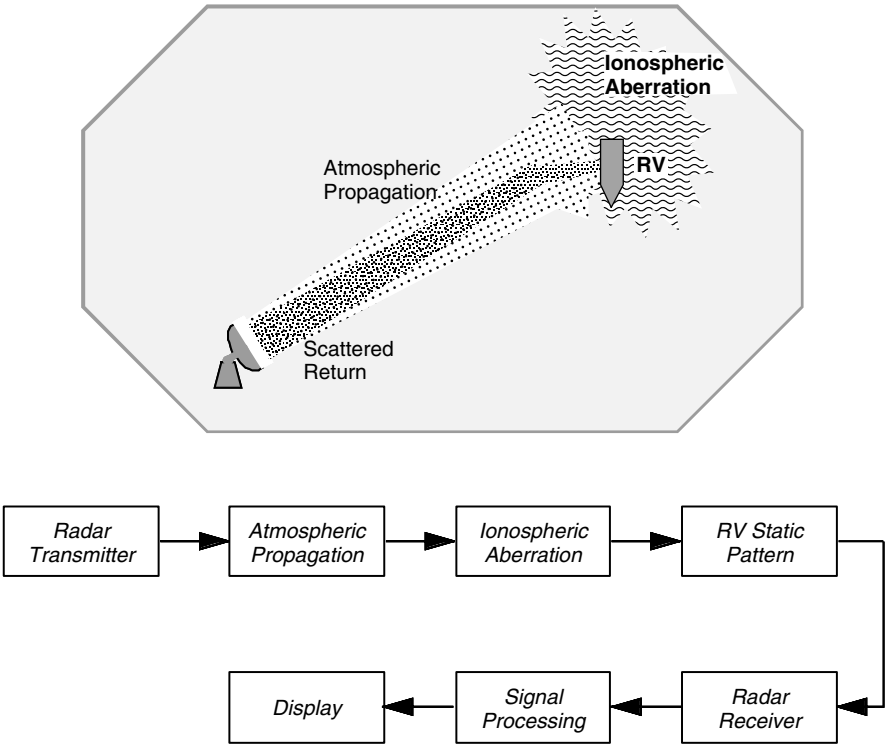



**Figure 9.3.** Signal model of radar transmission, atmospheric propagation, ionospheric aberration, RV scattering, and radar reception.

noise, $\mathcal{N}(0, \sigma_v^2)$; $A_p$, $A_v$ are the precession and nutation amplitudes; $f_p$, $f_v$ are the precession and nutation frequencies; and $\phi_p$, $\phi_v$ are the precession and nutation phase.

It can be shown [5] that by expanding this expression, using standard trigonometric identities and grouping terms, we obtain

$$s_k(t) = \sum_{k=1}^{N_v} c_{14}(k)\cos\left(2\pi(f_p - f_v(k))\,t\right) + c_{41}(k)\cos\left(2\pi(f_p + f_v(k))t\right) + c_{23}(k)\sin\left(2\pi(f_p + f_v(k))\,t\right) + c_{32}(k)\cos\left(2\pi(f_p - f_v(k))t\right) \tag{9.7}$$

where

$$c_{14}(k) = \frac{c_1(k) + c_4(k)}{2}, \quad c_{41}(k) = \frac{c_1(k) - c_4(k)}{2}$$
$$c_{23}(k) = \frac{c_2(k) + c_3(k)}{2}, \quad c_{32}(k) = \frac{c_2(k) - c_3(k)}{2} \tag{9.8}$$

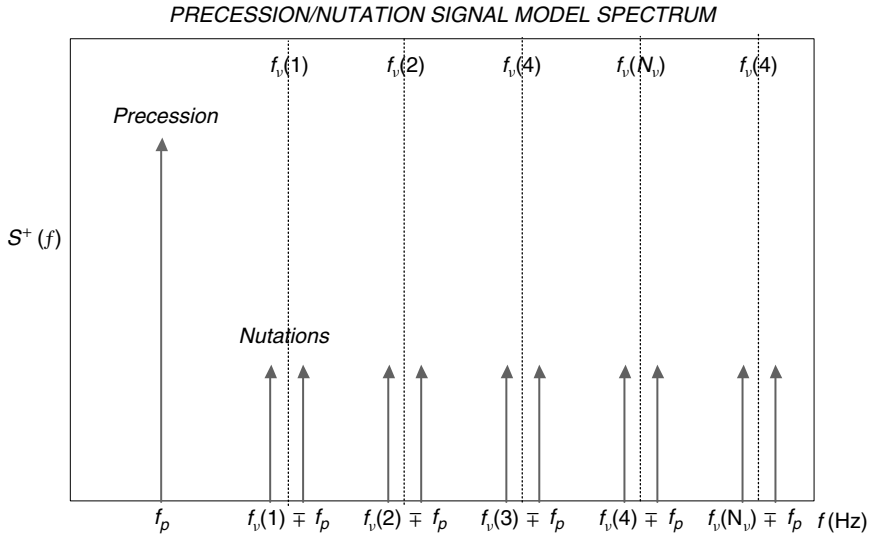



**Figure 9.4.** Precession/nutation spectrum.

and

$$c_1(k) = A_p A_\nu(k)\cos(\phi_p)\cos(\phi_\nu(k)), \quad c_2(k) = A_p A_\nu(k)\sin(\phi_p)\cos(\phi_\nu(k))$$
$$c_3(k) = A_p A_\nu(k)\cos(\phi_p)\sin(\phi_\nu(k)), \quad c_4(k) = A_p A_\nu(k)\sin(\phi_p)\sin(\phi_\nu(k))$$

Our final signal model for precession and nutation is given by

$$s(t) = A_p \sin(2\pi f_p t + \phi_p) + s_k(t) \tag{9.9}$$

with $s_k(t)$ given in Eq. (9.7) above.

Now, if we take the Fourier transform of Eq. (9.9) and consider only the positive frequencies ($f > 0$), then we obtain the expression for the *precession/nutation* signal spectrum [5] as

$$\begin{aligned} S^+(f) = &\left[A_p(\sin\phi_p + j\cos\phi_p)\right]\delta(2\pi(f - f_p)) \\ &+ \sum_{k=1}^{N_\nu}[c_{14}(k) - jc_{32}(k)]\delta(2\pi(f - (f_p - f_\nu(k))) \\ &+ [c_{41}(k) - jc_{23}(k)]\delta(2\pi(f - (f_p + f_\nu(k))) \end{aligned} \tag{9.10}$$

We display the positive spectrum in Figure 9.4 for $N_\nu$ pairs assuming equal amplitudes for simplicity. Note that we expect the major peak of the lowest

frequency to occur at the precession frequency, $f_p$, and a set of $N_\nu$-pairs of peaks at the lower and upper side band frequencies induced by the nutations, that is, $f_\nu(k) \pm f_p; k = 1, \ldots, N_\nu$. It is important to note that we cannot recover the nutation frequencies directly from this spectrum without further processing, since only the side bands are available.

We still must consider the atmospheric propagation and ionospheric effects diagrammed in Figure 9.3. The propagation of EM waves through the atmosphere follows the basic ray theory and homogeneous media laws; that is, we assume that the wave amplitude is attenuated as a function of range and the delay corresponds to the scatterer return. Once the wave enters the ionosphere, it is subject to bending and a dispersive filtering effect. The survey of [6] on ionospheric effects and characterization indicate that for weak scintillation, the ionosphere acts as a low-pass filter with cutoff between 1 and 2 Hz with a second-order roll-off. We tacitly lump the atmosphere and ionospheric effects, both forward and reverse (scattered), $H_{\text{ion}}(t)$ of our model into the so-called ionospheric filter, where we now replace $s(t) \rightarrow H_{\text{ion}}(t) * s(t)$ to obtain the new model

$$\tilde{x}(t) = H_{\text{ion}}(t) * s(t) + c(t) + v(t) \tag{9.11}$$

This completes the development of the precession/nutation signal model. Next we perform a simulation using this representation.

The signal model simulation of Eqs. (9.7) through (9.11) was executed over a section of the trajectory to investigate the effect of the nutations. A phase change in nutation was also synthesized in order to design signal processing to detect and extract this type of information which is important to our mission analysis. From archival data we have observed this phenomenon and have incorporated it in the signal simulator by creating this lumped ionospheric filter and convolving its impulse response with the signal producing the scattered return. The results are shown in Figure 9.5. We see the simulated data in Figure 9.5*a* and the trajectory compensated data (motion compensated or trend removed) in Figure 9.5*b*. We also note the subtle changes in nutational phase with the times annotated in the figure. From the spectrum we note the dominant precession frequency along with the nutational or wobble frequencies appearing as disturbances "riding" on top of the precession sinusoid in the time domain. The actual nutational frequencies are found at the valleys of the spectrum, while the peaks coincide to the side band frequencies depicted previously in Figure 9.4. The ionospheric effect on the spectrum and data is also shown. We note that the nutation frequency peaks are not as distinct as they would be without the ionospheric aberrations [6]. This is caused by the low-pass filtering effect of the ionosphere and is obvious in the resulting spectrum. We used the minimum variance distortionless response (*MVDR*) spectral estimator to produce spectral estimates directly proportional to the power [7]. All of these results lead us to a strategy to process the RCS data by noting the deleterious effects of the particular phenomenon and removing or enhancing the measurements to extract the desired information.

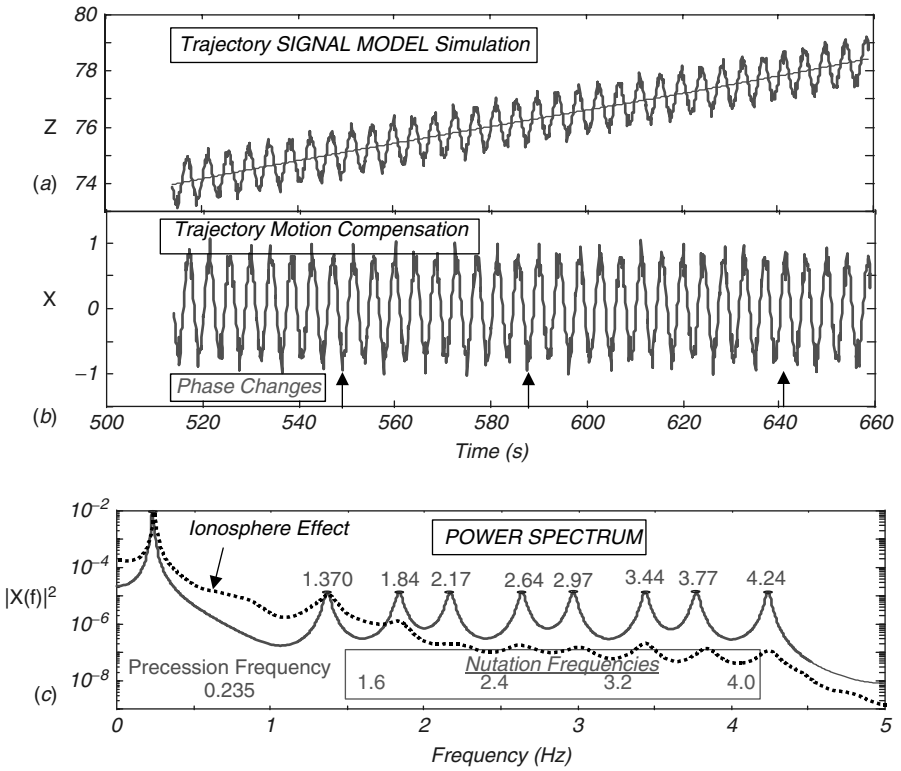



**Figure 9.5.** Trajectory simulation and tracking: (*a*) Simulation. (*b*) Motion compensation. (*c*) Spectral estimate with ionospheric effects (dotted).

### 9.1.3 Processing of RV Signatures

In this section we develop the model-based signal processing techniques used to extract the RV signature information contained in the noisy radar measurement data contaminated with noise and propagation aberrations. The approach is to use the signal processing model and extract all of the information after careful processing to produce an enhanced estimate of the signal. We begin by first defining the problem in terms of our signal models as the RV radar signal processing problem:

> **GIVEN** a set of noisy radar measurements, $\{\tilde{x}(t)\}$ of Eq. (9.11). **FIND** the "best" estimate of the radar signal, $\hat{s}(t)$, and its set of underlying parameters describing the RV motion dynamics: *precession* and *nutation* frequencies, $\hat{f}_p$, $\{\hat{f}_v(k)\}; k = 1, \ldots, N_v$.

We define this problem to specify the individual steps we take to first obtain an enhanced estimate of the RV signature data and then extract the parameters of interest. The signal processing of the raw data is performed in the subsequent steps.

*Trajectory Motion Compensation* The first step we take in signal processing is to perform the trajectory motion compensation (*TMC*) by fitting a polynomial to the raw data. A minimum variance estimator is designed to estimate the required position, velocity and acceleration coefficients using a second-order (or higher) polynomial model. We use the range polynomial model

$$c(t) = R_o + v_R t + a_r \frac{t^2}{2} = [1 \quad t \quad \frac{t^2}{2}] \begin{bmatrix} R_o \\ v_r \\ a_r \end{bmatrix} = \underline{\tau}'(t)\underline{\phi} \tag{9.12}$$

which can be modeled stochastically with additive gaussian noise, $n \sim \mathcal{N}(0, \sigma_n^2)$, as

$$\tilde{x}(t) = \underline{\tau}'(t)\underline{\phi} + \underline{n}(t) \tag{9.13}$$

The estimator follows by letting $t \to t_k;\ k = 0, \ldots, K-1$ and "stacking" the equations of Eq. (9.13) to create the batch measurement model,

$$\underline{\tilde{x}} = T\underline{\phi} + \underline{n} \tag{9.14}$$

with $\underline{\tilde{x}} := [\tilde{x}(t_o) \ \cdots \ \tilde{x}(t_{K-1})]\prime$, $T \in \mathcal{R}^{3\times 1}$, $\underline{\phi} \in \mathcal{R}^{3\times 1}$. The minimum variance solution to this problem is well known [7] and can be solved robustly using the *SVD* algorithm to yield

$$\underline{\hat{\phi}}_{\text{mv}} = \left(T'W^{-1}T\right)T'W^{-1}\underline{\tilde{x}} \tag{9.15}$$

where $W = \text{cov}(\underline{n})$. The minimum variance estimate of the corresponding trajectory is given by

$$\hat{c}_{\text{mv}}(t_n) = \underline{\tau}'(t_k)\underline{\phi}_{\text{mv}} = \hat{R}_o + \hat{v}_R t_k + \hat{a}_R \frac{t_k^2}{2} \tag{9.16}$$

Trajectory motion compensation is then performed by correcting the measurement data to obtain

$$x(t) = \tilde{x}(t) - \hat{c}_{\text{mv}}(t) \approx H_{\text{ion}}(t) * s(t) + v(t) \tag{9.17}$$

This completes the trajectory motion compensation algorithm. Next we consider ionospheric compensation.

*Ionospheric Compensation/Removal* Here the signal processing approach is developed to isolate and estimate the ionospheric filter from the *TMC* data and remove its effect by inverse filtering using a finite impulse response (FIR) Wiener filter [7]. The compensated data are used to design the required filters. Observing

the spectrum of Figure 9.5 and using the prior knowledge of the ionospheric filter function [6], we first construct a "notch" filter (0.15 to 0.30 Hz) to remove the precession frequency. Next a low-order (< 10) FIR filter is designed ($N = 7$) using the Durbin FIR algorithm [7]. This filter is essentially an estimate of the ionospheric impulse response function, $H_{\text{ion}}(t)$ or equivalently its transfer function $H_{\text{ion}}(f)$. The low order is chosen specifically to ensure that the filter does not estimate any of the nutation frequencies, since the inverse filter would remove them inadvertently from the data. An optimal inverse filter, $\hat{H}_{\text{ion}}^{-1}(t)$, is then estimated using a minimum mean-squared error design (optimal Wiener solution) by

$$\hat{H}_{\text{ion}}^{-1}(t) = \mathbf{R}_{HH}^{-1}\underline{r}_{\delta H} \tag{9.18}$$

where $\mathbf{R}_{HH}^{-1}$ is an $K \times K$ Toeplitz matrix and $\underline{r}_{\delta H}$ is an $K \times 1$ cross-correlation vector with $K$ the FIR filter order. This inversion can be performed efficiently using the Levinson-Wiggins-Robinson (*LWR*) algorithm of Chapter 4. Once the inverse filter is designed ($N = 16$), it can be applied to the motion compensated data to remove the ionospheric effect and enhance the precession and nutation frequencies,

$$\hat{x}(t) = \hat{H}_{\text{ion}}^{-1}(t) * x(t) = \hat{H}_{\text{ion}}^{-1}(t) * [H_{\text{ion}}(t) * s(t) + v(t)] \approx \hat{s}(t) + \tilde{v}(t) \tag{9.19}$$

where $\tilde{v}(t) = \hat{H}_{\text{ion}}^{-1}(t) * v(t)$. Thus the enhanced measurement, $\hat{x}(t)$, is now directly related to the signal which contains the desired precession and nutation frequencies as well as noise. Next we decompose the precession from the nutation frequencies for further analysis and eventual detection.

*Precession/Nutation Signal Decomposition* We develop an algorithm to extract the precession and nutation frequency information from the enhanced data above. We perform this decomposition, first, to separate and enhance the precession/nutation signals and, second (based on the signal model), to perform processing of the nutational frequencies required for the demodulation operation to extract the actual nutation frequencies.

Correlation canceling provides an optimal method to decompose the signal into its constituent parts for enhancement and further processing. From Eq. (9.19) we see that after enhancement, we are left with a processed signal containing both precession and nutation components as well as noise; that is, the signal consists of these two parts defined by

$$s(t) = s_p(t) + s_v(t) \tag{9.20}$$

where $s_p(t)$ is the precession component and $s_v(t)$ is the nutation component. The basic idea in correlation canceling is to have a reference channel that is correlated to one of the signals and use it for the desired decomposition (see Chapter 7). Assume that the reference channel, $r_p(t)$ is correlated to the precession signal such that

$$\hat{s}_p(t) = \hat{H}_p(t) * r_p(t) \tag{9.21}$$

then the decomposition is obtained by

$$\hat{s}_v(t) = s(t) - \hat{s}_p(t) = s_p(t) + s_v(t) - \hat{s}_p(t) \approx s_v(t) \tag{9.22}$$

The optimal (minimum error variance) solution to this problem is given again by the Wiener filter [7] as

$$\hat{H}_p(t) = R_{r_p r_p}^{-1} \underline{r}_{s r_p} \tag{9.23}$$

with $R_{r_p r_p}^{-1}$ a $N \times N$ Toeplitz matrix and $\underline{r}_{s r_p}$ an $N \times 1$ cross-correlation vector for a $N$th-order filter.

For our problem we do not have a *reference* channel, but we can observe the dominant precession spectrum, we create a reference by low-pass filtering (0.75 Hz cutoff) the enhanced data, $\hat{x}(t)$. The results of applying the canceler to our synthesized data for the ionosphere removed data are shown in Figure 9.6. In Figure 9.6*a* and 9.6*b* we see the canceler outputs for the enhanced precession and nutation responses. In Figure 9.6*c* we see the original spectrum and its decomposition into

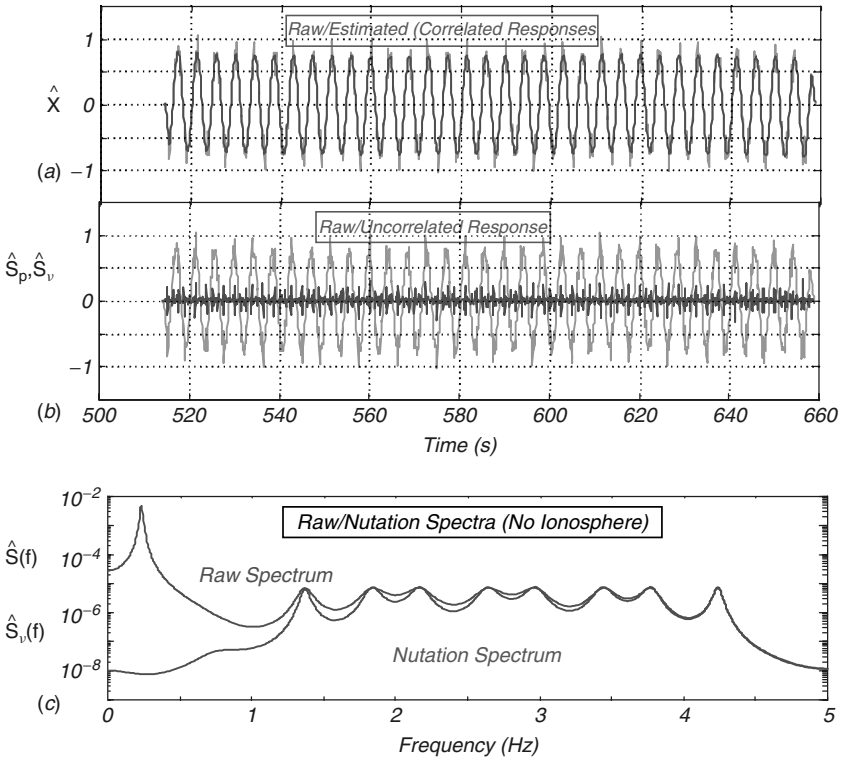



**Figure 9.6.** Precession/nutation correlation canceling (no ionosphere): (*a*) Estimated precession and raw data. (*b*) Estimated nutation and raw data. (*c*) Raw and nutation spectra.

precession and nutation components. It is clear from the spectra that the decomposition is capable of performing the optimal decomposition and extracting both enhanced signals. Next we analyze these signals to extract the desired information.

*Precession Frequency Estimation* The precession frequency can be estimated directly from the power spectrum of the enhanced data, $\hat{x}(t)$, or in cases of low *SNR*, the enhanced precession signal, $\hat{s}_p(t)$, available from one channel of the correlation canceler can be used. We use the high-resolution, maximum entropy method (*MEM*), spectral estimation algorithm coupled to a peak detector to estimate the desired precession frequency [7]. The estimator employs the autoregressive (*AR*) or all-pole model given by

$$S_{AR}(f) = \frac{\sigma_\epsilon^2}{|A(e^{j2\pi f})|^2} \tag{9.24}$$

where $A(e^{j2\pi f}) = 1 + a_1 e^{-j2\pi f} + \cdots + a_{N_a} e^{-j2N_a\pi f}$, and $\sigma_\epsilon^2$ is the prediction error variance. The algorithm recursively solves the following normal equations:

$$\underline{\hat{a}} = R_{\hat{x}\hat{x}}^{-1}\underline{r} \tag{9.25}$$

for $\hat{a}$ the estimated coefficients of the $N_a$-th order polynomial $A(e^{j2\pi f})$ and $R_{\hat{x}\hat{x}}$ is the $N_a \times N_a$ Toeplitz correlation matrix with $\underline{r}$ the $N_a \times 1$ correlation vector. This algorithm can efficiently be implemented recursively using either the Durbin or Burg recursions (see Chapter 4) [7]. Applying the Durbin algorithm to our enhanced data, that is, performing the spectral estimation/peak detection, we find that $\hat{f}_p = 0.234$ Hz. Next we estimate the nutation frequencies.

*Nutation Frequency Estimation* We develop an approach to extract the nutation or wobble frequencies from the estimated nutation signal $\hat{s}_\nu(t)$ obtained from the correlation canceller decomposition output discussed earlier. Recall that the precession frequency acts as a carrier frequency similar to amplitude modulation (AM) and only the side bands are available in the spectrum. Thus, as in an AM receiver, we can demodulate the estimated nutation signal to extract the baseband or nutation frequencies in this case. If we multiply the nutation by the precession (carrier), both available from the correlation canceler outputs, that is,

$$s_\nu(t) \approx \hat{s}_\nu(t) \times \hat{s}_p(t) = \hat{s}_\nu(t) \times \sin 2\pi f_p = s_k(t) \times \sin 2\pi f_p \tag{9.26}$$

Then multiplying by the precession sinusoid and using a set of trigonometric identities leads us to [5],

$$\begin{aligned} s_\nu(t) \;=\; & \frac{1}{2}\sum_{k=1}^{N_\nu}\left[\frac{c_{23}(t)+c_{32}(t)}{2}\right]\cos\left(2\pi f_\nu(k)t\right) + \left[\frac{c_{14}(t)-c_{41}(t)}{2}\right]\sin\left(2\pi f_\nu(k)t\right) \\ & + c_{14}(k)\sin\left(2\pi(2f_p - f_\nu(k))\,t + c_{41}(k)\sin\left(2\pi(2f_p + f_\nu(k))\,t\right.\right. \\ & - c_{23}(k)\cos\left(2\pi(2f_p - f_\nu(k))\,t + c_{32}(k)\cos\left(2\pi(2f_p + f_\nu(k))\,t\right.\right. \end{aligned} \tag{9.27}$$

which shows that upon taking the Fourier transform, we can expect to recover the desired set of nutational frequencies, $\{f_\nu(k)\};\ k = 1, \ldots, N_\nu$ as resonant peaks similar to those of Eq. (9.10) from the estimated spectrum along with multiple side bands at $\{2f_{p\nu}(k)\};\ k = 1, \ldots, N_\nu$. Thus the corresponding demodulation spectrum is ideally a set of impulses at the appropriate nutation frequencies extracted as in the precession estimation with a high-resolution spectral estimator coupled to a peak detector. In order to implement the demodulation, we simply use the precession channel output (Eq. 9.21) of the correlation canceler as our carrier as in Eq. (9.27) above.

For our simulation problem we applied the demodulator to the case with ionosphere synthesized and removed as shown in Figure 9.7. Here we see that the deleterious effect of the ionosphere even in this simple simulation is to decrease the *SNR* of the nutational peaks and emphasizes even more the need for processing. The estimated nutational spectral peaks $\hat{f}_\nu(k)$ in this case are (1.63, 2.42, 3.19, 3.98 Hz) compared to the actual peaks at (1.6, 2.4, 3.2, 4 Hz). Note also that low-pass filtering effect of the ionosphere drives the nutational peaks further into the noise as seen in Figure 9.7*b*. Here we note the nutation frequencies and their estimation using the peak detector. The estimated nutational spectral peaks with ionospheric

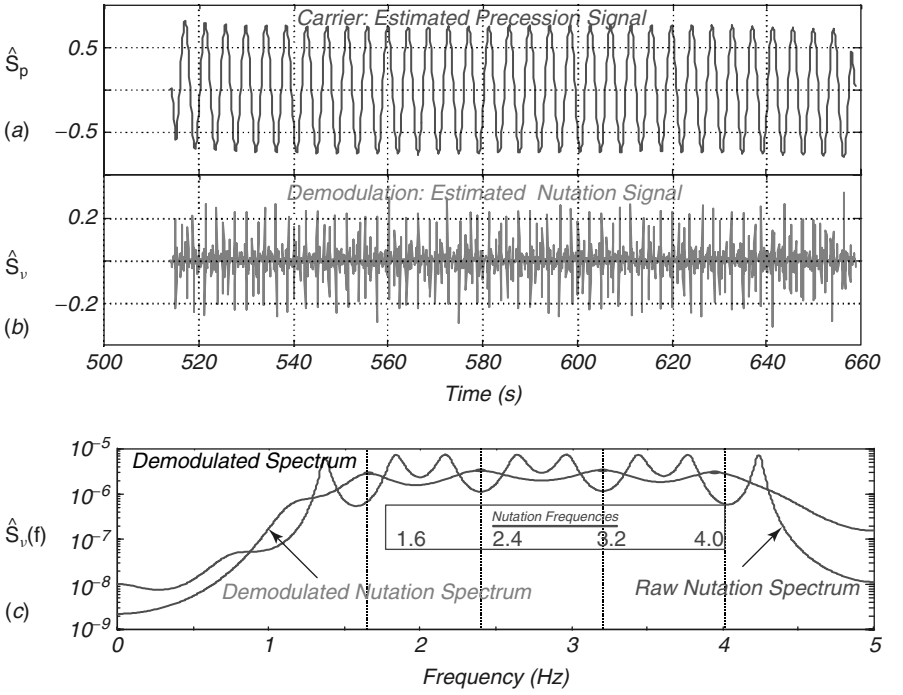



**Figure 9.7.** Demodulation processing for nutational frequency estimation (no ionosphere): (*a*) Carrier estimated data. (*b*) Demodulated nutation estimate. (*c*) Canceled and demodulated nutational spectra.

effects are: (1.61, 3.19, 4.2 Hz) with the peak at 2.4 Hz undetected. The next step in the processing is to develop a method to analyze the time evolution of the flight data and observe its instantaneous spectrum for events of interest.

*Spectrogram Estimation* We develop the recursive-in-time approach to instantaneous spectral estimation (see Section 7.6.4) enabling us to produce the desired spectrogram (amplitude vs. time vs. frequency) for event detection. Suppose that we parametrically model the enhanced measurement data by a time-frequency representation specified by an instantaneous autoregressive moving average (*ARMA*) model. This model has the general difference equation form, given by

$$A(q^{-1},t)y(t) = C(q^{-1},t)\varepsilon(t) \tag{9.28}$$

for the enhanced measurement, $y(t)$, contaminated with zero-mean, white gaussian noise, $\varepsilon \sim \mathcal{N}(0,\sigma_\varepsilon^2)$, with the corresponding instantaneous polynomials at the instant $t$ defined by

$$\begin{aligned} A(q^{-1},t) &= 1 + a_1(t)q^{-1} + \cdots + a_{N_a}(t)q^{-N_a} \\ C(q^{-1},t) &= c_o + c_1(t)q^{-1} + \cdots + c_{N_c}(t)q^{-N_c} \end{aligned} \tag{9.29}$$

Here the backward shift or delay operator is defined by, $q^{-i}y(t) := y(t-i)$, and therefore we can write Eq. (9.28) simply as

$$y(t) = -\sum_{k=1}^{N_a} a_k(t)y(t-k) + \sum_{k=0}^{N_c} c_k(t)\varepsilon(t-k) \tag{9.30}$$

If we take the *DFT* of the difference equation, then we obtain the *instantaneous transfer function* (ignoring stochastic aspect)

$$H(e^{j2\pi f},t) = \frac{Y(e^{j2\pi f},t)}{\mathrm{E}(e^{j2\pi f},t)} = \frac{C(e^{j2\pi f},t)}{A(e^{j2\pi f},t)} \tag{9.31}$$

or more appropriately the corresponding *instantaneous power spectrum* defined by

$$S(f,t) := \left|H(e^{j2\pi f},t)\right|^2 = \left|\frac{C(e^{j2\pi f},t)}{A(e^{j2\pi f},t)}\right|^2 \tag{9.32}$$

So we see that if we use the parametric *ARMA*($N_a$,$N_c$) representation of the enhanced measurement signal and transform it to the spectral domain, then we can obtain the instantaneous spectral estimate. There are a wealth of adaptive *ARMA* algorithms available in the literature ([7], [9]), but since we are primarily interested in estimating the spectrum at each time instant, we confine our choices to those that are recursive-in-time, enabling us to achieve our goal without the loss of temporal

resolution evolving from window-based methods such as the short-time Fourier transform.

We ran the instantaneous spectrogram estimator using the *RPEM* algorithm over the ionosphere-free enhanced measurements and the results are shown in Figure 7.15 of Section *7.6.4*. Here we see that the spectrogram image clearly displays the dominant precession spectral peak as it temporally evolves as well as the nutational side band frequencies. Note with this high *SNR* the synthesized "phase changes" are also clearly observable as indicated by the arrows. The enhanced measurement is time aligned and shown below the spectrogram as well. Next the estimator was run over the ionosphere removed measurement data, and we, again, observed that the precession and nutation side band spectral peaks. The phase change events were no longer as visible—again, indicating the detrimental effect of the ionosphere and requiring more sophisticated event detection methods that we discuss next.

*Event Detection* We develop a set of event detectors based on the instantaneous *ARMA* signal models and recursive algorithms discussed above. The underlying idea motivating this type of detector is based on the statistical properties of the prediction error or innovations sequence which is (simply) the difference between the measurement and its prediction using the *ARMA* model parameters. When the model and its parameters "track" the data (after initialization), then any abrupt change in parameter estimates or data will be reflected directly in the prediction error sequence. The subsequent detectors test the sequence for statistically significant changes indicating an event or possible phase change. The main point to realize is that since we are using a recursive-in-time estimator, the prediction error represents "how well the *ARMA* model fits the data at each time instant." If it is an optimal fit, then the prediction error sequence should be zero-mean and statistically white (uncorrelated) and the detectors evolve naturally from this property.

When using these statistical tests care must be taken. If the models are nonlinear or nonstationary, then the usual whiteness/zero-mean tests, that is, testing that 95% of the sample (normalized) prediction error correlations lie within the bounds given by

$$\left[\hat{c}_{\epsilon\epsilon}(k) \pm \frac{1.96}{\sqrt{N}}\right], \quad \hat{c}_{\epsilon\epsilon}(k) := \frac{\sigma_\epsilon^2(k)}{\sigma_\epsilon^2(0)} \tag{9.33}$$

The corresponding zero-mean test defined by

$$\left[\hat{m}_\epsilon(k) < 1.96\sqrt{\frac{\sigma_\epsilon^2(k)}{N}}\right] \tag{9.34}$$

Both definitions rely on quasi-stationary assumptions and sample statistics to estimate the required correlations. However, it can be argued heuristically that when the ARMA estimator is tuned, the nonstationarities are being tracked by the processor

even in the nonlinear case and therefore, the prediction errors should be covariance stationary.

When data are nonstationary, then a more reliable statistic to use is the weighted sum-squared residual (*WSSR*) which is a measure of the overall global estimation performance of the processor again determining the "whiteness" of the prediction error sequence (see Chapter 5). It essentially aggregates all the information available in the prediction error and tests whiteness by requiring that the decision function, $\rho(\ell)$, lies below the specified threshold to be deemed statistically white. If the *WSSR* statistic does lie beneath the calculated threshold, then theoretically the estimator is tuned and said to converge or in our problem there is no abrupt phase change. The *WSSR* statistic is estimated over some finite window of $N$-samples; that is, it is defined by

$$\rho(\ell) := \sum_{k=\ell-N+1}^{\ell} \frac{\epsilon^2(t)}{\sigma_\epsilon^2(k)}, \qquad \ell \geq N \tag{9.35}$$

The detection algorithm is

$$\rho(\ell) \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \tau \tag{9.36}$$

where $H_0$ is the hypothesis that the model "fits" the data (white prediction errors), while $H_1$ is the hypothesis that there is a mismatch or change specified by non-zero-mean, and/or nonwhite prediction errors. Under the zero-mean assumption, the *WSSR* statistic is equivalent to testing that the prediction error sequence is white. Under $H_0$, *WSSR* is distributed $\chi^2(N)$. It is possible to show [8] for a large $N > 30$ and a level of significance of $\alpha = 0.05$, the threshold is given by

$$\tau = N + 1.96\sqrt{N} \tag{9.37}$$

Here the $N$-sample window is designed to slide through the prediction error data and estimate its whiteness. Even in the worst case where these estimators may not prove to be completely consistent, the processor (when tuned) predicts the nonstationary prediction error covariance, $\sigma_\epsilon^2(k)$ enabling a simple (varying with $\ell$) confidence interval to be constructed and used for testing similar to Eq. (9.35) above. Thus overall performance of the processor and therefore the *event detector* can be assessed by analyzing the statistical properties of the prediction errors.

The final detector is ad hoc, but based on the premise above that abrupt changes in the radar data will be reflected by abrupt changes in the parameters of the *ARMA* model. The parametric change detector is obtained by averaging over all of the recursive parameter estimates and searching for significant peaks to detect

events. It is defined by

$$\underline{m}_a(t_k) = \frac{1}{N_a + N_c} \sum_{i=1}^{N_a+N_c} \underline{\alpha}_i(t_k)$$

$$\underline{m}_a(t_k) \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \tau \tag{9.38}$$

We executed these detectors over synthesized data sets in the ionosphere-free case with the results shown in Figure 9.8. We observe that in this case all three of the detectors perform well in detecting the changes in nutational phase. In Figure 9.8*a*, the *WSSR* or phase change detector shows three peaks above the threshold at the correct event times (550, 580, 630 seconds) as do the parametric change (shown in Figure 9.8*b* and 9.8*c*) and whiteness 9.8*d* detectors. The results

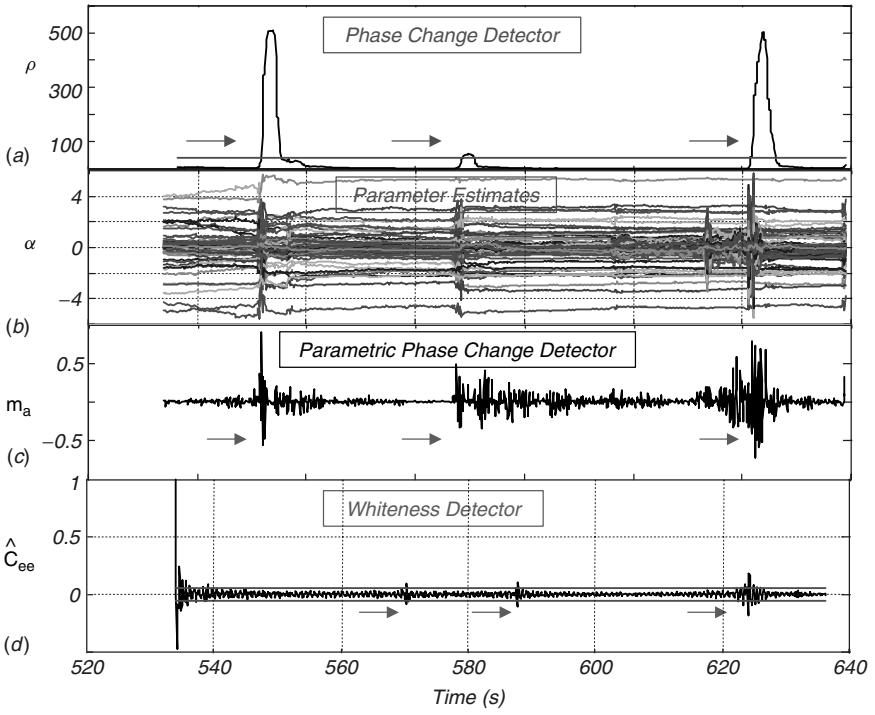



**Figure 9.8.** Event detection of enhanced measurement. (*a*) Phase change detection ($N = 25$, threshold = 38.8). (*b*) Parameter estimates, *ARMA*(32, 25). (*c*) Parametric change detector. (*d*) Whiteness event detector (4.4% out).

are not as apparent when the ionosphere is included and removed to produce the enhanced measurement. This completes our discussion of the signal processing approach taken to analyze the real (not complex) radar measurements, extract the required RV dynamics and detect abrupt events.

We summarize the model-based processing utilized to extract the RV dynamics as follows:

*Step 1.* Perform trajectory motion compensation on the measured radar data.

*Step 2.* Process the compensated data to isolate the ionospheric response, determine its impulse response and estimate the inverse filter used to remove its effects and produce the enhanced measurement.

*Step 3.* Decompose the enhanced measurement into the precession and nutation component signals using the correlation canceler.

*Step 4.* Estimate the precession frequency using the high-resolution *MEM* power spectral estimator and peak detector.

*Step 5.* Demodulate the nutation signal obtained from step 3, and perform a high-resolution spectral estimate/peak detection to extract the desired nutation frequencies.

*Step 6.* Estimate the instantaneous spectrogram of the enhanced measurement performing the recursive-in-time, time-frequency estimation.

*Step 7.* Detect any changes (in-time) by observing the spectrogram and perform the phase change, parameter change, or whiteness detectors to locate and analyze the events.

Next we apply these techniques to actual flight data and analyze their performance.

### 9.1.4 Flight Data Processing

Here we discuss the application of the proposed signal processing techniques developed above to a small "snapshot" of flight data demonstrating their performance. As before, the raw data was pre-processed with trajectory motion compensation and corresponding spectral analysis. The results are shown in Figure 9.9 with the estimated trajectory and compensated flight data. The precession and nutation frequencies are shown in the power spectrum above with the "low-pass" filtering effect of the ionosphere. With this preprocessing completed the environmental processing (ionospheric transfer function estimation and inversion) is performed next. The correlation canceling processor is designed to decompose the signal into the precession and nutation estimates. Once completed, the next step is to demodulate the nutational side bands to estimate the desired frequencies. The demodulation results are shown in Figure 9.10 below. Here we observe the precession carrier and demodulated nutation signals along with the resulting spectrum.

During the analysis of each of these processed signals, we estimated the spectrogram of the processed flight data including both the precession and nutational

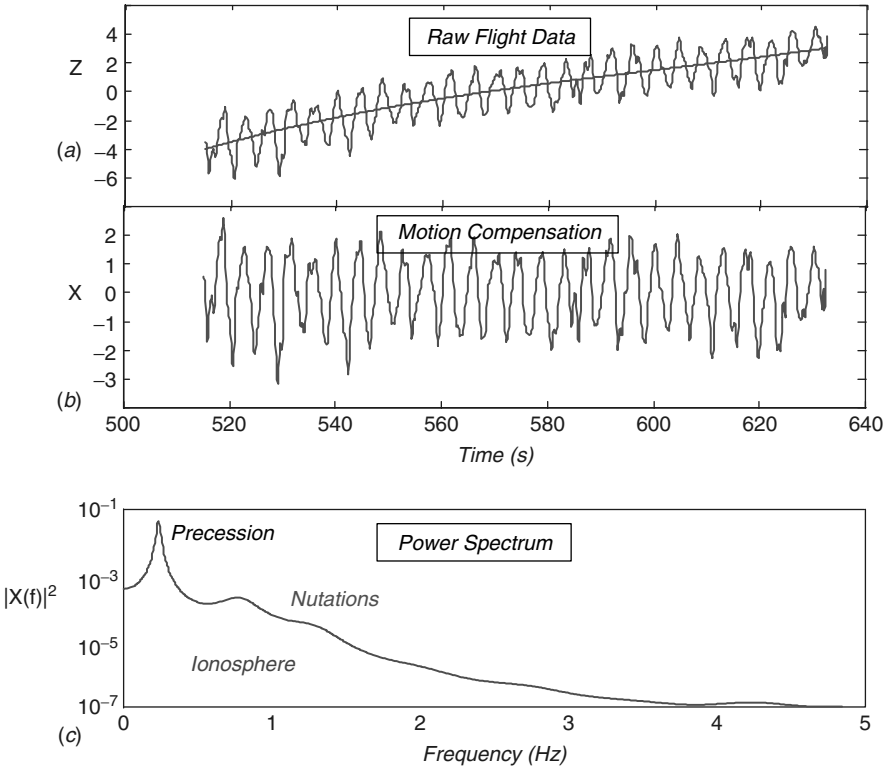



**Figure 9.9.** Raw flight data: (*a*) Trajectory estimation. (*b*) Motion compensation. (*c*) Power spectrum.

components. The results indicated the presence of both components in the spectrogram (time-frequency) domain. The precession appears reasonably constant during this portion of the flight, but the nutational frequencies (side bands) are not obvious during various portions of the data set. The phase or change detection algorithms were also applied to this data set. Here the possibilities of some events occurring were noted but cannot be confirmed from the spectrogram. In fact the Whiteness Detector does not indicate any events at all (statistically white sequence) probably owing to the fact that the precession dominates the response resulting in an excellent precessional frequency fit. Next we investigate just the nutational portion (higher frequencies) of the spectrum to observe if any events can be detected.

Performing the correlation cancelling discussed above, we investigate the nutational spectrogram and detection analysis. The results of applying the phase change detectors to this data are shown in Figure 9.11 where we see that the Whiteness Detector is no longer white indicating an event or possible change. The Phase Change detector also indicates the possibility of an event that are more clearly distinguished from the data that was not decomposed. Therefore we demodulate these data to estimate the nutations and perform change detection. The nutational

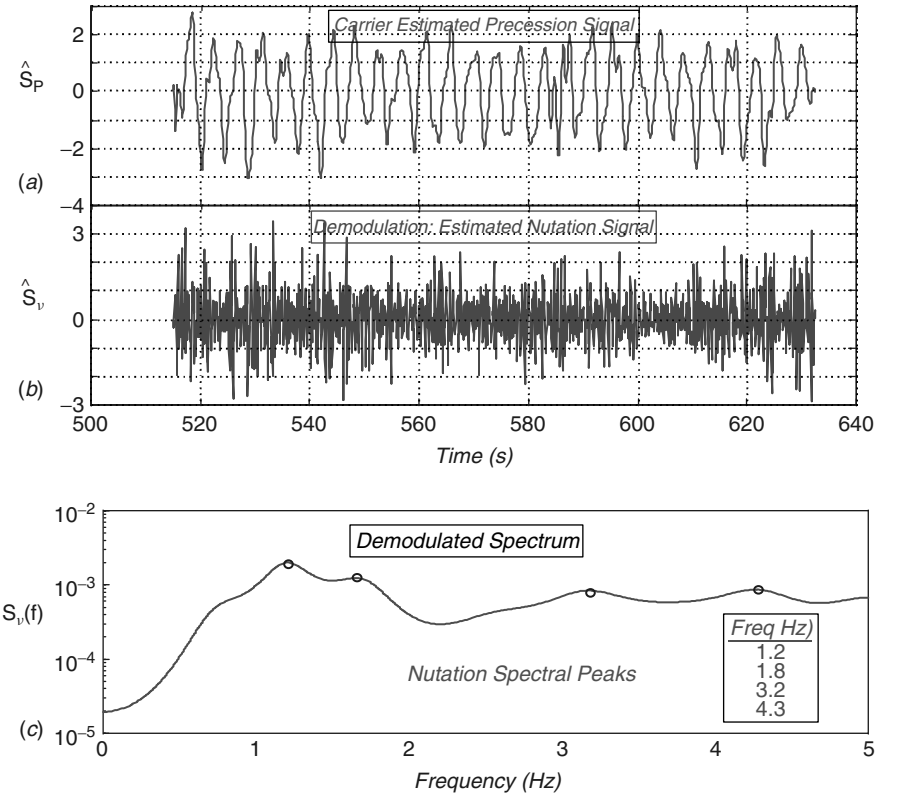



**Figure 9.10.** Demodulation processing: (*a*) Carrier estimate. (*b*) Nutation estimate. (*c*) Spectrum.

(side band) was demodulated and its corresponding spectrogram estimated. The spectrogram is shown in Figure 9.12 where the precession has been removed and the nutational frequencies enhanced even further. Here we can clearly observe the nutational frequency evolution and the potential events that is not obvious in the raw RCS data. Next we apply the event detection algorithms with the results shown in Figure 9.13. In Figure 9.13*a* we clearly see the possible events as indicated previously, but in this case the Parametric Detector appears to have also confirmed transients at the same approximate times as the Phase Change Detector. The Whiteness Detector is marginally nonwhite, indicating its lack of sensitivity in these low *SNR* regimes. This completes the analysis of a "snip-it" of flight data, demonstrating the performance of the processing developed to analyze the data.

We summarize the results of this application as follows:

Criterion: $J = \text{trace } E\{e^2(t)\}$

Models:

Signal: $s(t) = A(q^{-1})y(t)$

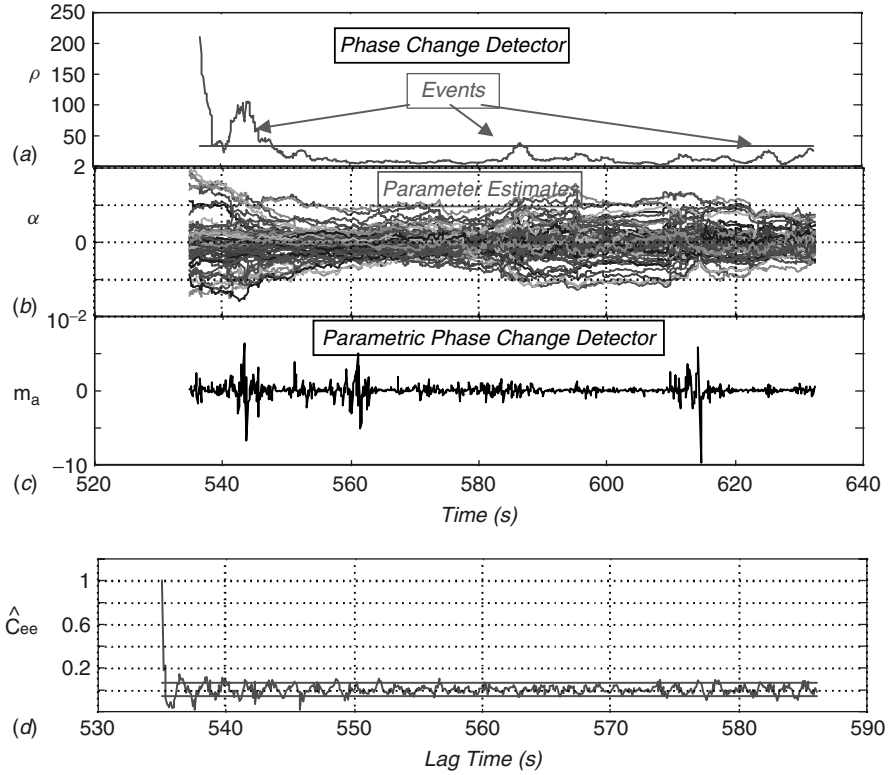



**Figure 9.11.** Process/nutation flight data detection analysis: (*a*) Phase change detector. (*b*) Parameter estimates. (*c*) Parametric detector. (*d*) Whiteness detector.

| | |
|---|---|
| Measurements: | $y(t) = s(t) + v(t)$ |
| Noise: | $e \sim \mathcal{N}(0, R_{ee})$ |
| Algorithm: | $\hat{s}(t\|t) = \hat{y}(t\|t) = \hat{A}(q^{-1})\hat{y}(t\|t-1)$ |
| Quality: | $R_{ee}$ |

This completes the application on reentry vehicle.

### 9.1.5 Summary

In this section we developed a suite of techniques to process radar signature data gathered from RV flights. The parameters of most interest are the precession frequency and the corresponding micro dynamics (nutations) as well as any changes in phase during flight. Based on the dynamics of the RV, we postulated a signal model to develop and test the processing algorithms. After developing this model, we constructed a simulation to motivate the underlying processing. We

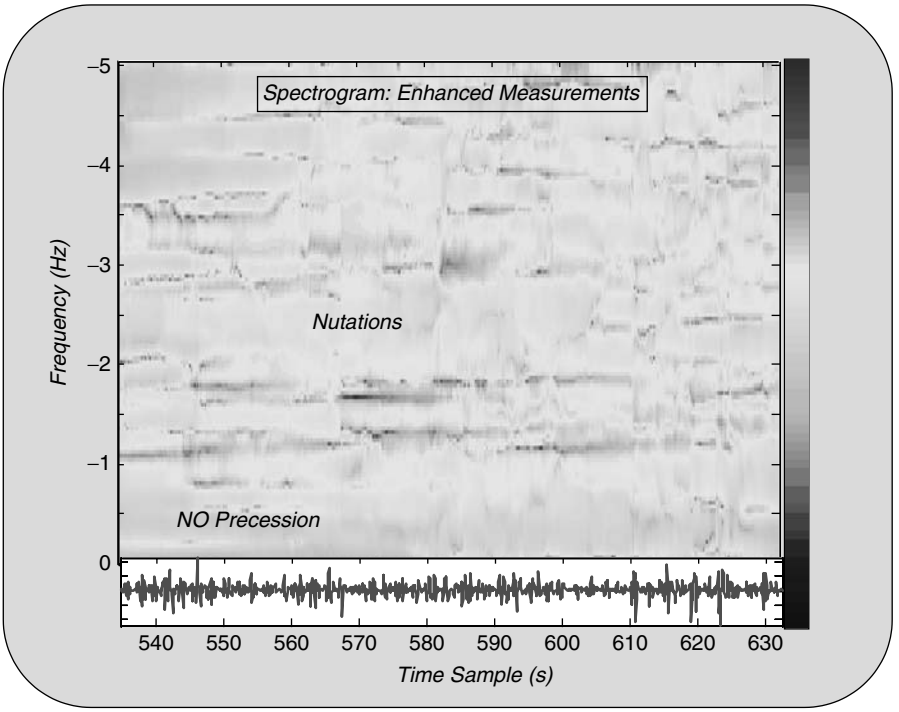



**Figure 9.12.** Demodulated flight data spectrogram.

showed that the basic preprocessing step consisted of trajectory motion compensation implemented by a polynomial range model (in time) with trend estimation and removal required. Next, and perhaps the most critical step, is that of the removal of both (lumped together) atmospheric and ionospheric effects. Here both a forward ionospheric transfer function and its inverse were estimated using optimal minimum variance designs (Wiener filter). The resulting signal provided processed data for spectrogram (time-frequency) estimation and analysis. The spectrogram enables the observation of the dynamic evolution of frequencies: precession, nutations and phase changes. It was also shown that further analysis and signal enhancement was required to extract the desired parameters.

A set of techniques was developed to enhance and separate the precession from the nutation signatures. Here an optimal, minimum variance, correlation canceling scheme evolved to solve this problem. Once accomplished, spectrograms for each of the parameter sets became available. Next a suite of detection techniques was developed based on the instantaneous (recursive-in-time) parametric estimator employed for spectrogram estimation. Each detector was developed from the estimator output. The prediction errors were used to detect events or changes in the phase of the underlying instantaneous autoregressive moving-average (*ARMA*) model coefficients. A Phase Change Detector based on the weighted sum-squared

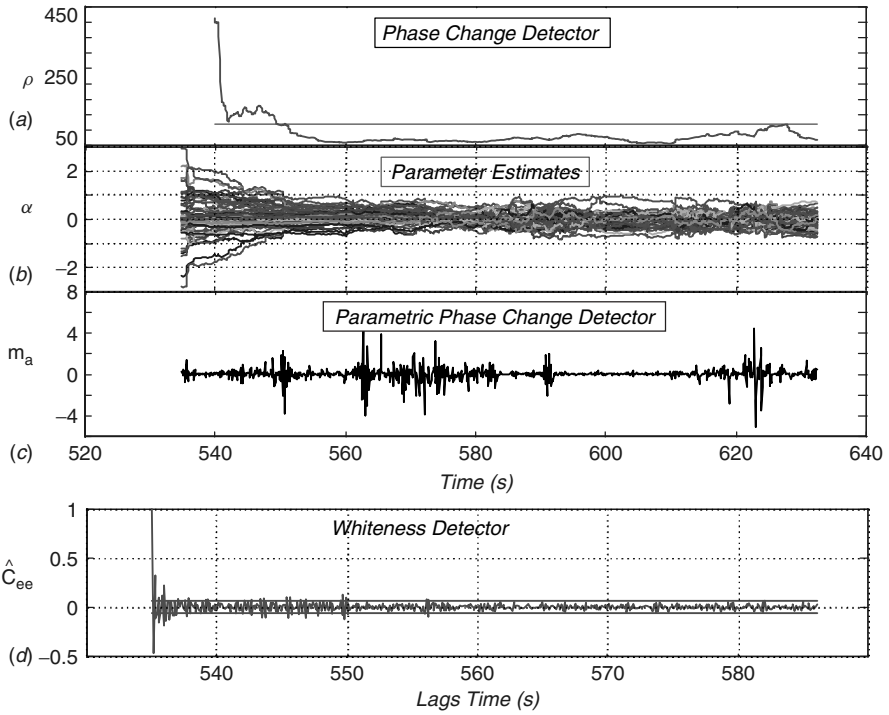



**Figure 9.13.** Processed/demodulated flight data detection analysis: (*a*) Phase change detector. (*b*) Parameter estimates. (*c*) Parametric phase change detector. (*d*) Whiteness detector.

residuals (*WSSR*) statistic was developed and appears to give the most robust performance on both simulated and measured data sets. An ad hoc Parametric Detector was also developed to detect changes in the *ARMA* coefficients along with the standard Whiteness Detector statistically testing the fact that the prediction error must be uncorrelated for a normal (no change) condition.

All the above-mentioned techniques were applied to both synthesized and measured radar signature or cross-section (RCS) data. In the cases shown, an analysis was performed to assess the information extracted by the processing techniques and how it could be used to answer questions about the underlying RV dynamics. The techniques performed reasonably in both cases in estimating the parameters accurately and detecting the events of highest interest (see [5] for details).

## 9.2 *MBP* FOR LASER ULTRASONIC INSPECTIONS

Laser ultrasonics is a remote nondestructive evaluation technique used to inspect critical parts (e.g., aircraft wings) for cracks and flaws [10]. It is based on focusing a laser on the part of interest, heating the material that thermally expands

generating an ultrasonic signal propagating through the material and processing for flaw detection. The use of laser-based ultrasonics in the testing of materials and structures offers various advantages over more traditional ultrasonic methods, but is often less sensitive when applied to real materials. Although high-energy laser pulses can generate large ultrasonic displacements, nondestructive evaluation requires that the ablation regime be avoided; thus, limiting the amount of optical energy which may be used and ultimately decreasing the output *SNR*. For this reason signal processing of laser generated ultrasonic waveforms detected using laser interferometers is required to extract the desired information from a nondestructive laser ultrasonic test. A model-based signal processing technique offers a new way to enhance the signal-to-noise ratios (*SNR*) significantly for these ultrasonic waveforms obtained using laser-based systems with the generation of the ultrasound occurring in the nondestructive thermoelastic regime.

Under ideal conditions good *SNR* can be achieved using laser-based ultrasonics. However, many materials that need to be tested have less than ideal surface finishes for optical detectors. The application of signal processing to laser-based ultrasonics provides the necessary improvement in sensitivity. Aussel and Monchalin used cross-correlation methods to extract acoustic velocities and elastic constants from noisy measurements [11]. Once the constants are made available through experimentation or calculation, it is possible to enhance the noisy interferometer measurements even further by generating a predicted or reference response using a propagation model that captures the essence of the displacement signal to be estimated. Using estimates of the required constants, a reasonable reference response can be generated that enables significant enhancement of the measured displacement. In this section, we discuss a model-reference approach used to increase the *SNR* in noisy laser-based ultrasonic waveforms.

### 9.2.1 Laser Ultrasonic Propagation Modeling

A model for calculating the surface displacement at any point on the surface of a plate resulting from an incident pulse of laser energy has been developed by Spicer [12]. The model calculates the surface displacements resulting from a thermoelastic source. The equations (in cylindrical coordinates) governing the laser ultrasonic waves in the plate are given in terms of scalar and vector potentials defined as $\phi(r, z, t)$ and $\psi(r, z, t)$, respectively. The vector surface displacement, $\mathbf{d}(r, z, t)$ is then constructed in terms of these potentials as

$$\mathbf{d}(r, z, t) = \nabla\phi(r, z, t) + \nabla \times \psi(r, z, t) \tag{9.39}$$

Calculations are performed in the frequency domain, and the final surface displacement is found by inverting the Hankel-Laplace representation in complex variables $p$ and $s$:

$$d(p, z, s) = \Phi(p, z, s) + p^2\psi(p, z, s) \tag{9.40}$$

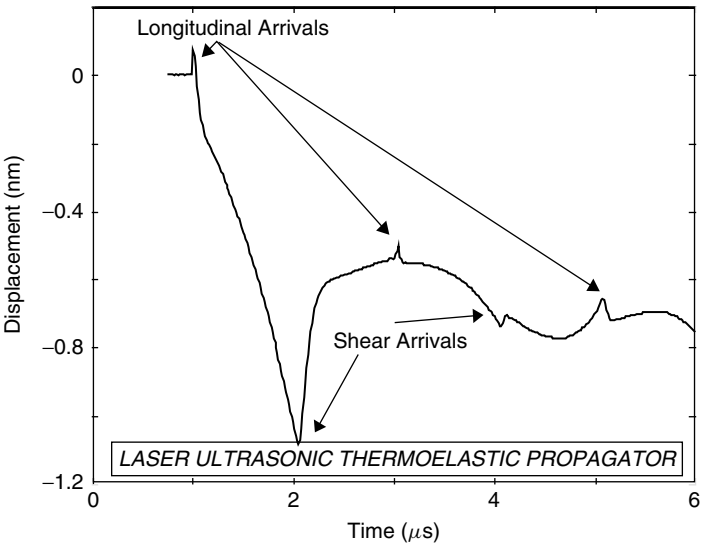



**Figure 9.14.** Synthesized surface displacement: 6 mm aluminum plate.

that is, the surface displacement is

$$d(r, z, t) = \mathbf{H}_o^{-1}\left[\mathbf{L}^{-1}\left[d(p, z, s)\right]\right] \tag{9.41}$$

The simulation algorithm solves these equations to produce the response for a given material, thickness, and so forth. Figure 9.14 shows the calculated waveform *on epicenter* on the side of a 6 mm-thick aluminum plate opposite that on which generation took place using the constants associated with isotropic material behavior. The following constants were used in the calculation: longitudinal wave speed, 6.32 mm/μs; shear wave speed, 3.11 mm/μs; thermal conductivity, 0.235 W/mm K; thermal diffusivity, $9.67 \times 10^{-5}$ mm$^2$/μs; and linear thermal expansion coefficient, $2.4 \times 10^{-5}$ °C$^{-1}$; along with the laser parameters obtained from manufacturer specifications. Figure 9.14 shows the typical surface displacement response, which includes the initial longitudinal and shear arrivals as well as longitudinal and shear arrivals corresponding to multiple traversals through the plate.

### 9.2.2 Model-Based Laser Ultrasonic Processing

For the experimentally obtained waveforms found in this work, a Michelson interferometer was used to measure the ultrasonic surface displacement. For a perfectly adjusted instrument and "small" surface displacements, the measured intensity is directly proportional to the displacement, that is,

$$I(z, t) \propto \alpha d(t) \tag{9.42}$$

where $\alpha$ is a constant that is a function of wave number, photodetector efficiency, carrier charge, frequency of the light, and required scale factors.

There are many sources of noise in interferometric systems, including detector quantum noise and dark current, generation-recombination noise, thermal noise from the electronics, and laser noise. All these noise sources are grouped together and considered simply a measurement noise term characterizing the measurement system model as

$$x(t_k) = I(z_\ell, t_k) + n(t_k) = \alpha d(t_k) + n(t_k) \tag{9.43}$$

with $x(t_k)$ the measured surface displacement at time sample $t_k$, $d(t_k)$ the actual or true displacement, and $n(t_k)$ the lumped random noise contaminating the process.

Ideally one would like to construct a processor that linearly combines the raw set of sampled (digitized) measurements $\{x(t_k)\}$, $k = 0, \ldots, N$ to produce an optimal estimate of the desired displacement, that is,

$$\hat{d}(t_k) = \sum_{\ell=0}^{L-1} w_\ell x(t_k - t_\ell) = \mathbf{w}^T \mathbf{X}(t_k) \tag{9.44}$$

where $\mathbf{w}^T = [w_o \cdots w_{L-1}]$ and $\mathbf{X}^T = [x(t_k) \cdots x(t_k - t_{L-1})]$.

One approach to solve this problem is to formulate it as a minimum variance estimation problem:

> **GIVEN** a set of noisy interferometric measurements, $\{x(t_k)\}$, $k = 0, \ldots, N$. **FIND** the best (minimum error variance) estimate of the surface displacement as $\hat{d}(t_k)$.

Mathematically the set of weights that minimizes the mean-squared error criterion

$$\min_{\mathbf{w}} \; J = E\{\epsilon^2(t_k)\} \qquad \text{for} \quad \epsilon(t_k) := d(t_k) - \hat{d}(t_k) \tag{9.45}$$

must be found. The solution to this problem is classical (see Chapter 4) and is easily found by differentiating $J$ with respect to the weights, setting the result equal to zero and solving for the set of optimal weights, which gives the *normal equations*

$$\mathbf{w}_{\text{opt}} = \mathbf{R}_{xx}^{-1} \mathbf{R}_{dx} \tag{9.46}$$

where $\mathbf{R}_{xx}$ is an $L \times L$ Toeplitz correlation matrix of the data and $\mathbf{R}_{dx}$ is the $L$-dimensional cross-correlation vector between the measurements and desired signal (model reference). The corresponding minimum mean-squared error is easily determined as

$$J_{\min} = \sigma_d^2 - \mathbf{R}_{dx}^T \mathbf{w}_{\text{opt}} \tag{9.47}$$

with $\sigma_d^2$ the variance of the desired or reference response. An efficient approach to solve for the optimal weights is obtained using the Levinson-Wiggins-Robinson (*LWR*) recursion (see Chapter 4) which recursively solves a slight variant of the above relation given by

$$\mathbf{w}_{\text{opt}}(L) = \left(\mathbf{R}_{xx}(L) + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{R}_{dx}(L) \tag{9.48}$$

where the two available "tuning" parameters are $L$, the filter order, and $\sigma_n^2$ the regularization parameter which can shown to be proportional to the measurement noise variance. The performance of the algorithm can be determined by tuning with a particular set of parameters, $[L,\ \sigma_n^2]$ and calculating the resulting mean-squared error, $J_{\min}$. The parameter set yielding the smallest mean-squared error is the best choice.

Another alternative approach to solving the normal equations is by developing an equivalent adaptive solution using the least mean-squared (*LMS*) algorithm (see Chapter 7). The advantage of this approach is computational in that it is proportional to $L$-operations to perform the required inverse ($\mathbf{R}_{xx}^{-1}$) instead of the $L^2$-operations required by the *LWR* algorithm. Also in this approach the instantaneous gradient replaces the stochastic gradient, allowing for the inclusion of nonstationary processes, since it is time recursive. The *LMS* recursion for updating the weights is given by

$$\mathbf{w}(t_{k+1}) = \mathbf{w}(t_k) + \triangle\epsilon(t_k)\mathbf{X}(t_k) \tag{9.49}$$

where the instantaneous error is given by

$$\epsilon(t_k) = d(t_k) - \hat{d}(t_k) = d(t_k) - \mathbf{w}^T(t_k)\mathbf{X}(t_k) \tag{9.50}$$

and $\triangle$ is the gradient (search) step-size (for convergence) bounded by 0 and $2/\lambda_{\max}$, with $\lambda_{\max}$ the maximum eigenvalue of $\mathbf{R}_{xx}$. It should also be noted that for stationary processes the adaptive algorithm will converge precisely to the optimal solution, that is, $\mathbf{w}(t_k) \longrightarrow \mathbf{w}_{\text{opt}}$. As before, the parameters to tune the *LMS* algorithm are $L$ the filter order, $\triangle$ the step-size parameter and $N_i$ the number of iterations through the data set, where the $i$th-iteration of the weight recursion becomes

$$\mathbf{w}_i(t_{k+1}) = \mathbf{w}_i(t_k) + \triangle_i\epsilon(t_k)\mathbf{X}(t_k) \tag{9.51}$$

The generic structure for the model-reference processor is shown in Figure 9.15. After preprocessing the raw interferometric data with a low-pass filter, the optimal or adaptive surface displacement estimate, is obtained using either the *LWR* or adaptive *LMS* algorithm. Figure 9.16 shows the application of the model-reference processor to some experimental data. In this figure the desired or model reference response predicted by the propagation model for the particular material along with the corresponding optimal (upper trace) and adaptive (lower trace) surface

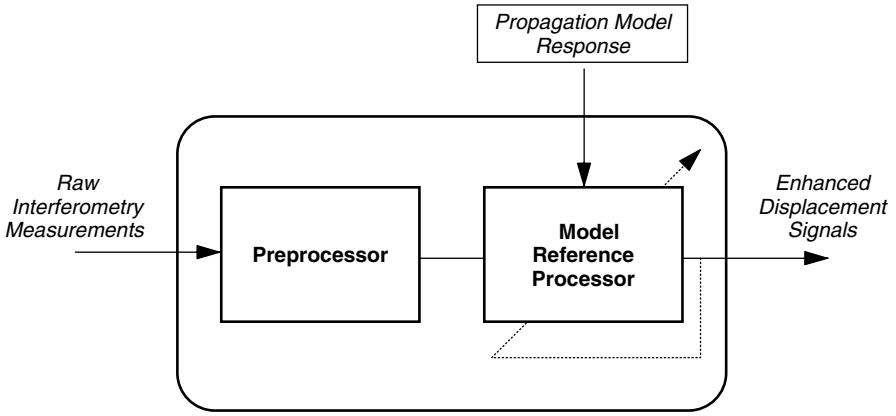



**Figure 9.15.** Model-reference processor structure: optimal and adaptive (*dotted lines*).

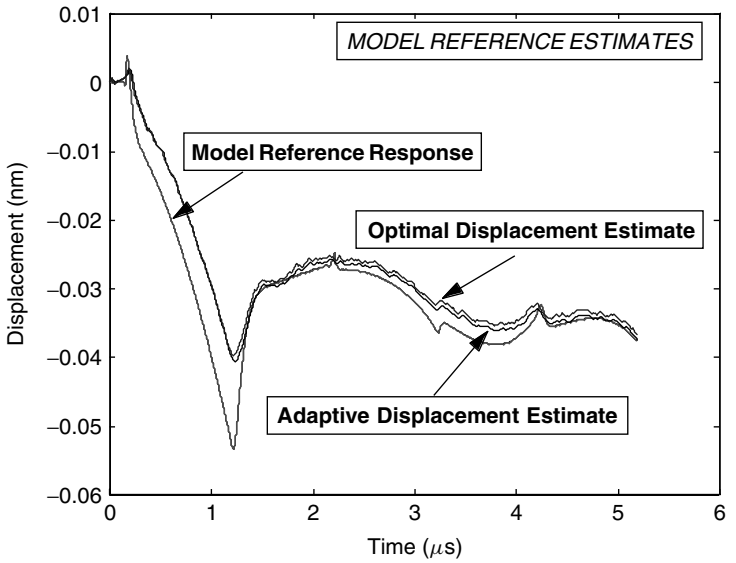



**Figure 9.16.** Typical enhanced surface displacement estimate: model reference, estimate, and adaptive estimate (28 dB *SNR*).

displacement estimates are shown. In this case it is clear that both implementations of the processor perform quite well in extracting and enhancing the signal. For a "real-time" implementation the adaptive processor would be the choice due to its lighter computational load and the fact that it can be tuned to near optimal performance. Next both algorithms are applied to various sets of experimental data.

### 9.2.3 Laser Ultrasonics Experiment

All of the signals recorded using the Michelson interferometer were taken on epicenter in a 6 mm-thick, aluminum alloy plate. Incident laser pulse energy was varied from a maximum of 35 mJ. The temporal origin of the signals is taken at the instant (time = 0) of laser firing. Thus, prior to firing, the initial portion of the data is a record of the system and background noise before the arrival of the laser pulse at the specimen surface. After the firing, electronic noise contaminates the laser generated ultrasonic signal before the arrival of the longitudinal wave. This noise is a problem in that it can continue for hundreds of nanoseconds interfering with the detection of the ultrasonic signal. A typical noisy measurement is shown in Figure 9.17.

The following pair of *SNR*'s are defined to quantify the performance of the processors. On input, the average power in the model-reference signal is defined by

$$P_s \equiv \frac{1}{T}\int_0^T d^2(t)dt \approx \frac{1}{N}\sum_{k=0}^{N} d^2(t_k) \tag{9.52}$$

along with the variance of the noise, $\sigma_n^2$, is used to calculate the input SNR

$$SNR_{\text{in}} \equiv \frac{P_s}{\sigma_n^2} \tag{9.53}$$

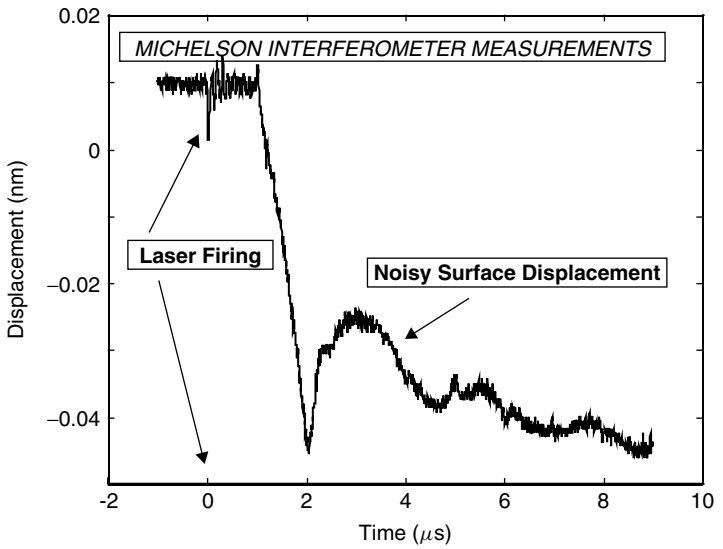



**Figure 9.17.** Typical noisy Michelson interferometer experimental measurement in 6 mm aluminum plate (30 dB *SNR*).

The noise variance is estimated by applying a 100-sample window averaging filter to the measured signal, subtracting this averaged signal and calculating the sample variance

$$\sigma_n^2 = \frac{1}{N}\sum_{k=0}^{N}(x(t_k) - \overline{x}(t_k))^2 \tag{9.54}$$

The output *SNR* uses the same signal average power, but the noise variance in this case is the estimation error variance with the bias removed, since it represents the residual remaining after model-reference processing; that is, with the error defined as before with $\epsilon(t_k) = d(t_k) - \hat{d}(t_k)$, we have

$$\sigma_\epsilon^2 = \frac{1}{N}\sum_{k=0}^{N}(\epsilon(t_k) - \overline{\epsilon}(t_k))^2 \tag{9.55}$$

giving

$$SNR_{\text{out}} \equiv \frac{P_s}{\sigma_\epsilon^2} \tag{9.56}$$

When applying this approach to the experimental data, the procedure consists of (1) preprocessing using a 10-sample window averaging filter (2) aligning/interpolating the preprocessed measurement with the model reference response (inputs to the model-reference processor) (3) designing both optimal and adaptive processors and (4) calculating the overall processing gain. It should be noted that there are two caveats that can limit the performance of the model-reference processor: first, the reference response is assumed to align temporally with the measurement, and second, the required parameters for the reference synthesis are known a priori. Also note that in designing both processors, various filter orders, $(L)$, and regularization factors, $\sigma_n^2$ were selected during the design procedure. For each run the mean-squared error is calculated, and the set that yields the smallest error is chosen. A typical set is $L = 32$ with regularization factor $\sigma_n = 1 \times 10^{-3}$ yielding a mean-squared error on the order of $10^{-6}$.

An example *SNR* calculation run is shown in Figure 9.18 which shows the noisy measurement data, the model-reference signal, estimated noise (mean removal) and the enhanced signal estimate with associated *SNR*. Typically the processing gain is 20 dB or greater. Figure 9.17 showed the results for processing a low *SNR* case while a high *SNR* case is shown in Figure 9.19, which corresponds to the measurement in Figure 9.18. This figure shows the raw measurement data $(x(t_k))$ and the optimal (upper trace) and adaptive (lower trace) processor surface displacement estimates $(\hat{d}(t_k))$ overlayed on the interpolated model reference input $(d(t_k))$. From these runs it is clear that the model-reference processor is capable of increasing the overall sensitivity ($> 20$ dB) of the Michelson interferometer measurements. This statement is further substantiated by calculating the *SNR* (as above) for 11 experimental data sets on the aluminum plate as shown in Table 9.1.

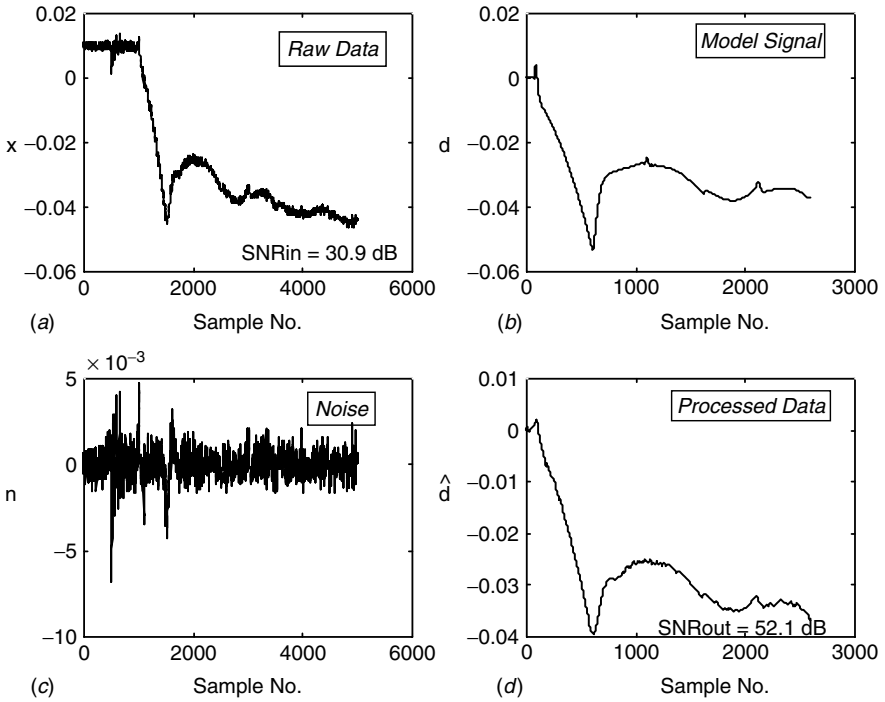



**Figure 9.18.** *SNR* estimation: (*a*) Noisy measurement (30.9 dB *SNR*). (*b*) Predicted model-reference response. (*c*) Estimated noise. (*d*) Optimal surface displacement estimate (52.1 dB *SNR*).

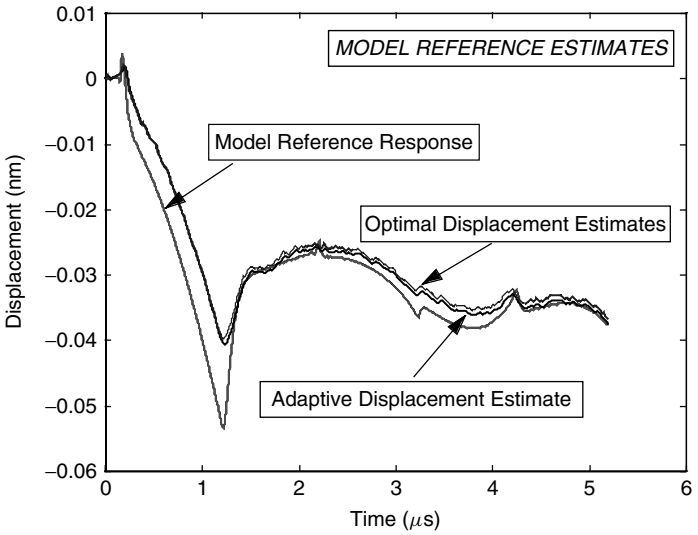



**Figure 9.19.** Enhanced surface displacement estimate: Model reference, estimate, and adaptive estimate (30 dB *SNR*).

**Table 9.1. Model-Based Reference Processing: SNR Gains (dB)**

| Run Number | $SNR_{\text{in}}$ | $SNR_{\text{out}}$ | $SNR_{\text{gain}}$ |
|---|---|---|---|
| 1 | 28.2 | 50.3 | 22.1 |
| 2 | 28.5 | 50.1 | 21.6 |
| 3 | 27.0 | 48.9 | 21.9 |
| 4 | 27.8 | 49.7 | 21.9 |
| 5 | 25.3 | 49.8 | 24.6 |
| 6 | 30.9 | 52.1 | 21.2 |
| 7 | 23.12 | 50.4 | 27.3 |
| 8 | 25.4 | 52.0 | 26.6 |
| 9 | 21.1 | 47.6 | 26.5 |
| 10 | 22.9 | 49.4 | 26.5 |
| 11 | 23.8 | 51.6 | 27.8 |

We summarize the results of this application as follows:

Criterion: $J = \text{trace} \quad E\{\epsilon^2(t)\}$

Models:

Signal: $s(t) = W(q^{-1})d(t)$

Measurements: $y(t) = s(t) + n(t)$

Noise: $n \sim \mathcal{N}(0, R_{nn})$

Algorithm: $\hat{s}(t|t) = \hat{W}(q^{-1})\hat{d}(t|t-1)$

Quality: $R_{\epsilon\epsilon}$

This completes the application on nondestructive evaluation for laser ultrasonics.

### 9.2.4 Summary

The sensitivity of a Michelson interferometric system to laser-generated ultrasonic signals can be enhanced significantly using a model-reference processing approach. A model-reference response for a given specimen may be obtained using a sophisticated propagation model to predict the surface displacement resulting from laser generated ultrasound. Once generated, the reference is used to construct both the optimal and adaptive versions of the processor for potential on-line implementation. Greater than 20 dB gain in output *SNR* or equivalently, overall sensitivity improvement in the interferometric measurements was achieved using this approach.

The application of this model-reference approach requires the development of a reference response based on the underlying physics of the process under investigation, in this case, thermoelastic generation of ultrasound [13]. To achieve a reasonable reference, the specimen and experimental parameters are required a priori, and once developed, the reference must be aligned with the measured signal

or performance of the algorithm could be limited significantly. That is, performance can deteriorate significantly if the reference is not aligned properly with the measurement displacement. For this work the alignment was accomplished interactively. An optimal approach that searches through all the data to find the "best" alignment uses the so-called Simpson sideways recursion (see [14] for details). This completes the application.

## 9.3 *MBP* FOR STRUCTURAL FAILURE DETECTION

Structural systems operating over long periods of time can be subjected to failures due to fatigue caused various external forces, the most severe of which, could be an earthquake. When this happens it is possible for the structure to collapse, causing complete deterioration and catastrophic failure. For instance, failure of the Bay Bridge during the 1989 earthquake. Thus, to attain reliable structural system performance, it is necessary to monitor critical structural components to provide timely information about current and/or imminent failures, and to predict the integrity of the underlying structure.

Standard approaches to detect failure mechanisms at the onset range from a simple accelerometer strategically placed to observe the Fourier spectrum of known response, to using cepstral analysis to identify periodic responses. Measures of failure can deteriorate significantly if noise is present—a common situation in an operational environment. Most of the current monitoring approaches for failure detection and isolation lead to single-channel processing of measured sensor data. Multiple sensors (e.g., accelerometers for vibrations, microphones for acoustics, and thermocouples for temperature) in a structure provide enhanced information about the system. This implies the application of a multichannel (multi-input, multi-output) system representation, which is most easily handled in state-space form without restrictions to single-channel spectral representations. This section is based on developing a model-based signal processing approach to solve the structural system fault detection and classification problem.

Model-based signal processing for structures involves incorporation of the process model (large-scale structure), measurement model (wireless sensor network), and noise models (instrumentation, environmental, parameter uncertainty, etc.) along with measured data into a sophisticated processing algorithm capable of detecting, filtering (estimating), and isolating a mechanical fault in a hostile operational environment. The model-based processor *MBP* will provide estimates of various quantities of high interest (modes, vibrational response, resonances, etc.), as well as on-line statistical performance measures which are especially useful for fault propagation experiments.

The model-based approach for fault detection, classification and failure prediction is shown in Figure 9.20. The basic concept is that the process or structural system under consideration is modeled either from first principles or using system identification techniques [9] to "fit" models to the data. Once the system models are developed, then the sensor suite (or measurement system) models are developed. Usually the bandwidth of each sensor is much wider than the dominant

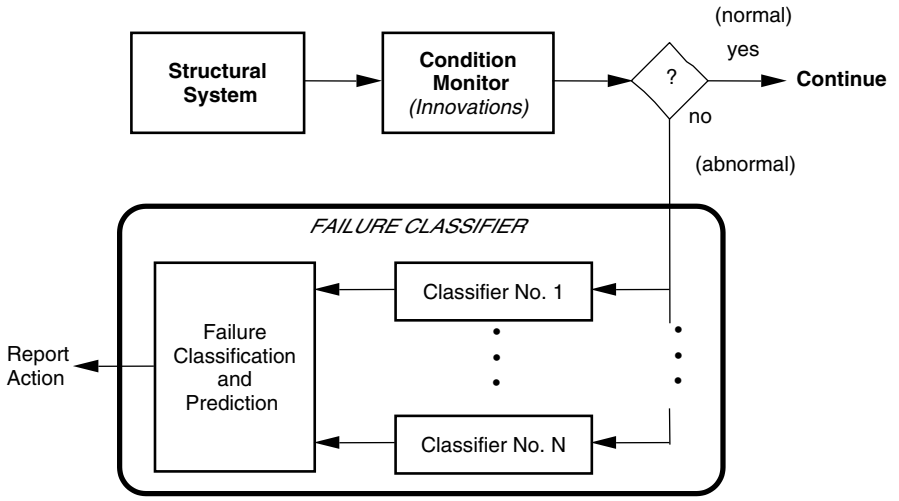



**Figure 9.20.** Detection, classification, and prediction of structural failures using the model-based approach.

dynamics or resonances of the system, and therefore each sensor is represented by a gain. However, if sensor dynamics must be included in the system model, the model-based approach easily accommodates them. Sensor dynamic models can be obtained from manufacturer specifications (transfer function, etc.). After these models are completed, it remains to model the noise. If noise data are unavailable, then a reasonable approach is to model the noise as additive and random, leading to a Gauss-Markov model. Once a representation of the overall system (structure, sensors, and noise) is developed, then a Failure or *Condition Monitor* can be developed to monitor the status of the structural system. Should a fault (i.e., a deviation from the normal operation) be detected, then it must be classified using information from known or measured failures. It is possible to predict the time to failure for the overall system, and decide on the appropriate action, including simply reporting the results. The idea is to develop the *MBP* based on the normal response of the structure (calibration phase) and then develop a monitor to detect the "change from normal" condition which could be due to long term aging or a transient event like an earthquake. Once the "abnormal" condition is *detected*, then it must be *classified* using other a priori information based on the particular type of failure mechanism that is characterized (modeled).

### 9.3.1 Structural Dynamics Model

The model-based approach to detect a fault in a structural system requires a design of a Condition Monitor which can be accomplished by performing the following steps: (1) structural, sensor, and noise model development; (2) simulation; (3) model-based processor design; and (4) prototype testing using real data.

The model development phase for this problem consists of developing models for the structural system in the form of linear dynamical equations. Once identified, a representative model of the system will be developed for this study. A linear, time-invariant mechanical system was selected that can be expressed as

$$M\ddot{d}(t) + C\dot{d}(t) + Kd(t) = p(t) \tag{9.57}$$

where $d$ is the $N_d \times 1$ displacement vector, $p$ is the $N_p \times 1$ excitation force, and $M$, $C$, $K$, are the $N_d \times N_d$ lumped mass, damping function, and spring constant matrices characterizing the structural process model, respectively. The structure of these matrices typically take the form

$$M = \begin{bmatrix} M_1 & 0 & 0 & 0 & 0 \\ 0 & M_2 & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & M_{N_d-1} & 0 \\ 0 & 0 & 0 & 0 & M_{N_d} \end{bmatrix}, \quad C = \begin{bmatrix} c_{ij} \end{bmatrix}$$

$$K = \begin{bmatrix} (K_1 + K_2) & -K_2 & 0 & 0 & 0 \\ -K_2 & (K_2 + K_3) & \ddots & 0 & 0 \\ 0 & \ddots & \ddots & -K_{N_d-1} & 0 \\ 0 & 0 & -K_{N_d-1} & \left(K_{N_d-1} + K_{N_d}\right) & -K_{N_d} \\ 0 & 0 & 0 & -K_{N_d} & K_{N_d} \end{bmatrix}$$

If we define the $2N_d$-state vector as $x(t) := \begin{bmatrix} d(t) & | & \dot{d}(t) \end{bmatrix}$, then the state-space representation of this process can be expressed as

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ \dot{x}(t) &= \begin{bmatrix} 0 & | & I \\ --- & | & --- \\ M^{-1}K & | & M^{-1}C \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ --- \\ M^{-1} \end{bmatrix} p(t) \\ y(t) &= \begin{bmatrix} M^{-1} & | & 0 \end{bmatrix} x(t) \end{aligned} \tag{9.58}$$

where we have assumed that an accelerometer sensor is used to measure the displacement directly at each story, $y \in \mathcal{R}^{N_d \times 1}$. Note that sensor models can capture the dynamics of the sensors, as they interact with the dynamics of the states. For example, in a typical structural system, this equation represents the outputs of a set of accelerometers which are wideband relative to the process.

Consider the special case of a five-story ($N_d = 5$) structure that we will investigate in this application depicted in Figure 9.21 (see [15], [16]). Here the $5 \times 5$ $M$, $C$, $K$ matrices are defined by the following parameters sets: $M$ diagonal with identical values of 4800 lb-s$^2$/in, $C = 0$ lb-s/in, and $K_i = 1.622 \times 10^6$ lb/in. A constant force of 0.5 lb was applied for 30.1 seconds with the data sampled at 0.1 s. The

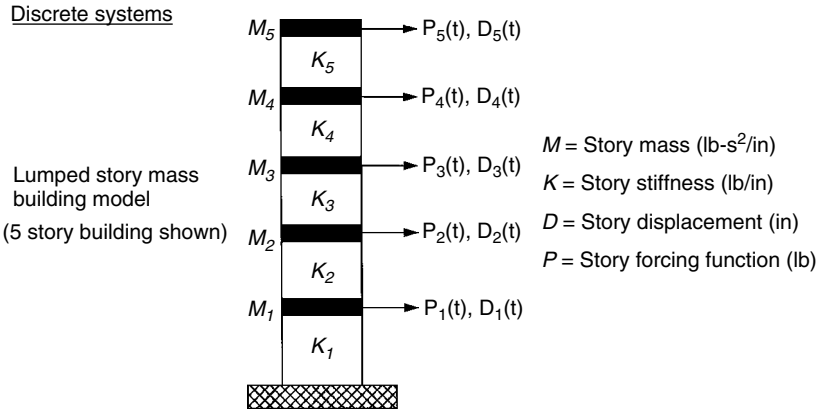



**Figure 9.21.** Structural failure detection problem: Five-story lumped structural model.

resulting deterministic structural simulation shows that the accelerometers measure a sinusoidal (modal) output at each story with the resulting frequencies that match a sophisticated modal simulation: $f = \{0.833, 2.43, 3.83, 4.92, 5.61\}$ Hz.

Noise models can be obtained from experimental measurements (background, instrumentation, etc.). In the absence of such measurements, noise is often modeled as additive, white, and gaussian, resulting in an approximate Gauss-Markov model as discussed previously (see Chapter 2). Within this framework the particular structural system identified will be modeled; it can involve fitting the model parameters ($M$, $C$, $K$ structural matrices) to measured response data. Once these models are developed (as above), a simulator based on these relations is generated and executed to provide synthesized sensor measurements at various signal-to-noise ratios to investigate the performance of the processor. The simulation, typically called a *truth model* (see Section 5.7), can also be used to perform an error analysis during model-based processor design. Additionally it can be used to bound detection and classifier performance as well as predict time to failure. We will discuss the results of the Gauss-Markov model simulation subsequently.

### 9.3.2 Model-Based Condition Monitor

The design of the *MBP* for the structural system also consists of various phases: processor model development, tuning, and error analysis. Development of various designs is usually based on choosing different simplified models, incorporating them into the algorithm, and analyzing the *MBP* performance statistically. The structure of the *MBP* for the structural system under development consists of the processor state (modes, displacement, etc.) and enhanced response measurement (sensors) estimates. The Condition Monitor (Figure 9.20) utilizes as input the innovations sequence, which provides an on-line test of the "goodness of fit" of the underlying structural process model to the data. The innovations sequence plays a major role in the recursive nature of this processor by providing information that

can be used to adaptively correct the processor and the process model itself, as well as the provide input to a sequential detection scheme ([17], [18]). It is critical that the *MBP* be tuned to accurately "track" the normal operation of the structural system; therefore, under such conditions the innovations sequence must be zero-mean and white. Once a fault occurs, it is detected by a change in the normal operating conditions of the system and it must be isolated to determine the severity of the failure and the overall consequences to the structure. The innovations sequence is given by

$$\epsilon(t) = y(t) - \hat{y}(t|t-1) \tag{9.59}$$

where in this application, the measurement is the output of the accelerometers positioned on each floor and $\hat{y}(t|t-1)$ is the predicted measurement (output of the *MBP*). The Condition Monitor design is based on the underlying principle that the *MBP* is "tuned" during the calibration stage to track the structural displacements under normal conditions. Recall from Chapter 5, that when optimal, the innovations are zero-mean and white; therefore we perform the statistical whiteness test (at each floor) testing that 95% of the samples (normalized) innovations correlations lie within the bounds given by

$$\left[\hat{c}_{\epsilon\epsilon}(k) \pm \frac{1.96}{\sqrt{N}}\right], \quad \hat{c}_{\epsilon\epsilon}(k) := \frac{\sigma_\epsilon^2(k)}{\sigma_\epsilon^2(0)} \tag{9.60}$$

for $\sigma_\epsilon^2(k)$ the sample covariance estimated from the data, along with the corresponding zero-mean test defined by

$$\left[\hat{m}_\epsilon(k) < 1.96\sqrt{\frac{\sigma_\epsilon^2(k)}{N}}\right] \tag{9.61}$$

Recall that both tests rely on quasi-stationary assumptions and sample statistics to estimate the required correlations.

When data are nonstationary, as in the case of a nonlinear or time-varying failure, then a more reliable statistic to use is the weighted sum-squared residual (*WSSR*) which is a measure of the overall global estimation performance of the processor again determining the "whiteness" of the innovations sequence [8]. It aggregates all of the sensor (floor accelerometers) information available in the innovations and tests whiteness by requiring that the decision function lies below the specified threshold to be deemed statistically white. Recall that the *WSSR* statistic is estimated over a finite window of $N$-samples, that is,

$$\rho(\ell) = \sum_{k=\ell-N+1}^{\ell} \epsilon'(t) R_{\epsilon\epsilon}^{-1} \epsilon'(t), \qquad \ell \geq N \tag{9.62}$$

The detection algorithm is

$$\rho(\ell) \underset{<}{\overset{\mathcal{H}_1}{>}} \tau \tag{9.63}$$
$$\mathcal{H}_0$$

where $H_0$ is the hypothesis that the vibration data is "normal" (white innovations), while $H_1$ is the hypothesis that there is a change or "abnormal" condition specified by non zero-mean, nonwhite innovations. Under $H_0$, *WSSR* is distributed $\chi^2(N_d N)$. For a large $N_d N > 30$ and a level of significance of $\alpha = 0.05$, the threshold is

$$\tau = N_d N + 1.96\sqrt{N_d N} \tag{9.64}$$

Here the window is designed to slide through the innovations data and estimate its whiteness. Thus overall performance of the processor can be assessed by analyzing the statistical properties of the innovations.

The *MBP* approach to the design of a classifier for multiple faults and potential structural failures is based on developing models of the particular faults and then estimating them directly from the data. For instance, it is known that the resonant frequencies associated with a structural system will decrease in magnitude when a crack develops in the system [19]; that is,

$$K \longrightarrow K - \Delta K \tag{9.65}$$

Alternatively, the equivalent mass changes to

$$M \longrightarrow M + \Delta M \tag{9.66}$$

The occurrence of these parametric changes creates a mismatch between the *MBP* process model and the data, and therefore it no longer tracks the process. As a consequence the innovations sequence becomes nonwhite, indicating the mismatch (see Chapter 5). Faults can be modeled analytically as well. As an example, consider the fatigue of a spring with an exponential decay function, which implies that the spring constant changes according to

$$K \longrightarrow Ke^{-K_\tau t} \tag{9.67}$$

The *MBP* can be developed using empirical relations such as these and the overall system can be examined also. The Failure Classifier (see Figure 9.20) performs a multiple hypothesis test to isolate the failure and classify it accordingly. The implementation of the classifier algorithm requires further development. One approach is to assume specific statistics associated with each particular type of fault (e.g., multivariate gaussian), and develop an optimal Neyman-Pearson multihypothesis scheme which results in a likelihood ratio test [20]. However, if the selected "feature" vectors cannot be characterized by a unique distribution, then an

alternative is to use an adaptive classifier such as the probabilistic neural network [21]. These type classifiers ([22], [23]) have been applied successfully in developing a multichannel classifier applied to the sounds emanating from a prosthetic heart valves. The appropriate classifier for this problem must be developed by the fault models and a priori knowledge.

Once reasonable models of the structural system and individual failure mechanisms are developed, it will be possible, via simulations, to predict the overall operation of the system and its time to failure. For instance, the model given from the structural response as a function of cracking may be represented by replacing the spring constant with the exponential decay mode and calculating the time its takes the singularities (poles) of the structural system to enter the right half plane (instability). Clearly, this leads to a failure model that will be capable of predicting the time to fail as well as its associated statistics.

### 9.3.3 Model-Based Monitor Design

In this subsection we develop the *MBP* and Condition Monitor to detect structural failures. We demonstrate the performance of these processors on simulations based on the five-story structure discussed previously and shown in Figure 9.21. First, we perform the Gauss-Markov simulation with process noise covariance matrix diagonal with variance, $\sigma_w^2 = 10^{-6}$ and measurement (accelerometer) noise variance also diagonal with variance, $\sigma_v^2 = 10^{-2}$. The initial displacements were assumed to be 1 inch with an uncertainty of $P_o = 10^{-4}$.

The calibration data was synthesized using *SSPACK_PC* [24] with the results for the normal run shown in Figure 9.22. As observed, the innovations are zero-mean and white as demonstrated by the individual whiteness tests performed on the innovations obtained on each floor. The *WSSR* also verifies this condition lying below the critical threshold which indicates a normal calibration output. Next an "abnormal" data set was generated by changing the stiffness values at the first floor, that is, $K \longrightarrow K - \Delta K$ with $\Delta K_5 = 0.9 \times K_5$. The results are shown in Figure 9.23 with the raw accelerometer data in 9.23*b* along with the non-zero mean and nonwhite innovations in 9.23*c*. We also note that the *WSSR* threshold is exceeded indicating a structural failure that must be further investigated by the classifier. Performing the additional analysis, we compare the power spectra of the innovations for both cases and the results are shown in Figure 9.24. We see relatively *flat* spectrum in the normal case, but sharp resonances (arrows) for the abnormal case indicating that the model does not match the data. From these results it appears that the model-based approach provides a feasible solution to the structural failure detection problem. Next we consider the application of this approach to a real-world problem.

### 9.3.4 *MBP* Vibrations Application

In this subsection we discuss the application of a *MBP* to a vibrating structure using a simple one-dimensional version of the model developed above. The application

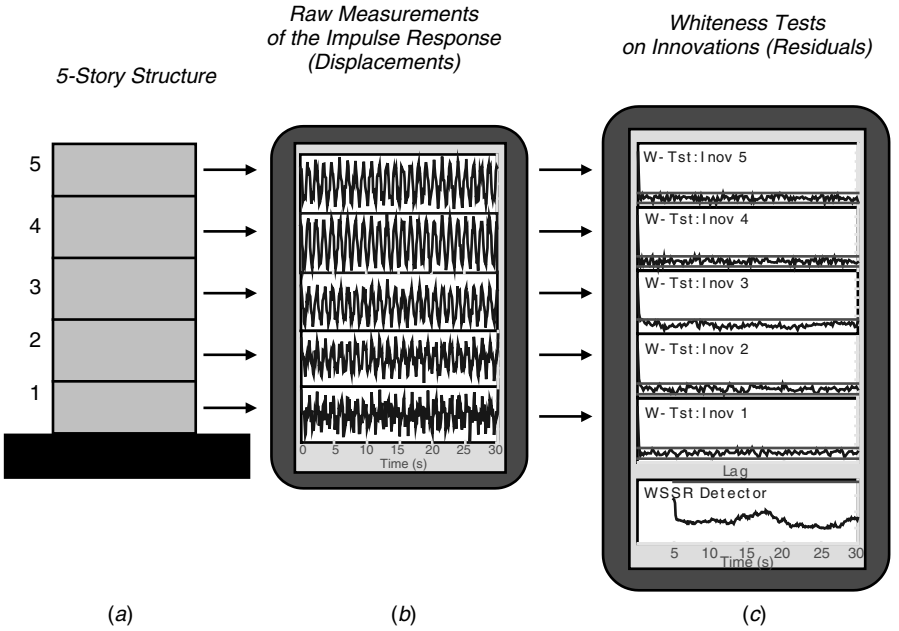



**Figure 9.22.** Structural failure detection problem: Calibration run for ``normal'' conditions: (*a*) Five-story structure. (*b*) Raw accelerometer data measured at each floor. (*c*) Innovations and WSSR detection results (zero-mean/white).

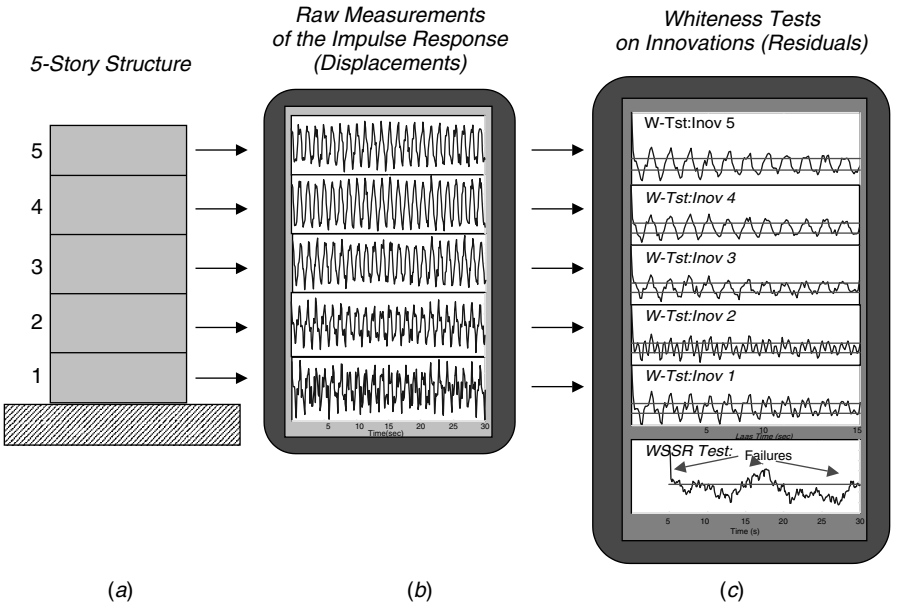



**Figure 9.23.** Structural failure detection problem: Calibration run for ``abnormal'' conditions: (*a*) Five-story structure. (*b*) Raw accelerometer data measured at each floor. (*c*) Innovations and WSSR detection results (non zero-mean/nonwhite).

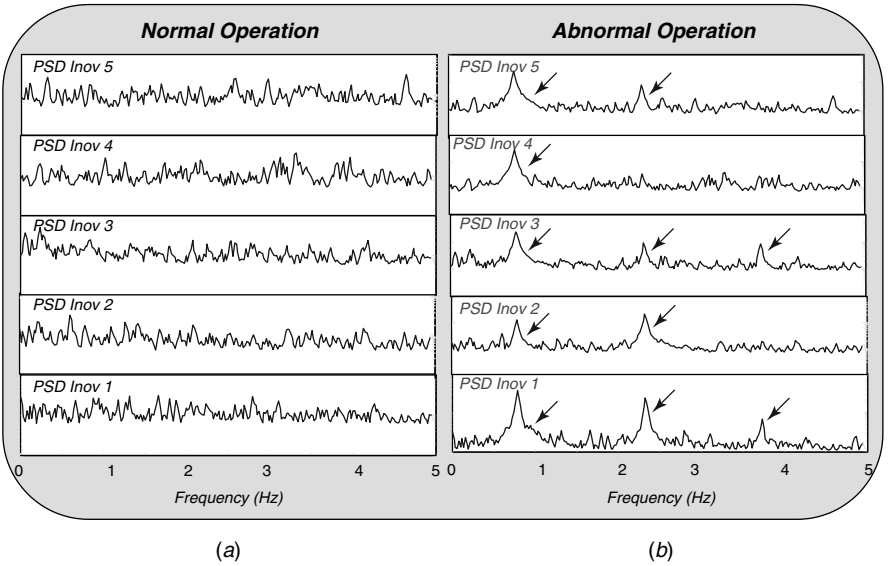



**Figure 9.24.** Structural failure detection problem: Innovations power spectral estimates. (*a*) Normal condition (no resonances). (*b*) Abnormal condition (resonances).

[26] is concerned with the nondestructive evaluation of parts for failure detection. An experiment was designed to investigate the feasibility of model-based signal processing techniques for failure detection. A given structure or part under investigation was excited and its shock response recorded using an accelerometer. The response data was digitized and recorded as depicted in Figure 9.25. The responses of two parts were measured: one good and one bad also shown in 9.25*a* and 9.25*b* of the figure. The approach we take is to characterize the model parameters by estimating them from measured response data from the good (nondefective) part and then constructing the *MBP* described previously to test the bad (defective) part for failure. Our search is for a technique that can be used on a production line to perform these evaluations in real time.

The basic structural model was to use the differential equations with scalar measurement model, that is,

$$\ddot{d}(t) + \frac{c}{m}\dot{d}(t) + \frac{k}{m}d(t) = \frac{1}{m}p(t) + w(t)$$

or in terms of normal (mode) coordinates[1]

$$\ddot{\tilde{d}}(t) + 2\sigma\dot{\tilde{d}}(t) + \left(\sigma^2 + \omega^2\right)\tilde{d}(t) = \tilde{p}(t) + \tilde{w}(t)$$

[1]The normal mode coordinates, $\tilde{d}$'s are simply a set of coordinates obtained by transformating the physical system coordinates $d$'s through a similarity transformation ([8], [15]) to eigenvector-eigenvalue form with the eigenvectors defined as the modal functions.

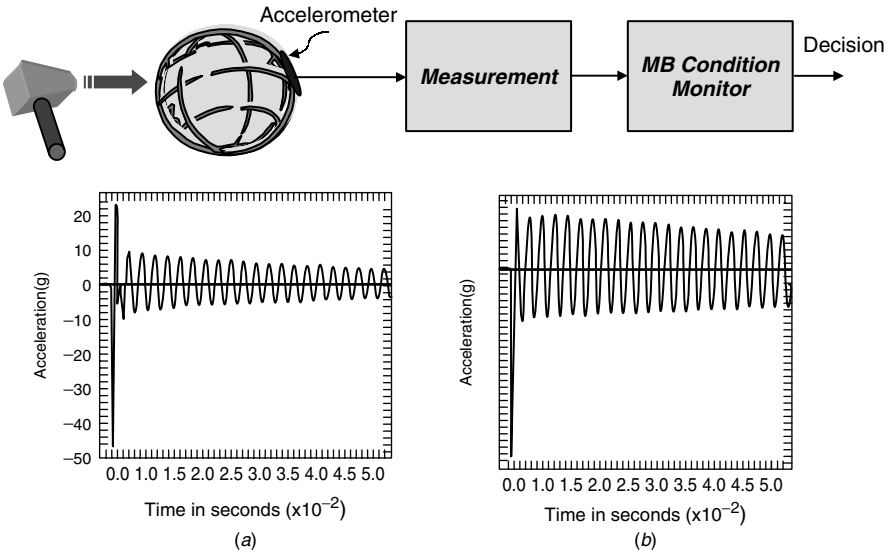



**Figure 9.25.** Structural failure detection problem for part and test setup with hammer, part, measurement and condition monitor. (*a*) Good (normal) part shock response. (*b*) Bad (abnormal) part shock response.

and

$$\tilde{y}(t) = \tilde{d}(t) + \tilde{v}(t)$$

where $\tilde{d}(t)$ is the displacement, $\tilde{y}(t)$ is the sampled accelerometer measurement contaminated with gaussian noise sources $\tilde{w}(t)$, $\tilde{v}(t)$ with respective covariances, $R_{\tilde{w}\tilde{w}}$ and $R_{\tilde{v}\tilde{v}}$ with $\tilde{p}(t)$ the shock input (impulse input function). It can be shown [26] that the parameters are: $\sigma = \zeta\omega_n$ and $\omega = \omega_n\sqrt{1-\zeta^2}$ with $\zeta$ related to the damping coefficient and $\sigma$ related to the attenuation. In terms of the original $MCK$ model: $\omega_n = \sqrt{\frac{k}{m}}$ with $\sigma = \frac{c}{\sqrt{km}}$. Once the parameters are estimated, the $MBP$ and therefore, the Condition Monitor can be constructed.

For this coordinate system, two-dimensional state-space model with $x(t) := \left[\begin{array}{c|c} \tilde{d}(t) & \dot{\tilde{d}}(t) \end{array}\right]$ takes the form

$$\frac{d}{dt}x(t) = \left[\begin{array}{c|c} -\sigma & \omega \\ \hline -\omega & -\sigma \end{array}\right] x(t) + \left[\begin{array}{c} 0 \\ \hline 1 \end{array}\right] \tilde{p}(t)$$

$$\tilde{y}(t) = [1 \mid 0\,]\, x(t) \tag{9.68}$$

In any case a parameter estimation using the AMBP was performed on the shock response of the good part with the following parameters identified: $\hat{\omega} = 2\pi\,\hat{f}$ where $\hat{f} = 451.2 \pm 0.4$ Hz and $\hat{\sigma} = 19.47 \pm 0.05$. Using these parameters in Eq. (9.68) a $MBP$ and condition monitor can now be constructed.

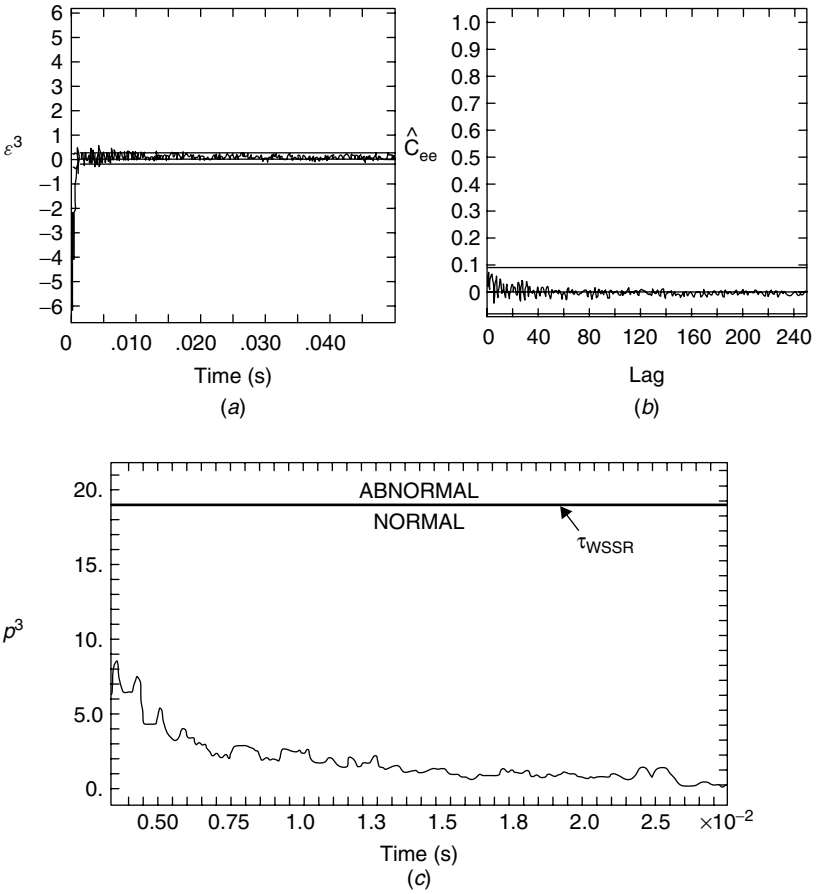



**Figure 9.26.** Structural failure detection problem for ''good'' (nondefective) part: (*a*) Innovations with bounds. (*b*) Whiteness test (0% out). (*c*) *WSSR* detector below threshold (normal).

The Condition Monitor using the *MBP* was calibrated using the good part and its estimated parameters. The resulting innovations sequence with bounds predicted by the *MBP* and whiteness test shown in Figure 9.26 respectively. The processor is tuned over the sampled data and is zero-mean and white. The *WSSR* is below the threshold indicating a normal part. Next the defective part was shocked, its response measured and the *WSSR* detector executed, the results are shown in Figure 9.27. It is clear from the innovations in 9.27*a* that the *MBP* does not match the defective part and this is further demonstrated by the *WSSR* test where the defect is almost detected immediately indicating a bad part which can then be classified. This completes the structural failure detection application using a model-based Condition Monitor based on a linear *MBP* design coupled with innovations type detectors again demonstrating the feasibility of the approach.

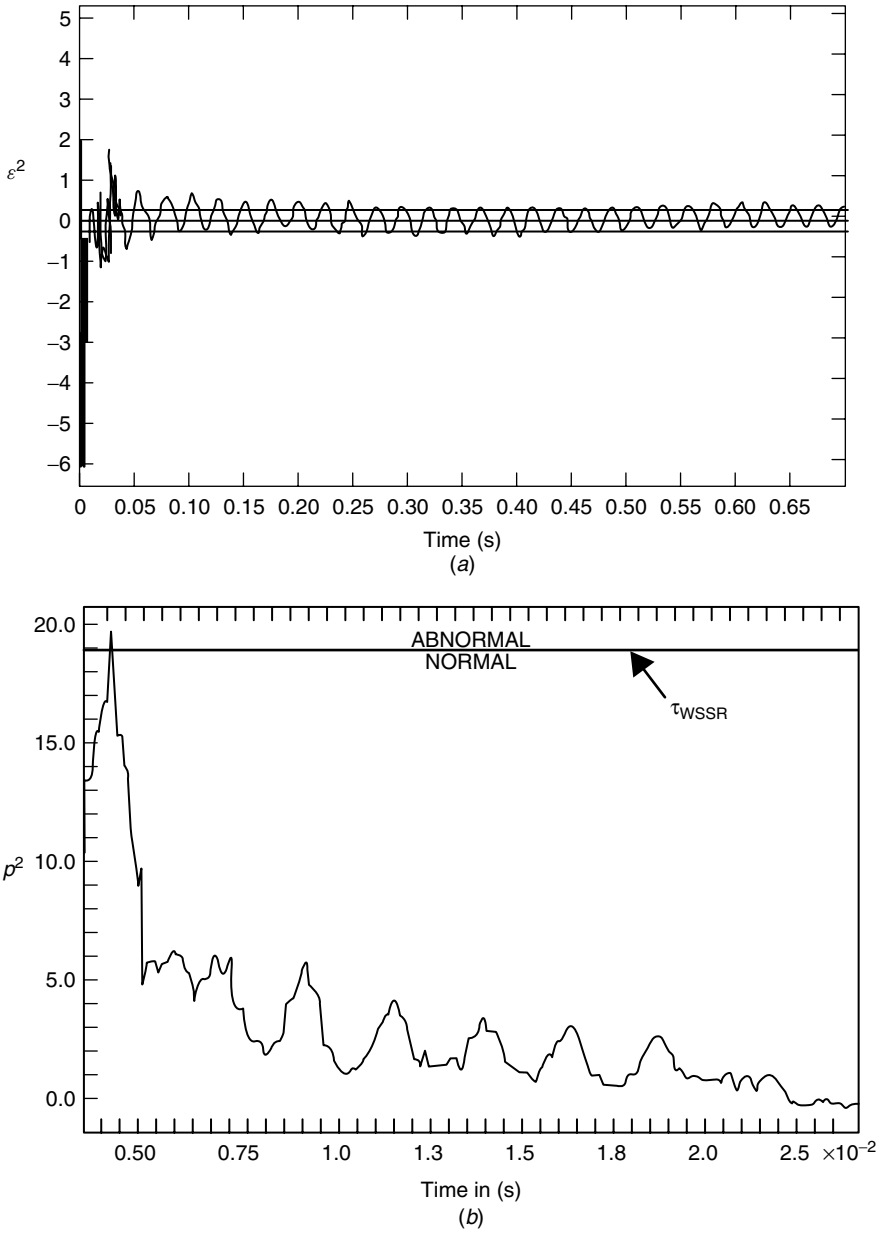



**Figure 9.27.** Structural failure detection problem for ``bad'' part: (*a*) Innovations exceed bounds. (*b*) *WSSR* detection above threshold.

We summarize the results of this application as follows:

| | |
|---|---|
| Criterion: | $J = \text{trace}\ \ \tilde{P}(t\|t)$ |
| Models: | |
| Signal: | $\ddot{d}(t) = -M^{-1}C\dot{d}(t) - M^{-1}Kd(t) + p(t) + w(t)$ |
| Measurements: | $y(t) = d(t) + v(t)$ |
| Noise: | $w \sim \mathcal{N}(0, R_{ww})$ and $v \sim \mathcal{N}(0, R_{vv})$ |
| Algorithm: | $\hat{d}(t\|t) = \hat{d}(t\|t-1) + K(t)e(t)$ |
| Quality: | $\tilde{P}(t\|t) = (I - K(t)C(t))\tilde{P}(t\|t-1)$ |

This completes the application the Condition Monitor for structural failure detection.

### 9.3.5 Summary

In this section we developed the idea of structural failure detection using a model-based Condition Monitor concept based on the innovations sequence of the incorporated *MBP*. We showed that the monitor actually determines the whether the innovations sequence of the structure under investigation matches the "normal" structure with the *MBP* tuned during the calibration stage of the development. We developed the underlying models based on the dynamics of a linear structural dynamic system. A five-story structure was simulated [15] and the *MBP* and accompanying Condition Monitor developed and analyzed. It was shown that the Condition Monitor could perform quite well to distinguish between normal and abnormal responses and detect the failures. Finally an application to real vibration shock data was discussed demonstrating the effectiveness of the model-based Condition Monitor. It was shown that the technique was a feasible method that could be applied to discriminate good and bad parts for nondestructive evaluation purposes.

## 9.4 *MBP* FOR PASSIVE SONAR DIRECTION-OF-ARRIVAL AND RANGE ESTIMATION

In this section we discuss the application of model-based processing techniques to the development of a processor capable of estimating the bearing of a fixed target from data acquired from a towed array of hydrophone sensors. The signal and measurement systems are converted into state-space form, allowing the unknown parameters of the model, such as multiple target bearings, to be estimated by a parameterically adaptive model-based processor (*AMBP*). A major advantage of this approach is that there is no inherent limitation to the degree of sophistication of the models used, and therefore it can deal with other than plane wave models, such as cylindrically or spherically spreading propagation models as well as more sophisticated wave representations.

Here we consider both the simple plane wave and spherical wave propagation models discussed previously in Chapter 2 and apply it to the problem of multiple target direction of arrival (*DOA*) and range estimation. We will see that this multichannel, adaptive approach casts the estimation problem into a rather general form that *eliminates* the need for an explicit beamformer as in much of the work performed in the literature [27], [28], [29], while evolving into a dynamic synthetic aperture structure in a natural way [30]. In model-based array processing [18], [31], it will become apparent that dynamic synthetic aperture[2] forms a framework in which to view this approach, since it is a time-evolving spatial process involving a moving array [32].

### 9.4.1 Model-Based Adaptive Array Processing for Passive Sonar Applications

Here we investigate the model-based solution to the space-time array processing problem by developing a general form of the model-based processor design with various sets of unknown parameters. We define the acoustic *array space-time processing problem* as follows:

> **GIVEN** a set of noisy pressure-field measurements and a horizontal array of $L$-sensors. **FIND** the "best" (minimum error variance) estimate of the target: *DOA* (bearing), range (where applicable), temporal frequencies and amplitudes.

We use the following nonlinear pressure-field measurement model for $M$ monochromatic plane wave targets. We will characterize each of the targets by a corresponding set of temporal frequencies, bearings, and amplitudes, $[\{\omega_m\}, \{\theta_m\}, \{a_m\}]$. That is,

$$p(x, t_k) = \sum_{m=1}^{M} a_m e^{j\omega_m t_k - j\beta(t_k)\sin\theta_m} + n(t_k) \tag{9.69}$$

where

$$\beta(t_k) = k_o(x(t_o) + vt_k) \tag{9.70}$$

and $k_o = 2\pi/\lambda_o$ is the wavenumber, $x(t_k)$ is the current spatial position along the $x$-axis in meters, $v$ is the array speed (m/s), and $n(t_k)$ is additive random noise. The inclusion of the motion in the generalized wave number, $\beta$, is critical to the improvement of the processing, since the synthetic aperture effect is actually created through the motion itself and not simply the displacement.

[2]A synthetic aperture is a fixed element moving array which "traces" out a larger array enabling a completely coherent structure with improved resolution.

If we further assume that the single sensor equation above is expanded to include an array of $L$-sensors, $x \longrightarrow x_\ell$, $\ell = 1, \ldots, L$, then we obtain

$$p(x_\ell, t_k) = \sum_{m=1}^{M} a_m e^{j\omega_m t_k - j\beta_\ell(t_k)\sin\theta_m} + n_\ell(t_k) \tag{9.71}$$

This expression can be written in a more compact vector form as

$$\mathbf{p}(t_k) = \mathbf{c}_m(t_k; \boldsymbol{\Theta}) + \mathbf{n}(t_k) \tag{9.72}$$

where $\mathbf{p}, \mathbf{c}_m, \mathbf{n} \in C^{L\times 1}$, are the respective pressure-field, measurement and noise vectors and $\boldsymbol{\Theta}$ represents the parameters of the plane wave targets and

$$\begin{aligned}
\mathbf{p}(t_k) &= [p(x_1, t_k) \quad p(x_2, t_k), \ldots, \quad p(x_L, t_k)]' \\
\mathbf{c}_m(t_k; \boldsymbol{\Theta}) &= [c(1, t_k; \boldsymbol{\Theta}) \quad c(2, t_k; \boldsymbol{\Theta}), \ldots, \quad c(L, t_k; \boldsymbol{\Theta})]' \\
\mathbf{n}(t_k) &= [n_1(t_k) \quad n_2(t_k), \ldots, \quad n_L(t_k)]'
\end{aligned}$$

The corresponding vector measurement model has been defined in terms of its component parameters, that is,

$$\mathbf{c}_m(t_k; \boldsymbol{\Theta}) := c_m(\ell, t_k; \boldsymbol{\Theta}) = \sum_{m=1}^{M} \alpha_m(t_k) e^{-j\beta_\ell(t_k)\sin\theta_m} \tag{9.73}$$

with $\alpha_m(t_k) := a_m(t_k)e^{j\omega_m t_k}$.

Since we model these parameters as constants ($\dot{\boldsymbol{\Theta}} = \mathbf{0}$), then an *augmented* Gauss-Markov state-space model evolves from first differences (as before)

$$\begin{bmatrix} \theta(t_k) \\ - - - \\ \omega(t_k) \\ - - - \\ \mathbf{a}(t_k) \end{bmatrix} = \begin{bmatrix} \theta(t_{k-1}) \\ - - - \\ \omega(t_{k-1}) \\ - - - \\ \mathbf{a}(t_{k-1}) \end{bmatrix} + \Delta t_k \mathbf{w}(t_{k-1}) \tag{9.74}$$

where $\theta(t_k) := [\theta_1(t_k) \cdots \theta_M(t_k)]'$, $\omega(t_k) := [\omega_1(t_k) \cdots \omega_M(t_k)]'$, $\mathbf{a}(t_k) := [a_1(t_k) \cdots a_M(t_k)]'$, and $\mathbf{w}$ is a zero mean, gaussian random vector with covariance matrix, $R_{ww}$ and $\triangle t_k := t_k - t_{k-1}$. Define the composite parameter vector, $\boldsymbol{\Theta}$ as

$$\boldsymbol{\Theta} := \begin{bmatrix} \theta \\ - - - \\ \omega \\ - - - \\ \mathbf{a} \end{bmatrix} \tag{9.75}$$

for $\boldsymbol{\Theta} \in R^{3M\times 1}$. Then the following *augmented* state prediction equation evolves for the model-based processor:

$$\hat{\boldsymbol{\Theta}}(t_k|t_{k-1}) = \hat{\boldsymbol{\Theta}}(t_{k-1}|t_{k-1}) \tag{9.76}$$

Since the state-space model is linear with no explicit dynamics, the process matrix $A = I$ (identity) and the prediction relations are greatly simplified. On the other hand, the correction equations are nonlinear due to the plane wave measurement model. This leads to the *AMBP* solution of Chapter 8, where the nonlinearities are approximated by a first-order Taylor series expansion. Here we require the measurement jacobian,

$$\mathbf{C}(t_k, \Theta) := \frac{\partial \mathbf{c}_m(t_k; \boldsymbol{\Theta})}{\partial \boldsymbol{\Theta}} = [\nabla_\theta \mathbf{c}_m(t_k; \boldsymbol{\Theta}) \mid \nabla_\omega \mathbf{c}_m(t_k; \boldsymbol{\Theta}) \mid \nabla_{\mathbf{a}} \mathbf{c}_m(t_k; \boldsymbol{\Theta})] \tag{9.77}$$

an $L \times 3M$ complex matrix.

Now we can calculate the required measurement jacobian matrix from these relations for $m = 1, \ldots, M$ as

$$\begin{aligned}
\nabla_{\theta_m} \mathbf{c}(\ell, t_k, \Theta) &= -j\alpha_m(t_k)\cos\theta_m e^{-j\beta_\ell(t_k)\sin\theta_m}, && m = 1, \ldots, M \\
\nabla_{\omega_m} \mathbf{c}(\ell, t_k, \Theta) &= -jt_k\alpha_m(t_k) e^{-j\beta_\ell(t_k)\sin\theta_m}, && m = 1, \ldots, M \\
\nabla_{a_m} \mathbf{c}(\ell, t_k, \Theta) &= e^{j(\omega_m t_k - \beta_\ell(t_k)\sin\theta_m)}, && m = 1, \ldots, M
\end{aligned} \tag{9.78}$$

Given these equations, we are now in a position to construct the *AMBP* estimate of the parameter (state) vector, $\boldsymbol{\Theta}$. The steps of the *AMBP* algorithm (Chapter 8) are as follows:

1. Given an initial or trial value of the *parameter (state) estimate*, $\hat{\boldsymbol{\Theta}}(t_{k-1}|t_{k-1})$, the parameter prediction equation is used to predict the value of $\hat{\boldsymbol{\Theta}}(t_k|t_{k-1})$. This calculation constitutes a prediction of the state vector for $t = t_k$ based on the data up to $t = t_{k-1}$.
2. The *innovation*, $\epsilon(t_k)$, is then computed as the difference between the new measurement taken at $t = t_k$ and the predicted measurement obtained by substituting $\hat{\boldsymbol{\Theta}}(t_k|t_{k-1})$ into the measurement equation, that is,

$$\epsilon(t_k) = \mathbf{p}(t_k) - \hat{\mathbf{p}}(t_k|t_{k-1}) = \mathbf{p}(t_k) - \mathbf{c}(t_k, \hat{\Theta}) \tag{9.79}$$

3. The *gain* or *weight*, $\mathbf{K}(t_k)$ is computed.
4. The *corrected estimate*, $\hat{\boldsymbol{\Theta}}(t_k|t_k)$, is then computed from

$$\hat{\boldsymbol{\Theta}}(t_k|t_k) = \hat{\boldsymbol{\Theta}}(t_k|t_{k-1}) + \mathbf{K}(t_k)\epsilon(t_k) \tag{9.80}$$

5. This corrected estimate, $\hat{\boldsymbol{\Theta}}(t_k|t_k)$, is then substituted into the prediction equation, $\hat{\boldsymbol{\Theta}}(t_k + 1|t_k)$, thereby initiating the next recursion.

This completes the discussion of the model-based array processor design, next we present some applications based on synthesized data.

### 9.4.2 Model-Based Adaptive Processing Application to Synthesized Sonar Data

In this subsection we will evaluate the performance of the *AMBP* to synthesized data assuming that there are two plane wave targets. We will also assume that the two targets are operating at the same frequency, $\omega_o$. Although, in principle, the speed of the array's motion, $v$, is observable, sensitivity calculations have shown that the algorithm is sufficiently insensitive to small variations in $v$ to the extent that measured ship speed will suffice as an input value. Finally, we will reduce the number of amplitude parameters, $\{a_m\}, \quad m = 1, 2, \ldots, M$, from two to one by rewriting the pressure relation as

$$p(x_\ell, t_k) = a_1 e^{j\omega_o t_k}[e^{-j\beta_\ell(t_k)\sin\theta_1} + \delta e^{-j\beta_\ell(t_k)\sin\theta_2}] + n_\ell(t_k) \tag{9.81}$$

Here $\delta = a_2/a_1$ and $\omega_1 = \omega_2 = \omega_0$. The parameter $a_1$ appearing outside the square brackets can be considered to be a data scaling parameter. Consequently we have four parameters so that our measurement equation becomes

$$p(x_\ell, t_k) = e^{j\omega_o t_k - j\beta_\ell(t_k)\sin\theta_1} + \delta e^{j\omega_o t_k - j\beta_\ell(t_k)\sin\theta_2}] + n_\ell(t_k) \tag{9.82}$$

simplifying the parameter vector to

$$\boldsymbol{\Theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ - - - \\ \omega_o \\ - - - \\ \delta \end{bmatrix} \tag{9.83}$$

We now have all the necessary equations to implement model-based algorithm. The calculations are carried out in *MATLAB* [25] using the *SSPACK_PC* Toolbox [24].

In the following plane wave examples, we assume that the two targets are radiating narrow band energy at a frequency of 50 Hz. The true values of the two bearings, $\theta_1$ and $\theta_2$, are $45^o$ and $-10^o$, respectively. The true amplitude ratio $\delta$ is 2. The corresponding initial values for $\theta_1$, $\theta_2$, $f_o = \omega_o/2\pi$, and $\delta$ are $43^\circ$, $-8^\circ$, 50.1 Hz, and 2.5, respectively. The problem geometry is shown in Figure 9.28.

*Case 1.* The number of hydrophones is 4, the array's speed of motion is 5 m/s, and the SNR on the unit amplitude hydrophone is 0 dB. Since the duration of the signal is 27 seconds, the array traces out an aperture of 6λ, a factor of four increase over the 1.5λ physical aperture. The parameters being estimated are $\Theta_1$, $\Theta_2$, $f_0 = \omega_0/2\pi$, and $\delta$ with the results shown in Figure 9.29. These estimates are clearly converging to the true values.

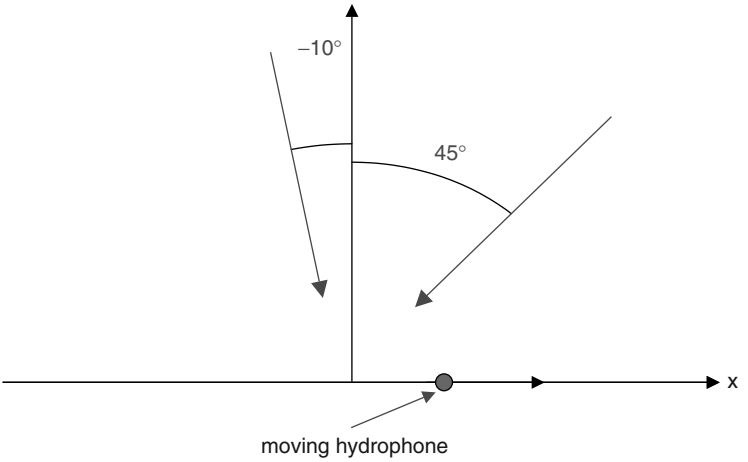



**Figure 9.28.** Plane wave estimation problem: Target arrivals at bearings of −10° and 45°.

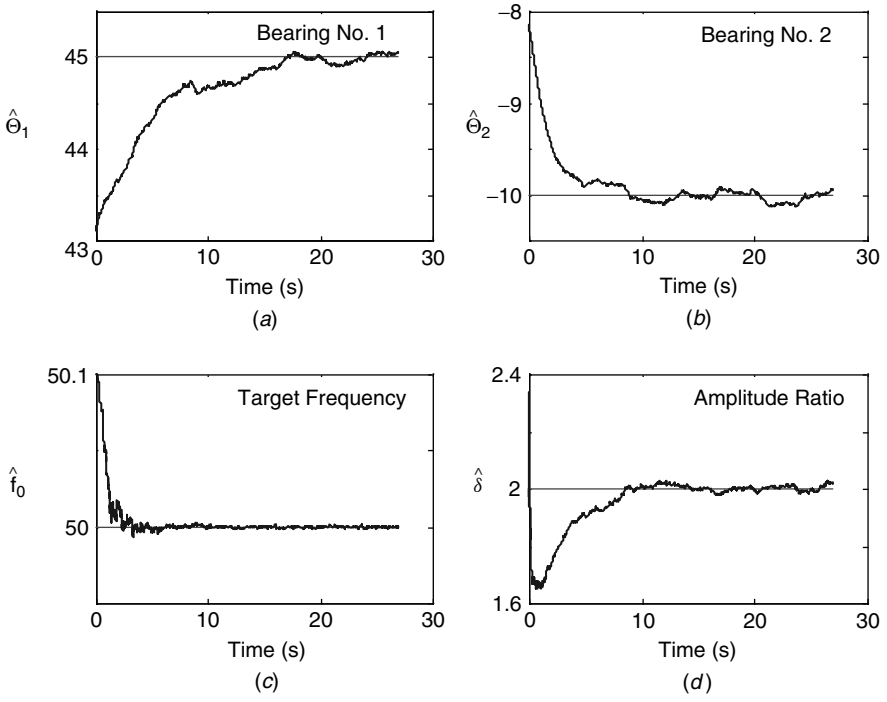



**Figure 9.29.** Case 1: Four-element array at 0 dB *SNR*: (*a*) Target bearing 1 estimate. (*b*) Target bearing 2 estimate. (*c*) Temporal frequency estimate. (*d*) Target amplitude ratio.

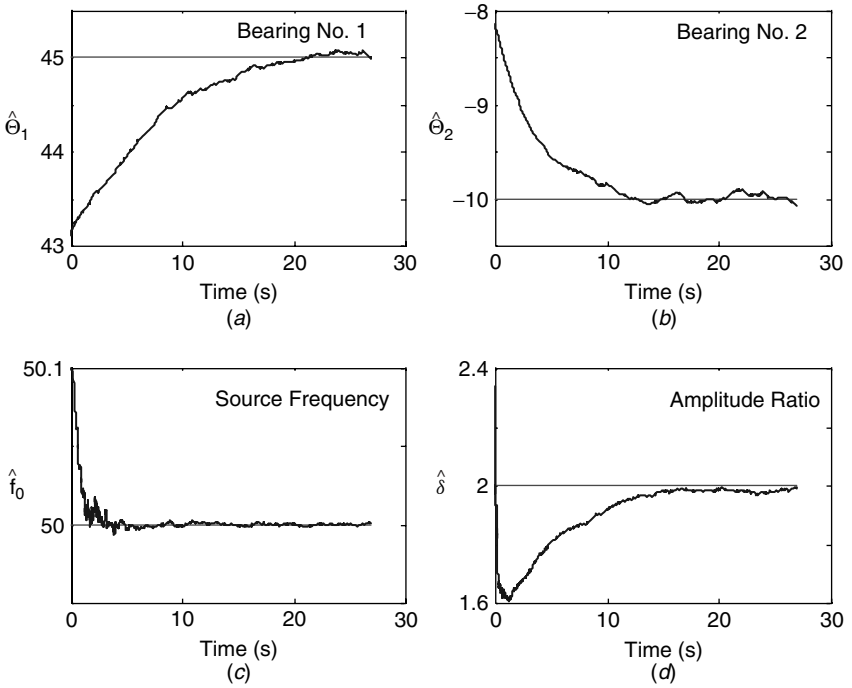



**Figure 9.30.** Case 2: Eight-element array at 0 dB *SNR*: (*a*) Target bearing 1 estimate. (*b*) Target bearing 2 estimate. (*c*) Temporal frequency estimate. (*d*) Target amplitude ratio.

**Table 9.2. Predicted Variances of Test Cases**

| | *Towed Array Synthesis Experiment* | |
|---|---|---|
| Case Number | Bearing Variance ($\Theta_1$) ($\text{deg}^2$): | Bearing Variance ($\Theta_2$) ($\text{deg}^2$) |
| 1 ($\delta = 2$) | $7.5 \times 10^{-5}$ | $24.0 \times 10^{-6}$ |
| 2 ($\delta = 2$) | $2.5 \times 10^{-5}$ | $6.0 \times 10^{-6}$ |

Note: True parameter values: $\theta_1$, $\theta_2$, $f_0$, and $\delta$ are: $45^\circ$, $-10^\circ$, 50 Hz.

*Case 2.* This is the same as the preceding example except that the number of hydrophones has been increased to eight increasing *SNR* and resolution. As can be seen in Figure 9.30, the quality of the estimates is significantly improved over the four-hydrophone case.

The predicted variances of the estimates for the *AMBP* are given in Table 9.2 where it is seen that, for the eight hydrophone moving array $v = 5$ m/s, the variance on the estimate of $\theta_1$, that is, the bearing associated with the signal with $SNR = 0$ dB, is $2.5 \times 10^{-5}\text{deg}^2$ whereas for the $v = 0$ case the predicted variance increases

to $14.0 \times 10^{-5}\text{deg}^2$. This is an improvement of approximately a factor of five in the moving array case over that where the array is stationary. We see in both cases that the associated variances are quite good. Next we consider the spherical wave case.

### 9.4.3 Model-Based Ranging

In this subsection we extend the *AMBP* to include both *DOA* and range estimates. Following [32], we investigate the model-based approach to estimate the range, bearing, and target temporal frequency using a hydrophone array as before. We assume that the wavefront is curved rather than planar as in the previous problem. This type of formulation enables the *time delay* between sensors and wavefront to be characterized by their position as a function of time. Once formulated, the problem is expressed in the model-based framework and again solved using the *AMBP* algorithm for parameter estimation.

Recall that the solution to the free-space spherical wave equation is given by the spatial pressure field

$$p(\mathbf{r};t) = \frac{1}{\sqrt{|\mathbf{r}|}}\mathbf{e}^{\mathbf{j}(\omega_\mathbf{o}t - \mathbf{k_o}\cdot\mathbf{r})} = \frac{1}{\sqrt{|\mathbf{r}|}}\mathbf{e}^{\mathbf{j}\omega_\mathbf{o}\left(t-\frac{|\mathbf{r}|}{c}\right)} \tag{9.84}$$

The relations above can be interpreted as a cylindrical wave[3] propagating outward from the origin (see Figure 9.31) having a temporal frequency of $\omega_o$ and a spatial frequency or wavenumber $\mathbf{k}_o$ with corresponding *dispersion relation*

$$|\kappa_o|^2 = \left(\frac{\omega_o}{c}\right)^2 \tag{9.85}$$

It is easy to show from the above relations that the distance from $r$ to $r_\ell$ traveled by the wavefront in Cartesian coordinates is given by

$$\sqrt{|r - r_\ell|} = \sqrt{(x - x_\ell)^2 + (y - y_\ell)^2} \tag{9.86}$$

Consider the application of an $L$-element sensor array sampling the propagated pressure field. In this case $\mathbf{r}$ is a set of discrete samples, $\{\mathbf{r}_\ell\}$ for $\ell = 1, \ldots, L$. The signal received by the array, $p(\mathbf{r};t)$, is characterized by a time delay, that is,

$$\mathbf{p}(t) = \begin{bmatrix} p_1(t) \\ \vdots \\ p_L(t) \end{bmatrix} = \begin{bmatrix} p\left(t - \tau_1(\Theta)\right) \\ \vdots \\ p\left(t - \tau_L(\Theta)\right) \end{bmatrix} \tag{9.87}$$

where the set of relative *delay times* are $\{\tau_\ell(\Theta)\}$, $\ell = 1, \ldots, L$.

Now let us formulate this problem in terms of the cylindrical wavefront as shown in Figure 9.31 using a horizontal array, that is, $y_i = 0$. In this case, using the law of cosines,[4] we have (from the figure) that the *incremental range curvature*

[3]The 3D spherical wave equation reduced to 2D in cylindrical coordinates is the cylindrical wave equation.

[4]The law of cosines defined for a triangle with angles, $\{A, B, C\}$ and opposite sides, $\{a, b, c\}$ is $a = \sqrt{b^2 + c^2 - 2bc\cos A}$.

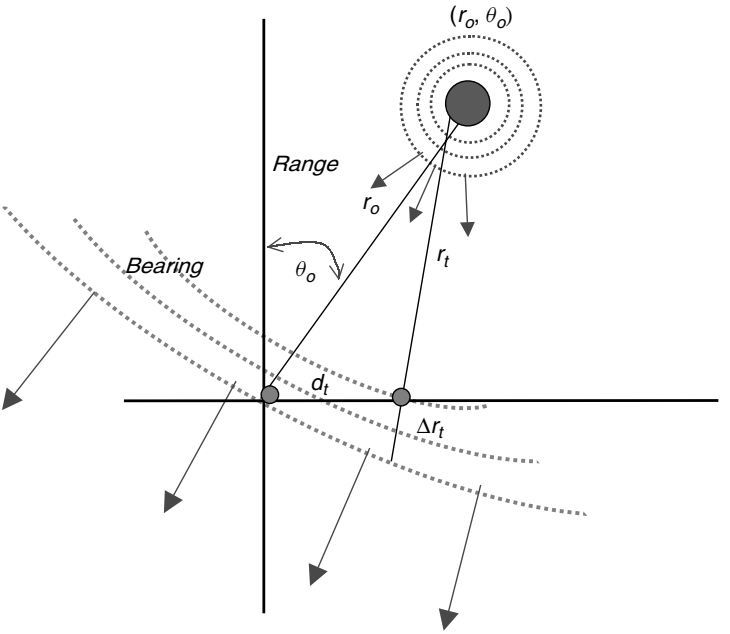



**Figure 9.31.** Geometry of spherical wavefront propagation to a two-element sensor array.

$\triangle r_\ell(t)$ is expressed in terms of the *reference range*, $r_o$, and the *dynamic range*, $r_\ell(t)$ as

$$\triangle r_\ell(t) = r_o - r_\ell(t) = r_o - r_o\sqrt{1 + \left(\frac{d_\ell(t)}{r_o}\right)^2 - 2\left(\frac{d_\ell(t)}{r_o}\right)\sin\theta_o} \tag{9.88}$$

where $d_\ell(t)$ is the *distance* between the $\ell$th-sensor and the reference $r_o$ at time $t$ and $\theta_o$ is the *angle* made with the vertical or *bearing angle* at $t = 0$. The *horizontal* array is in motion with speed, $v$; therefore the coordinate of the $\ell$th-sensor element is

$$d_\ell(t) = x_\ell + vt \tag{9.89}$$

The corresponding *dynamic time delay* is the incremental range over the propagation speed in the medium, that is,

$$\tau_\ell(t) = \frac{\triangle r_\ell(t)}{c} = \frac{r_o}{c}\left(1 - \sqrt{1 + \left(\frac{d_\ell(t)}{r_o}\right)^2 - 2\left(\frac{d_\ell(t)}{r_o}\right)\sin\theta_o}\right) \tag{9.90}$$

Therefore the narrowband, complex pressure-field measurement at the $\ell$th-sensor is simply

$$p_\ell(t) = \alpha_\ell e^{j2\pi f_o(t-\tau_\ell(t))} \tag{9.91}$$

where $\alpha_\ell$ is the amplitude, $f_o$ is the target frequency, $\theta_o$ and $r_o$, the respective initial target bearing and range.

Define the composite parameter vector, $\Theta(t) := [\alpha \mid f_o \mid \theta_o \mid r_o]'$, with the parameters assumed piecewise constant (as before). Thus we obtain the state-space relations

$$\frac{d}{dt}\begin{bmatrix} \alpha \\ f_o \\ \theta_o \\ r_o \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} \alpha \\ f_o \\ \theta_o \\ r_o \end{bmatrix} \tag{9.92}$$

with the corresponding pressure-field measurement from the $L$-element array as

$$\begin{bmatrix} p_1(t) \\ \vdots \\ p_L(t) \end{bmatrix} = \begin{bmatrix} \alpha_1 e^{j2\pi f_o(t-\tau_1(t))} \\ \vdots \\ \alpha_L e^{j2\pi f_o(t-\tau_L(t))} \end{bmatrix} \tag{9.93}$$

Assume that the parameters as well as the measurements are contaminated by additive, white gaussian noise with, $t \to t_k$, the sampled-data. Then the following set of dynamic relations can be rewritten succinctly as the discrete *Gauss-Markov wavefront curvature model* as

$$\begin{aligned} \Theta(t_k) &= \Theta(t_{k-1}) + \Delta t_k w(t_{k-1}) \\ p_\ell(t_k) &= c_\ell[t_k;\Theta] + v_\ell(t_k) = \theta_1(t_k) e^{j2\pi\theta_2(t_k)(t_k-\tau_\ell(\Theta;t_k))} + v_\ell(t_k), \\ &\qquad \ell = 1,\ldots,L \end{aligned} \tag{9.94}$$

where the time delay at the $\ell$th-sensor and time $t_k$ is given in terms of the unknown parameters of Eq. (9.92) by

$$\tau_\ell(\Theta;t_k) := \frac{1}{c}\left(\theta_4(t_k) - \sqrt{\theta_4^2(t_k) + d_\ell^2(t) - 2d_\ell(t)\theta_4(t_k)\sin\theta_3(t_k)}\right) \tag{9.95}$$

With this Gauss-Markov model in mind, the design of the model-based processor is straightforward, but we do require the following measurement jacobians to implement the *AMBP*. First, we simplify the notation by defining

$$\begin{aligned} c_\ell := c_\ell[t_k;\Theta] &= \theta_1 \mathcal{F}_\ell(\theta_2,\theta_3,\theta_4) \\ \mathcal{F}_\ell(\theta_2,\theta_3,\theta_4) &:= e^{j2\pi\theta_2(t_k)}[t_k - (\frac{1}{c}(\theta_4(t_k) - \beta_\ell(\theta_3,\theta_4;t_k))] \\ \beta_\ell(\theta_3,\theta_4;t_k) &:= \sqrt{\theta_4^2 + d_\ell^2(t_k) - 2d_\ell(t_k)\theta_4\sin\theta_3} \end{aligned} \tag{9.96}$$

From the definitions above, the Jacobians can be shown to be the following:

$$\begin{aligned}
\frac{\partial c_\ell}{\partial \theta_1} &= \mathcal{F}_\ell(\theta_2, \theta_3, \theta_4) \\
\frac{\partial c_\ell}{\partial \theta_2} &= j2\pi \left[t_k - \tau_\ell(\theta_3, \theta_4)\right] \theta_1 \mathcal{F}_\ell(\theta_2, \theta_3, \theta_4) \\
\frac{\partial c_\ell}{\partial \theta_3} &= \frac{\partial c_\ell}{\partial \tau_\ell} \times \frac{\partial \tau_\ell}{\partial \theta_3} \\
\frac{\partial c_\ell}{\partial \theta_4} &= \frac{\partial c_\ell}{\partial \tau_\ell} \times \frac{\partial \tau_\ell}{\partial \theta_4}
\end{aligned} \tag{9.97}$$

with

$$\begin{aligned}
\frac{\partial c_\ell}{\partial \tau_\ell} &= -j2\pi\theta_1\theta_2 \mathcal{F}_\ell(\theta_2, \theta_3, \theta_4) \\
\frac{\partial \tau_\ell}{\partial \theta_3} &= \frac{d_\ell(t_k)}{c} \theta_4 \cos\theta_3 \beta_\ell(\theta_3, \theta_4; t_k) \\
\frac{\partial \tau_\ell}{\partial \theta_4} &= \frac{1}{c} \left[1 - \beta_\ell(\theta_3, \theta_4; t_k)\, (\theta_4 - d_\ell(t_k) \sin\theta_3)\right]
\end{aligned}$$

Next we perform a simulation using *SSPACK_PC* [24] for the processor with initial range, frequency and bearing of 3 Km, 51.1 Hz, 27°. The *MBP* output of the corresponding *AMBP* is shown in Figure 9.32. It was tuned to provide the minimum variance estimates (zero-mean/white innovations). It is clear from the results that the processor is capable of providing a reasonable estimate of these dynamic parameters. This completes the model-based application to the space-time estimation problem.

We summarize the results of this application as follows:

| | |
|---|---|
| Criterion: | $J = \text{trace}\ \ \tilde{P}(t\|t)$ |
| Models: | |
| Signal: | $\Theta(t_k) = \Theta(t_{k-1}) + w(t_k)$ |
| Measurements: | $\mathbf{p}(t_k) = \mathbf{c}[t_k; \Theta] + \mathbf{n}(t_k)$ for $c_\ell[t_k; \Theta] = \theta_1(t_k) e^{j2\pi\theta_2(t_k)(t_k - \tau_\ell(\Theta; t_k))}$ |
| Noise: | $w \sim \mathcal{N}(0, R_{ww})$ and $n \sim \mathcal{N}(0, R_{nn})$ |
| Algorithm: | $\hat{\Theta}(t_k\|t_k) = \hat{\Theta}(t_k\|t_{k-1}) + K(t_k) e(t_k)$ |
| Quality: | $\tilde{P}(t_k\|t_k) = (I - K(t_k) C(t_k))\, \tilde{P}(t_k\|t_{k-1})$ |

This completes the application the *AMBP* for target identification and localization.

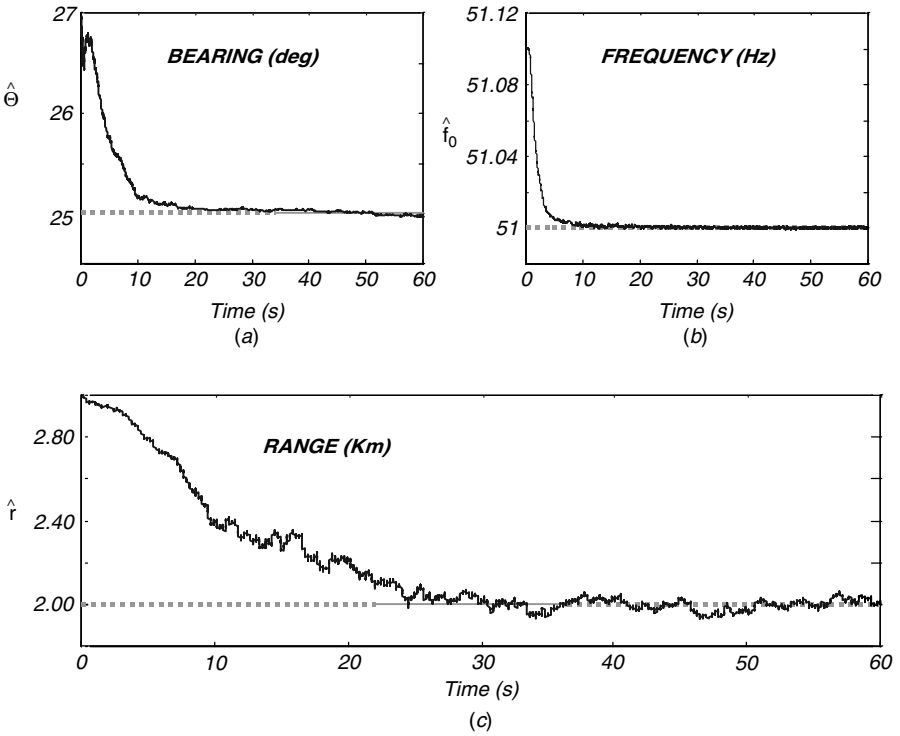



**Figure 9.32.** *AMBP* range-bearing-frequency estimation: (*a*) Bearing estimate (27°). (*b*) Frequency estimate (51.0 Hz). (*c*) Range estimate (2.0 Km).

### 9.4.4 Summary

A model-based approach to space-time acoustic array processing was presented [32]. By explicitly including the motion of the array in the signal model, improved bearing estimation and resolution performance is obtained. The technique is shown to be equivalent to a passive synthetic aperture processor which allows the motion of the array to effectively increase its useful aperture, thereby providing all of the associated improvement in performance. The advantage of this approach is that there is essentially no limit to the degree of sophistication allowed for the particular models chosen for the processor. In this work we have chosen the signal model to be a sum of plane or cylindrical waves. However, the method can easily be generalized to more sophisticated wave propagation models (e.g., normal-modes) as illustrated in the next application.

## 9.5 *MBP* FOR PASSIVE LOCALIZATION IN A SHALLOW OCEAN

In this section a model-based approach is developed to solve the passive localization problem in ocean acoustics using the state-space formulation. It is shown that

the inherent structure of the resulting processor consists of a parameter estimator coupled to a nonlinear optimization scheme. The parameter estimator is developed using the model-based approach in which an ocean acoustic propagation model is used in developing the model-based processor (*MBP*) required for localization. Recall that model-based signal processing is a well-defined methodology enabling the inclusion of environmental (propagation) models, measurement (sensor arrays) models and noise (shipping, measurement) models into a sophisticated processing algorithm. Here we design the parameter estimator or more appropriately the *adaptive* model-based processor (*AMBP*) for a normal-mode propagation model developed from a shallow water ocean experiment. After simulation we apply it to a set of experimental data demonstrating the capability of this approach and analyze its performance.

### 9.5.1 Ocean Acoustic Forward Propagator

The ocean is an extremely hostile environment compared to other media. It can be characterized by random, nonstationary, nonlinear, space-time varying parameters that must be estimated to "track" its dynamics. Not only does the basic ocean propagation medium depend directly on its changing temperature variations effecting the sound speed directly but also other forms of noise and clutter that create uncertainty and contaminate any array measurements. The need for a processor is readily apparent. However, this hostile operational environment places unusual demands on it. For instance, the processor should be capable of (1) "learning" about its own operational environment including clutter, (2) "detecting" a target with minimal target information, (3) "enhancing" the signal while removing both clutter and noise, (4) "localizing" the target position, and (5) "tracking" the target as it moves. A *MBP* is capable of satisfying these requirements, but before we motivate the processor, we must characterize the shallow ocean environment.

In this subsection we briefly develop state-space signal processing models from the corresponding ocean acoustic normal-mode solutions to the wave equation. The state-space model will eventually be employed as a "forward" propagator in a model-based signal processing schemes [18]. Note that this approach does *not* offer a new solution to the resulting boundary value problem but actually requires that solution be available a priori in order to propagate the normal-modes recursively in an initial value (marching) scheme.

For our propagation model, we assume a horizontally-stratified ocean of depth $h$ with a *known* source position $(x, y, z)$. We assume that the acoustic energy from a point source propagating over a long range, $r$, $(r >> h)$ toward a receiver can be modeled as a trapped wave characterized by a waveguide phenomenon. For a layered waveguide model with sources on the $z$-axis (or vertical), the pressure field, $p$, is symmetric about $z$ (with known source bearing) and is therefore governed by the cylindrical wave equation, which is given by [33]

$$\nabla_r^2 p(r,z,t) + \frac{1}{r}\nabla_r p(r,z,t) + \nabla_z^2 p(r,z,t) = \frac{1}{c^2}\nabla_t^2 p(r,z,t) \tag{9.98}$$

The solution to this equation is accomplished by using the separation of variables technique, that is,

$$p(r, z, t) = \mu(r)\phi(z)T(t) \tag{9.99}$$

Substituting Eq. (9.99) into Eq. (9.98), assuming a harmonic source,

$$T(t) = e^{jwt} \tag{9.100}$$

and defining separation constants $\kappa_z, \kappa_r$, we obtain the following set of ordinary differential equations

$$\begin{aligned} \frac{d^2}{dr^2}\mu(r) + \frac{1}{r}\frac{d}{dr}\mu(r) &= -\kappa_r^2\mu(r) \\ \frac{d^2}{dz^2}\phi(z) &= -\kappa_z^2\phi(z) \\ \kappa^2 &= \frac{\omega^2}{c^2(z)} \\ \kappa^2 &= \kappa_r^2 + \kappa_z^2 \end{aligned} \tag{9.101}$$

where solutions to each of these relations describe the propagation of the acoustic pressure in cylindrical coordinates assuming a harmonic source with the speed of sound a function of depth, $c = c(z)$.

The depth relation of Eq. (9.101) is an eigenvalue equation in $z$ with

$$\frac{d^2}{dz^2}\phi_m(z) + \kappa_z(m)\phi_m(z) = 0, \qquad m = 1, \ldots, M \tag{9.102}$$

whose eigensolutions $\{\phi_m(z)\}$ are the *modal functions* and $\kappa_z$ is the wavenumber in the $z$-direction. These solutions depend on the sound speed profile, $c(z)$, and the boundary conditions at the surface and bottom.

Using the orthogonality property of the modal functions with a known point source located at $z_s$, we obtain the total wavenumber in the corresponding *dispersion relation*

$$\kappa^2 = \frac{\omega^2}{c^2(z)} = \kappa_r^2(m) + \kappa_z^2(m), \qquad m = 1, \ldots, M \tag{9.103}$$

where $\kappa_r, \kappa_z$ are the respective wave numbers in the $r$ and $z$ directions with $c$ the depth-dependent sound speed profile and $\omega$ the harmonic source frequency.

For our purpose we are concerned with the estimation of the pressure field. Therefore we remove the time dependence, normalize units, and obtain the *acoustic pressure propagation model*,

$$p(r_s, z) = q\sum_{m=1}^{M} H_0(\kappa_r(m)r_s)\phi_m(z_s)\phi_m(z) \tag{9.104}$$

where $p$ is the acoustic pressure, $q$ is the source amplitude, $\phi_m$ is the $m$th modal function at $z$ and $z_s$, $\kappa_r(m)$ is the horizontal wavenumber associated with the $m$th mode, $r_s$ is the source range, and $H_0(\kappa_r(m)r_s)$ is the zeroth-order Hankel function (range solution).

To develop the state-space "forward" propagation model, we use the cylindrical wave equation and make the appropriate assumptions leading to the normal-mode solutions of Eq. (9.102). Since the depth relation is a linear, space-varying coefficient (for each layer) differential equation, it can easily be placed in state-space form where the state-vector is defined as $\underline{x}_m := [\phi_m(z)\frac{d}{dz}\phi_m(z)]' = [\phi_{m1}(z)\phi_{m2}(z)]'$. Examining mode propagation in more detail, we see that each mode is characterized by a set of second-order, ordinary differential equations, which can be written

$$\frac{d}{dz}\underline{x}_m(z) = A_m(z)\underline{x}_m(z) = \begin{bmatrix} 0 & 1 \\ -\kappa_z^2(m) & 0 \end{bmatrix}\underline{x}_m(z) \tag{9.105}$$

The solution to this equation is governed by the state-transition matrix, $\Phi_m(z, z_0)$, where the state equation is solved by

$$\underline{x}_m(z) = \Phi_m(z, z_0)\underline{x}_m(z_0), \qquad m = 1, \ldots, M \tag{9.106}$$

and the transition matrix satisfies

$$\frac{d}{dz}\Phi_m(z, z_0) = A_m(z)\Phi_m(z, z_0), \qquad m = 1, \ldots, M \tag{9.107}$$

with $\Phi_m(z_0, z_0) = I$.

If we include all of the $M$-modes in the model, then we obtain

$$\frac{d}{dz}\underline{x}(z) = \begin{bmatrix} A_1(z) & \cdots & O \\ \vdots & & \vdots \\ O & \cdots & A_M(z) \end{bmatrix}\underline{x}(z) \tag{9.108}$$

or simply

$$\frac{d}{dz}\underline{x}(z) = A(z)\underline{x}(z) \tag{9.109}$$

This approach leads to a Gauss-Markov representation which includes the second order statistics. The measurement noise can represent the near-field acoustic noise field, flow noise on the hydrophone and electronic noise. The modal (process) noise can represent sound speed errors, distant shipping noise, errors in the boundary conditions, sea state effects and ocean inhomogeneities. By using the Hankel function $H_0(\kappa_r(m)r_s)$ for range, we reduce the state-space model to that

of "depth only" and the Gauss-Markov representation for this model is given with state vector[5]

$$\frac{d}{dz}\underline{\phi}(z) = A(z)\underline{\phi}(z) + \mathbf{w}_{\phi}(z) \tag{9.110}$$

where $A(z) := \text{diag}[A_1(z) \cdots A_M(z)]$ and

$$A_m(z) = \begin{bmatrix} 0 & 1 \\ -\kappa_z^2(m) & 0 \end{bmatrix}, \qquad m = 1, \ldots, M \tag{9.111}$$

Since our array spatially samples the pressure field at each sensor location $z_\ell$, the corresponding measurement model is given by

$$p(r_s, z_\ell) = C^T(r_s, z)\underline{\phi}(z_\ell) + v(z_\ell) \tag{9.112}$$

where

$$C^T(r_s, z) = [\beta_1(r_s, z_s) \quad 0|\beta_2(r_s, z_s) \quad 0| \cdots |\beta_M(r_s, z_s) \quad 0] \tag{9.113}$$

with $\beta_m(r_s, z_s) = q\phi_m(z_s)/\int_0^h \phi_m^2(z)dz[H_0(k_r(m)r_s)]$. The random noise vector $\mathbf{w}_\phi$ and $\mathbf{v}$ are assumed gaussian, zero-mean with respective covariance matrices, $R_{w_\phi w_\phi}$ and $R_{vv}$. Note that due to the recursive nature of the processor that *each* sensor is processed *sequentially* along the array. This is a huge computational savings when processing long element arrays [36]. The localization problem solution evolves from the measurement equation of the depth only Gauss-Markov model, where we can write the sampled pressure field in terms of range-depth dependent terms as

$$p(r_s, z_\ell) = \sum_{m=1}^{M} \beta_m(r_s, z_s)\phi_{m1}(z_\ell) + v(z_\ell) \tag{9.114}$$

For the two-dimensional localization problem, we can decompose the pressure measurement further as

$$p(r_s, z_\ell) = \sum_{m=1}^{M} \gamma_m(r, z)\theta_m(r_s, z_s)\phi_{m1}(z_\ell) + v(z_\ell) \tag{9.115}$$

where $\gamma_m$ represents the *known* parametric functions and $\theta_m(r_s, z_s)$ the *unknown* functions of position. Equating these functions with the modal coefficients, $\beta_m$, we have that

$$\gamma_m(r, z) := \frac{q}{\int_0^h \phi_m^2(z)dz} \tag{9.116}$$

[5]Here we have modified the state vector notation from $\underline{x}_m \to \underline{\phi}_m \to \phi_m = \phi_{m1}(z)$ and $\phi_{m2} = \frac{d}{dz}\phi_m(z)$, $\underline{\phi}(z) \to [\phi_{m1}(z) \quad \phi_{m2}(z)]'$.

and

$$\theta_m(r_s, z_s) := H_o(k_r(m)r_s)\phi_{m1}(z_s) \tag{9.117}$$

an implicit, separable function of $r_s$ and $z_s$ which we define as the source *range-depth function*. With these definitions in mind, it is now possible to define the *model-based localization problem* as follows:

> **GIVEN** a set of noisy pressure-field and sound speed measurements, $[\{p(r_s, z_\ell)\}, \{c(z_\ell)\}]$ along with the normal-mode propagation model and parameters. **FIND** the "best" (minimum error variance) estimate of the source position $(r_s, z_s)$; that is, find $\hat{r}_s$ and $\hat{z}_s$.

In order to solve this problem, we first estimate the "unknown" range-depth function, $\theta_m(r_s, z_s)$, from the noisy pressure-field measurement model and then use numerical optimization techniques to perform the localization $(r_s, z_s)$. We discuss the processor used to perform the required parameter estimation in the next subsection.

### 9.5.2 *AMBP* for Localization

In this subsection a parametrically adaptive, model-based approach (see Chapter 8) is developed to solve the passive localization problem in ocean acoustics using the state-space formulation. It is shown that the inherent structure of the resulting processor consists of a parameter estimator coupled to a nonlinear optimization scheme. First, let us examine the structure of the adaptive model-based localizer shown in Figure 9.33. Here we see that it consists of two distinct parts: a parameter estimator implemented using a *AMBP* and a nonlinear optimizer to estimate the source position. We see that the primary purpose of the parameter estimator is to provide estimates of the inherent localization functions that then must be solved (implicitly) for the desired position.

We develop the model-based localization solution to our ocean acoustic problem and show how it is realized by utilizing a *AMBP* coupled to a nonlinear optimizer. It will also be shown that the *AMBP* provides an enhanced estimate of the required range-depth function, which is essentially the scaled modal coefficients that are

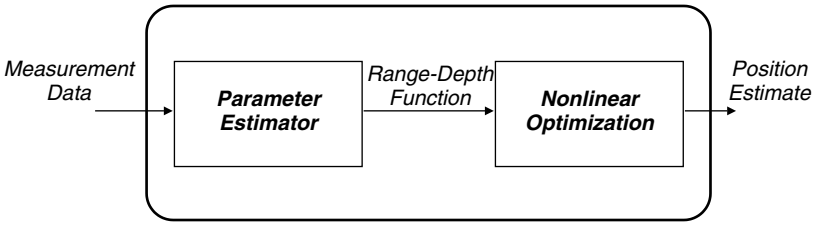



**Figure 9.33.** Model-based localization: The basic processor.

supplied to the optimal position estimator. The localizer is then applied to experimental data demonstrating the impact of using the *AMBP* for signal enhancement prior to localization. We start by motivating the need for the *AMBP* from the optimization viewpoint and then proceed with its actual development.

*Range-Depth Optimizer* To motivate the development of this processor, we must first investigate the structure of the optimizer for localization and then investigate just how the *AMBP* is to be used. In the design of a localizer, we choose a nonlinear least squares approach. Thus the optimization problem is to find the source position, $(r_s, z_s)$, that minimizes

$$J(r_s, z_s) := \frac{1}{M} \sum_{m=1}^{M} (\theta_m(r_s, z_s) - H_o(\kappa_r(m) r_s)\, \phi_{m1}(z_s))^2 \tag{9.118}$$

From this cost function $J(r_s, z_s)$, we see that we must have an estimate of the range-depth function, $\theta_m(r_s, z_s)$, and this is provided by our *AMBP*. However, we must also have estimates of the associated Hankel function, $H_0(\kappa_r(m) r_n)$ at search range, $r_n$, and the corresponding modal functions evaluated at the current search iterate depth, $z_n$ as $\phi_{m1}(z_n)$. The *AMBP* provides us with estimates of these modal functions $\{\hat{\phi}_{m1}(z_\ell)\}$; $m = 1, \ldots, M$; $\ell = 1, \ldots, L$ at each sensor location (in depth). Since the optimizer requires a finer mesh (in depth) than the modal function estimates at each sensor to perform its search, we use the state-space propagator to generate the estimates at a finer sampling interval in depth, that is,

$$\Delta z_n := \frac{\Delta z_\ell}{N} \qquad \text{for } N \in \mathcal{I} \tag{9.119}$$

Thus, for a given value of "search" depth $z_n$, we find the closest available depths from the estimator (array geometry) to bracket the required target search depth, $z_{\ell-1} < z_n < z_\ell$, and use the lower bound $z_{\ell-1}$ to select the initial condition vector for our propagator. We then propagate the modal function at the finer $\Delta z_n$ to obtain the desired estimate at $\hat{\phi}_{m1}(z_n)$. Note that the propagator evolves simply by discretizing the differential equation using first differences

$$\frac{d}{dz}\phi(z) \approx \frac{\phi(z_n) - \phi(z_{n-1})}{\Delta z_n} \tag{9.120}$$

which leads to the corresponding *state-space propagator* given by

$$\underline{\hat{\phi}}(z_n) = [\mathbf{I} + \Delta z_n A(z_n)]\, \underline{\hat{\phi}}(z_{n-1}) \quad \text{for} \quad \underline{\hat{\phi}}(z_{n-1}) = \underline{\hat{\phi}}(z_{\ell-1}) \tag{9.121}$$

In this way the state-space propagator is used to provide functional estimates to the nonlinear optimizer for localization. Thus, we see that the *AMBP* (next section) is designed to not only provide estimates of the range-depth function but

also provide enhanced estimates of the modal functions at each required search depth iteration, that is,

$$\left[\{\hat{\theta}_m(r_s, z_s)\}, \{\hat{\phi}_{m1}(z_\ell)\}\right] \longrightarrow \left[\{\hat{\phi}_m(z_n)\}, (\hat{r}_s, \hat{z}_s)\right] \tag{9.122}$$

From an estimation viewpoint, it is important to realize the ramifications of the output of the processor and its relationship to the position estimates. The respective range-depth and modal estimates $\hat{\theta}(z)$ and $\hat{\phi}(z)$ provided by the *AMBP* are minimum variance estimates (approximately). In the case of gaussian noise they are the maximum likelihood (maximum a posteriori) estimates. Therefore the corresponding maximum likelihood *invariance theorem* guarantees that the solutions for the $(r_s, z_s)$ are also the maximum likelihood estimates of position. This completes the description of the localizer. Next we discuss how the range depth and modal functions are estimated from noisy pressure-field measurements by developing the *AMBP*.

*Parametrically Adaptive MBP* In this subsection a parametrically adaptive, model-based approach is developed to solve the passive localization problem in ocean acoustics using the state-space formulation. We design the parameter estimator or more appropriately the *AMBP* for a propagation model developed from the Hudson Canyon shallow water ocean experiment [34].

Let us examine the detailed inherent structure of the adaptive model-based localizer shown in Figure 9.34. Here we see that it consists of two distinct parts: a parameter estimator implemented using a *AMBP* as discussed above and a nonlinear optimizer to estimate the source position. We see that the primary purpose of the parameter estimator is to provide estimates of the inherent localization functions that then must be solved (implicitly) for the desired position. Thus we see that it

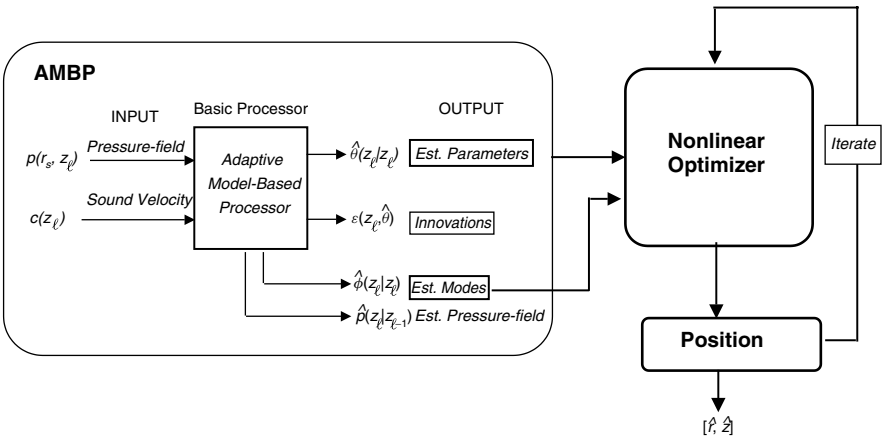



**Figure 9.34.** Detailed model-based localization processor structure: (*a*) *AMBP*. (*b*) Optimizer.

is the *AMBP* that provides the heart of the model-based localization scheme by enabling an enhanced estimate of the required range-depth functions (scaled modal coefficients) that are supplied to the optimal position estimator. Next we briefly outline the processor, show the structure of the embedded *AMBP* and apply it to real hydrophone data demonstrating its performance.

It is clear from the localization discussion in the previous section that we must estimate the vector source range-depth function, $\theta(r_s, z_s)$, directly from the measured data as well as the required modal functions. The basic approach we take is to first realize that at a given source depth the implicit range-depth function is *fixed*. Therefore we can assume that $\theta(r_s, z_s)$ is a constant ($\dot{\theta}(z) = 0$) or a random walk with a *discrete* Gauss-Markov representation given by

$$\theta(r_s, z_\ell) = \theta(r_s, z_{\ell-1}) + \Delta z_\ell w_\theta(z_{\ell-1}) \tag{9.123}$$

The underlying model for our ocean acoustic problem becomes the normal mode propagation model (in discrete form) using first differences with an augmented parameter space (ignoring the noise sources):

$$\begin{aligned}
\phi_{m1}(z_\ell) &= \phi_{m1}(z_{\ell-1}) + \Delta z_\ell \phi_{m2}(z_{\ell-1}) \\
\phi_{m2}(z_\ell) &= -\Delta z_\ell \kappa_z^2(m)\phi_{m1}(z_{\ell-1}) + \phi_{m2}(z_{\ell-1}), \qquad m = 1, \ldots, M \\
\theta_1(r_s, z_\ell) &= \theta_1(r_s, z_{\ell-1}) \\
&\vdots \qquad\qquad \vdots \\
\theta_M(r_s, z_\ell) &= \theta_M(r_s, z_{\ell-1})
\end{aligned} \tag{9.124}$$

with the corresponding *scalar* measurement model given by

$$p(r_s, z_\ell) = \sum_{m=1}^{M} \gamma_m(r, z_\ell)\theta_m(r_s, z_\ell)\phi_{m1}(z_\ell) + v(z_\ell) \tag{9.125}$$

or equivalently in vector-matrix form as

$$\begin{bmatrix} \underline{\phi}(z_\ell) \\ - - - \\ \underline{\theta}(z_\ell) \end{bmatrix} = \begin{bmatrix} A(z_{\ell-1}, \theta) & | & 0 \\ - & - & - \\ 0 & | & I_M \end{bmatrix} \begin{bmatrix} \underline{\phi}(z_{\ell-1}) \\ - - - \\ \underline{\theta}(z_{\ell-1}) \end{bmatrix} + \begin{bmatrix} \underline{w}(z_{\ell-1}) \\ - - - \\ \underline{w}_\theta(z_{\ell-1}) \end{bmatrix}$$

and

$$p(r_s, z_\ell) = \begin{bmatrix} \underline{C}'(r_s, z_\ell) & | & \underline{0} \end{bmatrix} \begin{bmatrix} \underline{\phi}(z_\ell) \\ - - - - \\ \underline{\theta}(z_\ell) \end{bmatrix} + v(z_\ell) \tag{9.126}$$

We choose this general representation where the set $\{\theta_m(r_s, z_\ell)\}; m = 1, \ldots, M$ is the *unknown* implicit function of range and depth, while the parameters,

$\{\gamma_m(r, z_\ell)\}$, $\kappa_r$ and $c(z)$ represent the *known* a-priori information that is included in the processor. The basic *prediction* estimates for the $m$th mode are

$$\begin{aligned}
\hat{\phi}_{m1}(z_\ell|z_{\ell-1}) &= \hat{\phi}_{m1}(z_{\ell-1}|z_{\ell-1}) + \triangle z_\ell \hat{\phi}_{m2}(z_{\ell-1}|z_{\ell-1}) \\
\hat{\phi}_{m2}(z_\ell|z_{\ell-1}) &= -\triangle z_\ell \kappa_z^2(m)\hat{\phi}_{m1}(z_{\ell-1}|z_{\ell-1}) + \hat{\phi}_{m2}(z_{\ell-1}|z_{\ell-1}), \qquad m = 1, \ldots, M \\
\hat{\theta}_1(r_s, z_\ell|z_{\ell-1}) &= \hat{\theta}_1(r_s, z_{\ell-1}|z_{\ell-1}) \\
&\vdots \qquad\qquad\qquad \vdots \\
\hat{\theta}_M(r_s, z_\ell|z_{\ell-1}) &= \hat{\theta}_M(r_s, z_{\ell-1}|z_{\ell-1})
\end{aligned} \tag{9.127}$$

The corresponding innovations (parameterized by $\theta$) are given by

$$\epsilon(z_\ell; \theta) = p(r_s, z_\ell) - \sum_{m=1}^{M} \gamma_m(r, z_s)\hat{\theta}_m(r_s, z_\ell|z_{\ell-1})\hat{\phi}_{m1}(z_\ell|z_{\ell-1}; \theta) \tag{9.128}$$

with the vector *correction equations* (see Chapter 8 for *AMBP* details):

$$\begin{aligned}
\hat{\phi}(z_\ell|z_\ell) &= \hat{\phi}(z_\ell|z_{\ell-1}) + K_\phi(z_\ell)\epsilon(z_\ell; \theta) \\
\hat{\theta}(r_s, z_\ell|z_\ell) &= \hat{\theta}(r_s, z_\ell|z_{\ell-1}) + K_\theta(z_\ell)\epsilon(z_\ell; \theta)
\end{aligned} \tag{9.129}$$

So we see (simply) how the unknown range-depth parameters are augmented into the *AMBP* algorithm to enhance the required signals and extract the desired parameters. Recall the detailed structure of the model-based localizer incorporating the *AMBP* in Figure 9.34.

### 9.5.3 *AMBP* Application to Experimental Data

We use the Hudson Canyon experimental data to analyze the localizer performance [34]. A 23-element vertical array was deployed from the bottom with 2.5 m separation to measure the pressure field. Through spectral analysis the following average horizontal wave numbers $\{0.208, \ 0.199, \ 0.183, \ 0.175, \ 0.142\}$ $\mathrm{m}^{-1}$ for the five modes supporting the water column from a 36 m deep, 50 Hz source at 0.5 Km range resulted. Using *SSPACK_PC* [24], a Toolbox available in *MATLAB* [25], we investigate the design using the experimental hydrophone measurements from the Hudson Canyon [34]. Here we initialize the *AMBP* with the average set of horizontal wave numbers. The resulting estimates are quite reasonable as shown in Figure 9.35. The results are better than those reported previously for this data set (see [35]) primarily because we have allowed the processor to dynamically adapt (parameter estimator) to the changing parameters. The results for the higher order modes follow those predicted by the model as observed in the figure and corresponding estimation errors. The reconstructed pressure field and innovations are also quite reasonable as shown in Figure 9.36 and indicate a "tuned" processor with its zero-mean ($1.0 \times 10^{-3} < 2.6 \times 10^{-3}$) and white innovations ($\sim 8.3\%$ out

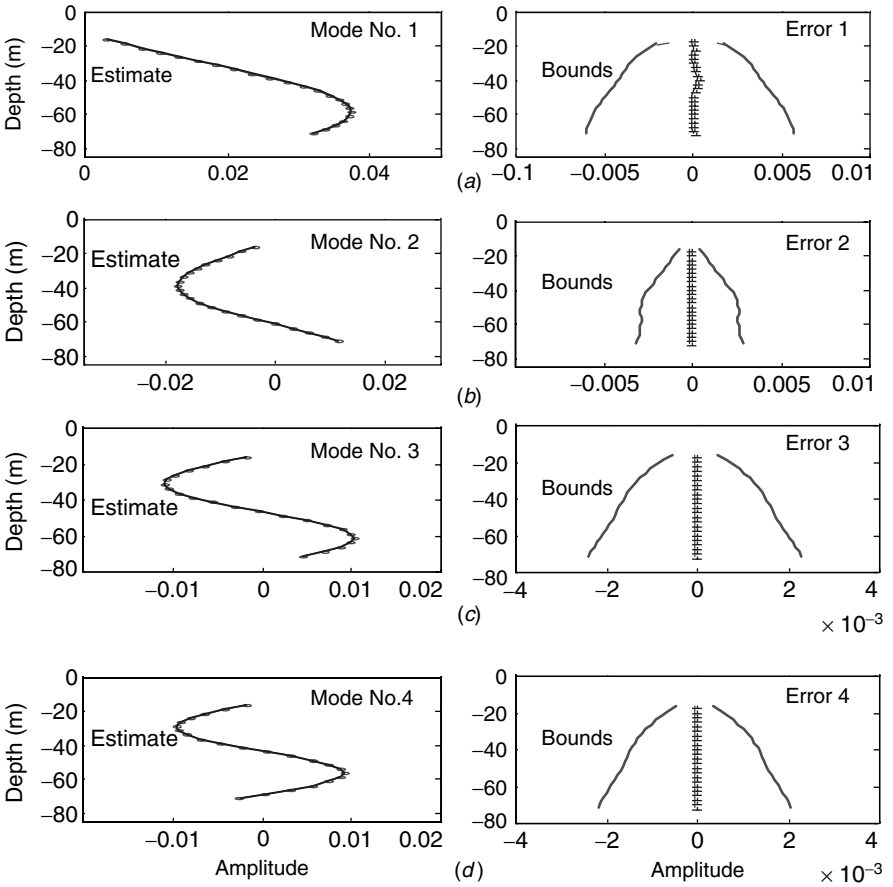



**Figure 9.35.** Adaptive model-based processing of Hudson Canyon experiment (0.5 Km): (*a*) Mode 1 and error (0% out). (*b*) Mode 2 and error (0% out). (*c*) Mode 3 and error (0% out). (*d*) Mode 4 and error (0% out).

& $WSSR < \tau$ ). The final parameter estimates with predicted error statistics for this data are also included in Table 9.3 for comparison to the simulated. We see again that the *AMBP* appears to perform better than the *MBP* simply because the range-depth parameters (scaled modal coefficients) are "adaptively" estimated, on-line, providing a superior fit to the raw data as long as we have reasonable estimates to initialize the processor. The high uncertainties in the estimates are crude because we only use a single pass over the array. Ensemble statistics should be used to get a better feel for performance. This completes the discussion on the design of the *AMBP*, we take these intermediate results and apply the nonlinear optimizer of the previous section to obtain a solution to the localization problem.

Next we consider the application of the *AMBP* to the experimental Hudson Canyon data set. Here we use the *AMBP* to provide estimates of $[\{\hat{\theta}_m(r_s, z_s)\}$,

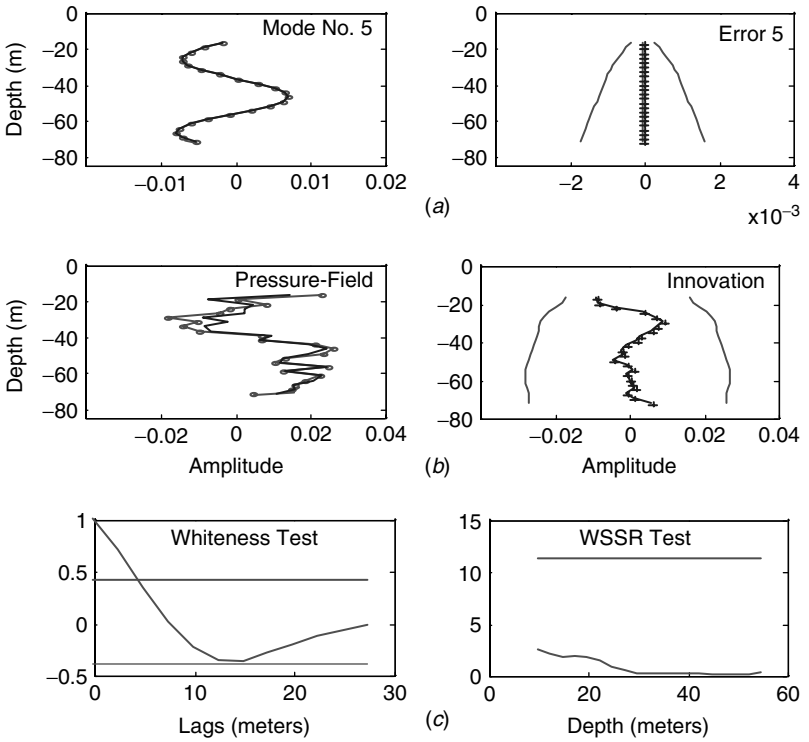



**Figure 9.36.** Adaptive model-based processing of Hudson Canyon experiment (0.5 Km): (*a*) Mode 5 and error (0% out). (*b*) Pressure field and innovation (0%–0% out). (*c*) Whiteness test (8.3% out) and *WSSR*.

**Table 9.3. *AMBP:* Range-Depth Parameter Estimation**

| | *Hudson Canyon Experiment* | | |
|---|---|---|---|
| Wave Numbers | Model Prediction | Simulation Est/Err | Experiment Est/Err |
| $\theta_1$ | 1.000 | $1.014 \pm 0.235$ | $1.015 \pm 0.362$ |
| $\theta_2$ | 0.673 | $0.701 \pm 0.238$ | $0.680 \pm 0.364$ |
| $\theta_3$ | 0.163 | $0.127 \pm 0.430$ | $0.163 \pm 0.393$ |
| $\theta_4$ | 0.166 | $0.138 \pm 0.476$ | $0.166 \pm 0.393$ |
| $\theta_5$ | 0.115 | $0.141 \pm 0.463$ | $0.116 \pm 0.364$ |

$\{\hat{\phi}_{m1}(z_\ell)\}]$ and then use a polytope search algorithm along with the state-space propagator to provide the localization discussed previously [37], [38], [39]. We applied the optimizer to the resulting range-depth parameters estimated by the *AMBP*. The results of the localization are shown in Figure 9.37. Here we see the range-depth parameter estimates from the *AMBP*, the true values from the

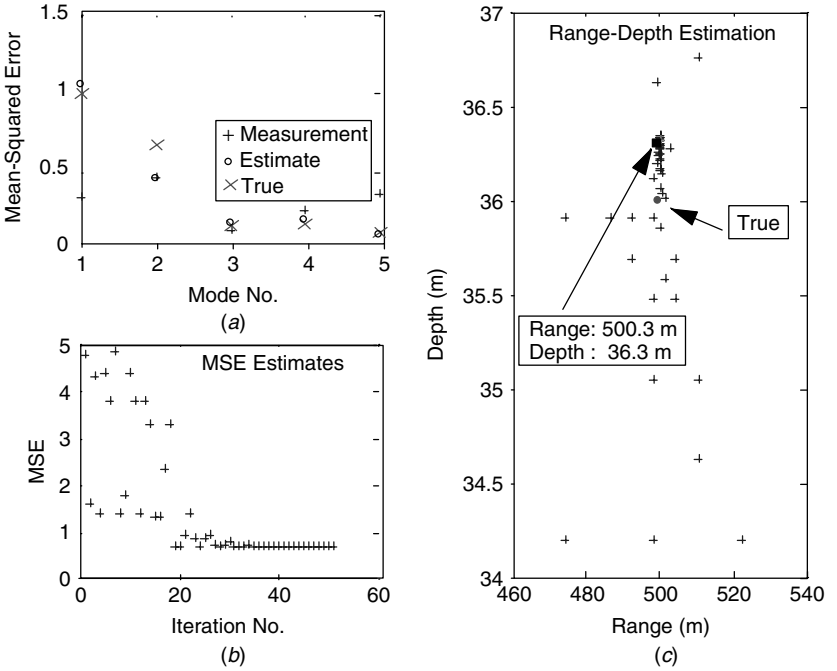



**Figure 9.37.** Hudson Canyon experiment localization: (*a*) Estimated range-depth parameters. (*b*) Mean-squared error. (*c*) True and estimated localization (500.3 m, 36.3 m).

simulator and the estimates developed by the optimizer given respectively by the **+**, **x**, **o**, characters on the plots. The corresponding mean-squared errors are also shown indicating the convergence of the optimizer after about 30 iterations as well as the actual range-depth search (position iterates) with the *true* (500 m, 36 m) and *estimated* (500.3 m, 36.3 m) position estimates shown. The algorithm appears to converge quite readily demonstrating that the model-based localization is able to perform quite well over this data set.

To understand why we can achieve this quality of localization performance, we observe that the effect of the *AMBP* is to enhance these noisy measurements enabling the optimizer to converge rapidly to the correct position. The parametrically adaptive *AMBP* enhancement capability is clear from Figures 9.35 and 9.36. The effect of the *AMBP* leads very closely to the true depth function estimate, since the modal function estimates are quite good. Thus we can think of the *AMBP* as providing the necessary enhancements in *SNR* as well as decreasing the dimensionality of the search space to that of the modal space by using the estimated range-depth functions from the *AMBP* as the "raw" measurement data input (along with the modal estimates) to the polytope optimizer [38].

We summarize the results of this application as follows:

Criterion: $J = \text{trace}\ \ \tilde{P}(z_\ell|z_\ell)$
Models:
Signal:

$$\frac{d}{dz}\phi_m(z, z_0) = A_m(z)\phi_m(z, z_0) + w_m(z), \qquad m = 1, \ldots, M$$

$$\frac{d}{dz}\theta_m(z) = w_m(z), \qquad m = 1, \ldots, N_\theta$$

Measurements: $p(r_s, z_\ell) = \sum_{m=1}^{M} \gamma_m(r, z)\theta_m(r_s, z_s)\phi_{m1}(z_\ell) + v(z_\ell)$
Noise: $w \sim \mathcal{N}(0, R_{ww})$ and $v \sim \mathcal{N}(0, R_{vv})$
Algorithm: $\hat{x}(z_\ell|z_\ell) = \hat{x}(z_\ell|z_{\ell-1}) + K(z_\ell)\epsilon(z_\ell)$
Quality: $\tilde{P}(z_\ell|z_\ell) = (I - K(z_\ell)C(z_\ell))\tilde{P}(z_\ell|z_{\ell-1})$

This completes the application the *AMBP* for ocean acoustic localization.

### 9.5.4 Summary

In this section we developed an online, parametrically adaptive, model-based solution to the ocean acoustic localization problem, that is, a target position localization estimation scheme based on coupling the normal-mode propagation model to a functional model of position. The algorithm employed was the parametrically adaptive model-based processor (*AMBP*) of Chapter 8 in predictor/corrector form coupled to a direct search optimizer using the polytope approach. We showed that the model-based localization evolves quite naturally from the *AMBP*. The results of applying the localization scheme to the raw experimental data from the Hudson Canyon experiment were quite good (see [40] for more details).

## 9.6 *MBP* FOR DISPERSIVE WAVES

Wave propagation through various media represents a significant problem in many applications in acoustics and electromagnetics especially when the medium is dispersive. We pose a general dispersive wave propagation model that could easily represent many classes of dispersive waves and proceed to develop a model-based processor employing this underlying structure. The general solution to the model-based dispersive wave estimation problem is developed using the Bayesian maximum a posteriori (*MAP*) approach which leads to the nonlinear extended model-based processor (*XMBP*) or equivalently the extended Kalman filter of Chapter 6.

### 9.6.1 Background

Dispersive wave propagation through various media is a significant problem in many applications ranging from radar target identification where electromagnetic waves propagate through the atmosphere to discern the nature of a reflected pulse classifying an intruder as friend or foe, or in submarine detection and localization where the propagation of acoustic waves through the ever-changing dispersive ocean medium causes great concern when trying to detect the presence of a hostile target and track its movements. The propagation of dispersive sonic waves in boreholes is used in exploration seismology to decide whether to construct a production oil well, while the detection of dispersive seismic waves is used to locate the epicenter of an earthquake. From the scientific viewpoint many wave-type phenomena must propagate through a hostile, noisy environment that changes rapidly in short periods of time causing great distortions in signal content that can lead to false estimates and conclusions. The characteristic common in each of these applications is the frequency dependence of the wave speed. The dispersive nature of a wave system may result from either an intrinsic frequency dependence of bulk properties of the medium or from the geometrical or mechanical properties of the system such as ocean surface waves, internal gravity waves and wave propagation in a waveguide. Therefore there is a need to develop generic characterizations of dispersive waves which do not depend on the details of the physical system primarily because the required details such as governing equations for the system and their solutions may be imperfectly known. In this section we will show that a model-based signal processing scheme applicable to *any* dispersive wave system can be developed from the basic properties of wave propagation in a dispersive medium.

Our approach will be to develop a state-space description of a dispersive wave measured by a sensor or array of sensors. For simplicity we restrict ourselves to one-dimensional waves, but the generalization to higher dimensions is straightforward. The wave pulse is assumed to be generated by an impulsive source at a known position and a known time. We consider the source pulse as a superposition of wave components of many frequencies. Since the system is dispersive, each component propagates at a different speed resulting in a spreading or dispersing of the pulse over space and time as it propagates. This spreading is described by the dispersion relation of the system which relates the frequency of each component to its wave number. We will show that a complete state-space representation of the wave can be formulated from the dispersion relation combined with an envelope or amplitude modulation function. The dispersion relation completely describes the propagation properties of the dispersive system, while the envelope is related to the initial conditions. Once specified, it is then possible to develop a generic model-based processing scheme for dispersive waves. The primary motivation for this approach follows the dispersive wave characterization developed in the text by Whitham [41]. This processor evolves directly from the modified plane wave, internal wave techniques developed using an approximation of the dispersion relation [42]. In contrast, the generic dispersive approach relies exclusively on the underlying envelope and dispersion relation to develop an optimal Bayesian processor.

It is this model-based approach [8] that we employ using a dynamic propagation model incorporated into an optimal estimation scheme to provide the necessary internal wave enhancement.

First, we present the general dispersive wave representation and outline the corresponding model-based processor (*MBP*) using the Bayesian maximum a posteriori (*MAP*) approach for the underlying wave estimation or equivalent signal enhancement problem. Next, employing this general solution, we apply it to the problem of internal wave estimation directly from an empirical dispersion relation, analyzing its performance on simulated (truth model) yet quite realistic internal ocean wave data.

### 9.6.2 Dispersive State-Space Propagator

In this subsection we develop the underlying dispersive wave model and cast it into state-space form. Once this is accomplished, a Bayesian maximum a posteriori solution is outlined and the resulting processor is shown to lead to the *XMBP* of Chapter 6. The complete model-based solution is then specified in terms of the model and algorithm showing that intuitively the dispersion relation is the fundamental quantity that must be known a priori.

First, we develop the state-space representation of a general dispersive wave system obtained from a simple physical characterization of a dispersive wave measured by a sensor or array of sensors. In contrast to the usual approach of classifying nondispersive waves in terms of their inherent differential equations (hyperbolic, elliptic, etc.), we use a solution rather than propagation equation for our dispersive prototype. Therefore, following Whitham [41], we define a generic *dispersive wave* as any system that admits solutions of the general form

$$u(x,t) = \alpha(x,t)\sin[\theta(x,t)] \tag{9.130}$$

where $u$ is the measured field and $\alpha(x,t)$, $\theta(x,t)$ are the respective envelope or amplitude modulation and phase functions. The phase is assumed to be monotonic in $x$ and $t$, and the envelope is assumed to be slowly varying compared to the phase. The phase function describes the oscillatory character of a wave, while the slowly varying envelope allows modulation of the wave without destroying its wave-like character. The local values of *wavenumber* and *frequency* can be defined as

$$\kappa(x,t) := \frac{\partial\theta}{\partial x}, \qquad \omega(x,t) := -\frac{\partial\theta}{\partial t} \tag{9.131}$$

These functions are also assumed to be slowly varying and describe the frequency modulation of a dispersive wave train. By slowly varying, we mean that we can approximate the *phase function* by

$$\theta(x,t) \approx \kappa(x,t)x - \omega(x,t)t \tag{9.132}$$

The combination of Eqs. (9.130) and (9.132) can be considered an asymptotic solution to some dispersive wave system. To complete the specification of a dispersive

wave system, we define the *dispersion relation*, $\omega := \omega(\kappa, x, t)$. This is generally an algebraic function of $\kappa(x, t)$, but can also depend on $x$ and $t$ separately to represent time-varying, nonuniform wave systems. Here we will write $\omega = \omega(\kappa)$ where the $x$ and $t$ dependence is through the wavenumber function, $\kappa(x, t)$, and any system nonuniformity is implied. This and the envelope are the only parts of the description that are unique to the particular type of wave system under investigation. The choice of *dispersion relation* enables the differentiation between acoustic radiation, electromagnetic radiation, ocean surface waves, internal gravity waves, or any other wave type. Thus, the dispersion relation is equivalent to the governing equations for a particular wave system [41]. Our only restriction is that it is independent of the envelope, $\alpha(x, t)$, restricting the formulation to *linear dispersive waves*.

From Eqs. (9.131) and (9.132) it can be shown that the phase fronts of any wave travel at the *phase speed* defined by

$$c_p(\kappa) := \frac{\omega}{\kappa} \tag{9.133}$$

while the points of constant wave number $\kappa$ travel at the *group velocity* defined by

$$c_g(\kappa) := \frac{\partial \omega}{\partial \kappa} \tag{9.134}$$

These two speeds are not the same in general and are functions of wavenumber $\kappa$. The group velocity has the additional significance of being the *energy* propagation speed for the wave system, that is, the energy in the wave packet is carried at this velocity. As such, it plays a central role in the state-space formulation of a general dispersive wave system.

Now consider the problem where an impulse occurs at time $t = 0$ at the spatial origin, $x = 0$. The impulse can be represented by the superposition of wave components with various wavenumbers. A wave train is generated by the impulse and propagates always from the origin. Each wavenumber component in the train propagates with group velocity given by Eq. (9.134). If a sensor is placed at a distance $x$ away from the origin, then the *local* wavenumber, $\kappa(x, t)$, observed at time, $t > 0$, is related to $x$ by the group velocity,

$$x = c_g\left(\kappa(x, t)\right) t \tag{9.135}$$

This relation is simply a restatement of the definition of the group velocity as the speed at which a given wavenumber, $\kappa(x, t)$, propagates in the wave train. It is the group velocity that plays the dominant role in dispersive wave propagation. Note also that the resulting wave train does not have constant wavelength, since the whole range of wavenumbers is still present, that is, $\lambda(x, t) = 2\pi/\kappa(x, t)$.

The actual sensor measurement, $u(x; t)$, at $x$ is given by combining Eqs. (9.130) and (9.132), that is,

$$u(x; t) = \alpha(x, t; \kappa) \sin[\kappa(t)x - \omega(\kappa)t] \tag{9.136}$$

where we have suppressed the dependence of $\kappa$ on $x$ and allowed the envelope to be a function of $\kappa$. We will choose the wavenumber $\kappa(t)$ at $x$ as our state variable and develop a dynamical equation for its temporal evolution by differentiating Eq. (9.135) using the chain rule

$$\frac{dc_g(\kappa)}{dt} = \frac{d\kappa}{dt} \times \frac{dc_g(\kappa)}{d\kappa} \tag{9.137}$$

to obtain

$$0 = \frac{d\kappa}{dt}\left[\frac{dc_g(\kappa)}{d\kappa}t\right] + c_g(\kappa) \tag{9.138}$$

Now, solving for $d\kappa/dt$ and substituting the expression for group velocity in terms of our original dispersion relation, we obtain

$$\frac{d\kappa}{dt} = -\frac{1}{t}\left[\frac{d\omega(\kappa)}{d\kappa}\right]\left[\frac{d^2\omega(\kappa)}{d\kappa^2}\right]^{-1}, \qquad t > 0 \tag{9.139}$$

which shows how the wavenumber evolves dynamically (temporally) as a function of the underlying dispersion relation $\omega(\kappa)$. If we couple this expression back to the original dispersive wave solution, then we can have a general continuous-time, spatiotemporal, dispersive wave, state-space representation with state defined by $\kappa(t)$.

Suppose that we sample this wave with an array of $L$-sensors *oriented* in the direction of propagation, that is, $x \to x_\ell$; $\ell = 1, \ldots, L$, giving $L$ wavenumbers and $L$ initial conditions. If the entire state-space is to be initialized at the same time, care must be taken to select the initialization time to be after the leading edge of the wave has passed through the entire array. Let $t_o$ be the time the leading edge passes the sensor $L$, the sensor farthest from the origin, then $x_L = c_g\left(\kappa_L(t_o)\right)t_o$, where $\kappa_L(t_o)$ is the wave number for the maximum group velocity. The initial conditions for the other sensors in the array are obtained by solving $x_\ell = c_g\left(\kappa_\ell(t_o)\right)t_o$ for each $\ell$. Thus the "spatially" sampled, dispersive wave, state-space model is given by

$$\begin{aligned}
\frac{d\kappa_\ell}{dt} &= -\frac{1}{t}\left[\frac{d\omega(\kappa_\ell)}{d\kappa_\ell}\right]\left[\frac{d^2\omega(\kappa_\ell)}{d\kappa_\ell^2}\right]^{-1}, \qquad t \geq t_o \\
u_\ell(t) &= \alpha\left(t;\kappa_\ell\right)\sin[\kappa_\ell x_\ell - \omega(\kappa_\ell)t] \\
&\quad \kappa_\ell(t_o), \quad \ell = 1, \ldots, L
\end{aligned} \tag{9.140}$$

We can further discretize this model, temporally, by sampling $t \to t_k$, and also by replacing the derivatives with their first difference approximations. Since we know that the dispersive medium in which the wave propagates is uncertain, we can also characterize uncertainties with statistical models, one of which is the *Gauss-Markov*

model [8]. Performing these operations, we achieve our desired result, the discrete, spatiotemporal, *dispersive wave state-space* model:

$$\kappa_\ell(t_{k+1}) = \kappa_\ell(t_k) - \frac{\triangle_{t_k}}{t}\left[\frac{d\omega}{d\kappa}\right]\left[\frac{d^2\omega}{d\kappa^2}\right]^{-1} + \triangle_{t_k} w_\ell(t_k)$$
$$u_\ell(t_k) = \alpha\,(t_k;\kappa_\ell)\sin\left[\kappa_\ell(t_k)x_\ell - \omega(\kappa_\ell)t_k\right] + v_\ell(t_k)$$
$$\kappa_\ell(t_o), \quad \ell = 1,\ldots,L$$

where $w_\ell(t_k)$ and $v_\ell(t_k)$ are assumed zero-mean, gaussian noise sources, with respective covariances, $R_{w_\ell w_\ell}(t_k)$, $R_{v_\ell v_\ell}(t_k)$. The general vector Gauss-Markov form can be found in Refr. [8] and is simply given by

$$\kappa(t_{k+1}) = \mathbf{a}[\kappa, t_k] + \triangle_{t_k}\mathbf{w}(t_k),$$
$$\mathbf{u}(t_k) = \mathbf{c}[\kappa, t_k] + \mathbf{v}(t_k), \tag{9.141}$$

where $\mathbf{a}[\cdot]$, $\mathbf{c}[\cdot]$ are the respective nonlinear vector system and measurement functions with the corresponding state and measurement covariances defined by: $\mathbf{P}(t_{k+1})$ and $R_{v_\ell v_\ell}(t_k)$, with the system and measurement jacobians, $\mathbf{A}[\kappa] := \partial\mathbf{a}/\partial\kappa$ and $\mathbf{C}[\kappa] := \partial\mathbf{c}/\partial\kappa$. The subsequent development of our processor will rely on this statistical formulation for both simulation and estimation.

### 9.6.3 Dispersive Model-Based Processor

Next we outline the *MBP* based on the vector representation of the wavenumbers and wave field, that is, we define the vectors, $\mathbf{u}(t_k) := [u_1(t_k),\ldots,u_L(t_k)]'$ and $\kappa := [\kappa_1(t_k),\ldots,\kappa_L(t_k)]'$. Once the wave is characterized by the underlying Gauss-Markov representation, then *dispersive wave estimation problem* can be specified as follows:

> **GIVEN** the approximate Gauss-Markov model (above) characterized by the dispersive wave state-space model of Eq. 9.141 and a set of noisy measurements, $\{\mathbf{u}(t_k)\}$. **FIND** the best (minimum error variance) estimate of the wave, that is, find $\hat{\mathbf{u}}(t_k)$.

The minimum variance solution to this problem can be obtained by the maximizing *a posteriori* density, leading to the so-called *MAP equation*,

$$\nabla_k \ \ln \Pr\ (\kappa(t_{k+1})|U_{k+1})\Big|_{\kappa=\hat{\kappa}_{\text{MAP}}} = \mathbf{O} \tag{9.142}$$

Differentiating the posterior density and noting that $\hat{\kappa}(t_{k+1|k})$ and $\epsilon(t_k)$ are both functions of the data set, $U_k$, we obtain that $\hat{\kappa}_{\text{MAP}}(t_{k+1}) = \hat{\kappa}(t_{k+1|k+1})$ as

$$\hat{\kappa}(t_{k+1|k+1}) = \hat{\kappa}(t_{k+1|k}) + \mathbf{K}(t_{k+1})\epsilon(t_{k+1}) \tag{9.143}$$

where this expression is the corrected estimate (below) and shown in the *XMBP* algorithm of Chapter 6. Thus, the model-based solution to this wave enhancement problem can be achieved using the nonlinear *XMBP* algorithm which is given (simply) as follows:

*Prediction*

$$\hat{\kappa}_\ell(t_{k+1|k}) = \hat{\kappa}_\ell(t_{k|k}) - \frac{\triangle_{t_k}}{t}\left[\frac{d\omega}{d\kappa}\right]\left[\frac{d^2\omega}{d\kappa^2}\right]^{-1}, \qquad \ell = 1, \ldots, L \tag{9.144}$$

*Innovation*

$$\begin{aligned}\epsilon(t_k) &= \mathbf{u}(t_k) - \hat{\mathbf{u}}(t_{k+1|k}) \\ \hat{u}_\ell(t_{k+1|k}) &= \alpha(t_k; \hat{\kappa}_\ell(t_{k+1|k}))\sin\left[\hat{\kappa}_\ell(t_{k+1|k})x_\ell - \omega(\hat{\kappa}_\ell)t_k\right], \qquad \ell = 1, \ldots, L\end{aligned}$$

*Correction*

$$\hat{\kappa}(t_{k+1|k+1}) = \hat{\kappa}(t_{k+1|k}) + \mathbf{K}(t_k)\epsilon(t_k) \tag{9.145}$$

*Gain*

$$\mathbf{K}(t_k) = \tilde{\mathbf{P}}(t_{k+1|k})\mathbf{C}^T(t_k)\mathbf{R}_{\epsilon\epsilon}^{-1}(t_k) \tag{9.146}$$

Here the predicted and corrected covariances are given Chapter 6. From that chapter we see that in order to construct the optimal dispersive wave model-based processor, we must not only specify the required initial conditions, but also the respective system and measurement jacobians: $da[\cdot]/d\kappa$ and $dc[\cdot]/d\kappa$. For our general solution, we see that

$$\begin{aligned}a\left[\kappa_\ell, t_k\right] &= \kappa_\ell(t_k) - \frac{\triangle_{t_k}}{t_k}\left[\frac{d\omega}{d\kappa}\right]\left[\frac{d^2\omega}{d\kappa^2}\right]^{-1} \\ c\left[\kappa_\ell, t_k\right] &= \alpha(t_k; \kappa_\ell)\sin\left[\kappa_\ell(t_k)x_\ell - \omega(\kappa_\ell)t_k\right], \qquad \ell = 1, \ldots, L\end{aligned}$$

The jacobians then follow easily as

$$A\left[\kappa_\ell(t_k), t_k\right] = 1 - \frac{\triangle_{t_k}}{t_k}\left(1 - \frac{\left[\frac{d\omega}{d\kappa}\right]\left[\frac{d^3\omega}{d\kappa^3}\right]}{\left[\frac{d^2\omega}{d\kappa^2}\right]^2}\right), \qquad \ell = 1, \ldots, L$$

$$\begin{aligned}C\left[\kappa_\ell(t_k), t_k\right] &= \frac{d\alpha_\ell(t_k)}{d\kappa_\ell}\sin\left[\kappa_\ell(t_k)x_\ell - \omega(\kappa)t_k\right] \\ &\quad + \alpha_\ell(t_k)\cos\left[\kappa_\ell(t_k)x_\ell - \omega t_k\right]\left(x_\ell - \frac{d\omega}{d\kappa}t_k\right)\end{aligned}$$

Before we complete this section, let us consider a simple example and see how it fits into this framework. Let us assume that we have a simple, *nondispersive*, one-dimensional monochromatic wave model given by

$$u(x,t) = \alpha \sin\left[\kappa_o x - \omega_o t\right] \tag{9.147}$$

where the frequency, $\omega_o = \kappa_o \times c$ and $c$ is the velocity in the medium. Clearly, the phase and group velocities are identical, since

$$c_p(\kappa) = \frac{\omega_o}{\kappa_o} = c, \quad c_g(\kappa_o) = \frac{d\omega_o}{d\kappa} = c \tag{9.148}$$

In this example we use the fact that the wavenumber is a constant ($d\kappa/dt = 0$), therefore, our propagation model in the *discrete* state-space framework is simply given by

$$\begin{aligned} \kappa_\ell(t_{k+1}) &= \kappa_\ell(t_k) \\ u_\ell(t_k) &= \alpha \sin\left[\kappa_\ell x_\ell - \omega_o t_k\right], \qquad \ell = 1, \ldots, L \end{aligned} \tag{9.149}$$

Correspondingly the model-based processor then for this problem takes on the simplified form:

$$\hat{\kappa}(t_{k+1|k+1}) = \hat{\kappa}(t_{k+1|k}) + \mathbf{K}(t_k)\left[\mathbf{u}(t_k) - \hat{\mathbf{u}}(t_{k+1|k})\right] \tag{9.150}$$

where the optimal wave estimate at the $\ell$th sensor is

$$\hat{u}_\ell(t_{k+1|k}) = \alpha \sin\left[\hat{\kappa}_\ell(t_{k+1|k})x_\ell - \omega_o t_k\right] \tag{9.151}$$

and the required jacobians are simply

$$A[\kappa_\ell, t] = 1, \quad C[\kappa_\ell, t] = x_\ell - ct_k \tag{9.152}$$

It is interesting to note that if we assume a source at bearing angle $\theta$, then the component wavenumber along the array direction is $\kappa_\ell = \kappa_o \sin\theta_o$. This completes the section on dispersive wave estimation. Next we consider the application of this processor for internal ocean waves.

### 9.6.4 Internal Wave Processor

When operating in a stratified environment with relatively sharp density gradients any excitation that disturbs the pycnocline (density profile) will generate *internal ocean waves* ([43], [44], [45]). To apply our processor to the internal wave enhancement problem, we first recall the original dispersive wave system of Eqs. (9.130) and (9.132) and apply this structure to the internal wave dynamics where $u$ represents the measured *velocity field* and $\alpha(x,t)$ and $\theta(x,t)$ are the respective envelope

and phase to be specified by the internal wave structure. We define the *internal wave dispersion relation* by

$$\omega := \omega_o(\kappa) + \kappa(x,t)\nu \tag{9.153}$$

where we have included the additive velocity term to account for the effects of a doppler shift created by the ambient current. That is, in the original formulation we have replaced the position variable by $x \to x - \nu t$, which leads to the equation above. For $\omega_o(\kappa)$ we use a dispersion model based on some empirical results, the Barber approximation [45], for internal wave dispersion and group velocity. Thus we have

$$\omega_o(\kappa) = \frac{C_o\kappa(t)}{1 + \dfrac{C_o}{N_o}\kappa(t)} \tag{9.154}$$

with $C_o$ is the initial phase velocity and $N_o$ is the maximum of the buoyancy frequency profile ([41], [42]). It is also possible to derive the following approximation to the amplitude modulation function as

$$\alpha(t_k) = \frac{A}{\sqrt{t}}\left[c_p(\kappa)\right]^{3/2}\sin\left[\omega(\kappa,t_k)T_w\right] \tag{9.155}$$

where $A$ is a constant amplitude governing the overall envelope gain, $T_w$ is a temporal window width, and $c_p(\kappa)$ is the phase speed.

With this information in hand, we specify the required dispersive wave state-space model as

$$\begin{aligned}
\kappa_\ell(t_{k+1}) &= \kappa_\ell(t_k) + \frac{\triangle_{t_k}}{t_k}\frac{N_o}{2}\frac{1+\dfrac{\nu}{c_{g_o}(\kappa_\ell)}}{\sqrt{C_o c_{g_o}(\kappa_\ell)}}, \qquad t_k \geq t_o \\
u_\ell(t_k) &= \alpha_\ell(t_k)\sin\left[\kappa_\ell(t_k)x_\ell - (\omega_o(\kappa_\ell) + \kappa_\ell(t_k)\nu)t_k\right], \qquad \ell = 1,\ldots,L
\end{aligned}$$

Thus the *MAP* estimator can now be constructed using the formulation of the previous section. Now all that remains to completely define the MAP processor is the required system and measurement jacobians. From the internal wave dynamics above we have that

$$\begin{aligned}
a\left[\kappa_\ell,t_k\right] &= \kappa_\ell(t_k) + \frac{\triangle_{t_k}}{t_k}\frac{N_o}{2}\frac{1+\dfrac{\nu}{c_{g_o}(\kappa_\ell)}}{\sqrt{C_o c_{g_o}(\kappa_\ell)}}, \qquad t_k \geq t_o \\
c\left[\kappa_\ell,t_k\right] &= \alpha_\ell(t_k)\sin\left[\kappa_\ell(t_k)x_\ell - (\omega_o(\kappa_\ell) + \kappa(t_k)\nu)t_k\right]
\end{aligned} \tag{9.156}$$

The jacobians are then

$$A[\kappa_\ell, t_k] = 1 - \frac{\triangle_{t_k}}{t_k}\left[1 - \frac{3}{2}\frac{c_{g_o}(\kappa_\ell) + v}{c_{g_o}(\kappa_\ell)}\right], \qquad t_k \geq t_o$$

$$C[\kappa_\ell, t_k] = \frac{d\alpha_\ell(t_k)}{d\kappa_\ell}\sin[\kappa_\ell(t_k)x_\ell - (\omega_o(\kappa_\ell) + \kappa(t_k)v)t_k] + \alpha_\ell(t_k)\cos[\kappa_\ell(t_k)x_\ell - (\omega_o(\kappa_\ell) + \kappa(t_k)v)t_k]\left(x_\ell - \left(\frac{d\omega_o(\kappa_\ell)}{d\kappa_\ell} + v\right)t_k\right) \tag{9.157}$$

with the required derivatives available from the internal wave dispersion and envelope functions as

$$\frac{d\omega(\kappa_\ell)}{d\kappa_\ell} = \frac{C_o}{\left(1 + \frac{C_o}{N_o}\kappa_\ell\right)^2} + v$$

$$\frac{d\alpha_\ell(t_k)}{d\kappa_\ell} = \frac{A}{C_o\sqrt{t_k}}\left(\frac{\omega(\kappa_\ell)}{\kappa_\ell}\right)^{5/2}\left[T_w\left(\frac{\omega(\kappa_\ell)}{\kappa_\ell}\right)\cos(\omega(\kappa_\ell)T_w) - \frac{3}{2}\frac{C_o}{N_o}\sin(\omega(\kappa_\ell)T_w)\right] \tag{9.158}$$

Next we evaluate the performance of the *MBP* applied to noisy internal wave dynamics. We assess the overall performance of the processor using the results of recursive estimation theory [8] as before; that is, we perform basic statistical tests that are used to indicate whether or not the processor is "tuned" and then observe the enhancement achieved. The data used in this study are synthesized initially by a sophisticated internal wave simulator [46] (truth model), which has been used to both design experiments and analyze the phenomenology observed. We use this simulator to synthesize internal wave dynamics corresponding to an internal wave field experiment performed in Loch Linnhe, Scotland, in 1994 ([47], [48]). The simulation (shown in Figure 9.38) was performed based on the *SNR* defined by $SNR := \sigma^2/R_{vv}$, where $\sigma$ is the energy in the true image (scaled to unit variance) and $R_{vv}$ is the measurement noise variance extracted from experimental data. The spatio-temporal velocity field includes the entire range of 361 temporal samples at $\triangle t = 5$ s representing a propagation time of approximately half an hour. Spatially we assume a line array of 30 sensors (enough to illustrate the wave structure) spaced at $\triangle x = 4$ m representing an aperture of 120 m. For this internal wave simulation, we choose a peak buoyancy frequency of $N_o = 0.137$ r/s and a long wave ($\kappa = 0$) phase speed of $C_o = 0.34$ m/s with the ambient current (doppler shift) of $v = -0.1$ cm/s. The data was contaminated with additive gaussian noise at a $-23$ *dB SNR*. Here we observe the effect of additive gaussian noise in obscuring (visually) the internal wave dynamics of the synthesized velocity field. We see the noisy internal wave spatiotemporal signals synthesized in Figure 9.38*a*.

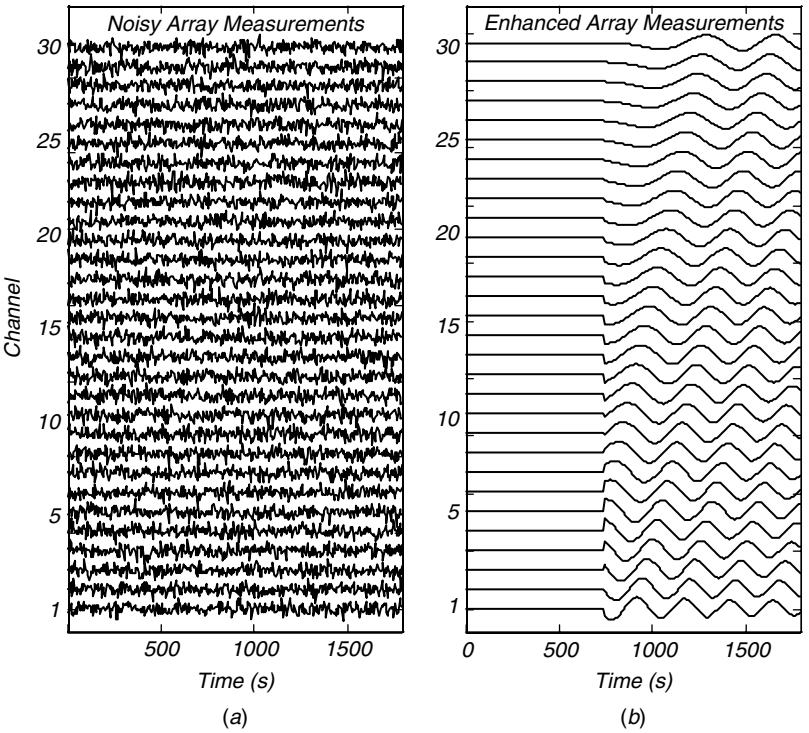



**Figure 9.38.** Equivalent model-based internal wave spatiotemporal enhancement: (*a*) Synthesized wave (−23 dB SNR). (*b*) Enhanced internal wave.

The model-based processor is designed as follows. After choosing the various initial conditions and noise variances by trial and error, we "tune" the processor. A global quantification of overall *MBP* tuning performance is found by investigating the properties of the measurement residuals or the innovation sequences and their associated statistics. When the model can "explain" or "fit" the data, we expect very little information about the process to remain in the residuals, therefore, the resulting sequence should be completely random (uncorrelated or white) and unbiased (zero mean). A necessary and sufficient condition for an optimal *XMBP* is that the innovation sequence is *zero-mean* and *white*, then the processor is considered to be approaching the optimal solution. It should also be noted that the *MBP* is tuned primarily by adjusting the process noise covariance matrix ($R_{ww}$) and testing the resulting innovations sequence as described previously.

We use *SSPACK_PC*, [24] a model-based signal-processing package available in *MATLAB* [25] to design the processor. The model-based wave enhancer is able to extract the internal wave signatures quite effectively even though the embedded dispersive wave model is just an approximation to the actual wave dynamics. We show the spatiotemporal interpretation (and display) of the noisy and enhanced signals in Figure 9.39. To confirm the processor performance that we observed

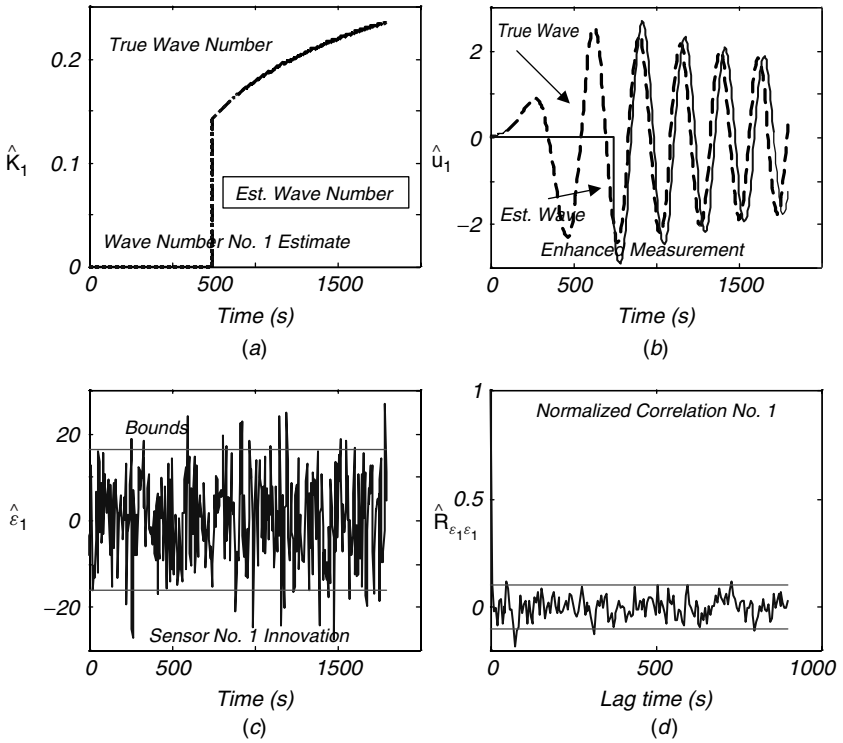



**Figure 9.39.** Model-based internal wave estimation: (*a*) Estimated (*solid line*) and true (*dashed line*) wavenumber 1. (*b*) Estimated (*solid line*) and true (*dashed line*) internal wave. (*c*) Residual/innovation (8.9% out). (*d*) Zero-mean/whiteness testing ($0.02 < 1.5$, 3.9% out).

visually, we perform individual whiteness tests on each of the temporal sensor outputs. A set of statistical tests for this particular simulation are shown in Figures 9.39 to 9.41, where we have chosen the following sensor outputs: 1, 20, 30 to give a feel for the processor performance across the array. In each figure we show the performance at that sensor. First, we observe the wave number (state) estimate (solid line) along with the true wavenumber (dashed line) from the synthesizer. Clearly, the estimates essentially overlay the true wavenumber dynamics. Next we see the estimated velocity field (solid line) at the noted sensor position and that provided by the sophisticated synthesizer (dashed line). As expected, the estimates are not perfect lacking enough of the detailed dynamics to generate the initial onset and propagation along with some phase differences. However, the estimates based on the generic model appear reasonably adequate for this application. Next we observe the corresponding sensor innovation indicating the difference between the noisy and predicted measurement along with the bounds predicted by the processor. Here we note that 95% of the innovations should lie within the bounds indicating a reasonable processor. In this case the samples exceed the bounds, somewhat, indicating

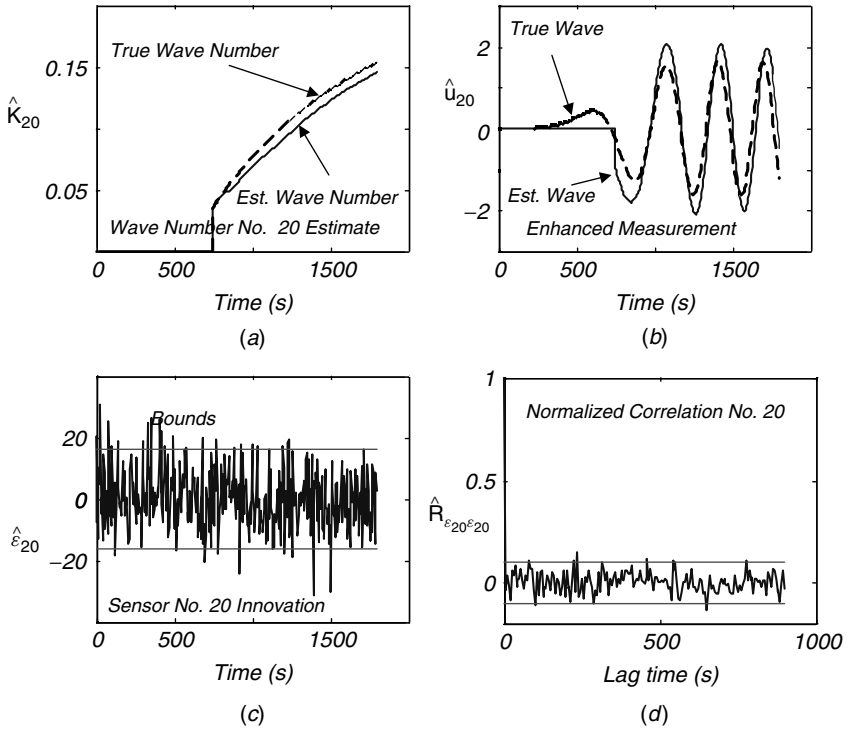



**Figure 9.40.** Model-based internal wave estimation: (*a*) Estimated (*solid line*) and true (*dashed line*) wavenumber 20. (*b*) Estimated (*solid line*) and true (*dashed line*) internal wave. (*c*) Residual/innovation (8.9% out). (*d*) Zero-mean/whiteness testing ($0.14 < 1.4$, 4.4% out).

the processor could probably be tuned even better, but for demonstration purposes these results are satisfactory. Finally, the corresponding zero-mean/whiteness tests for the selected sensors are shown. As mentioned these statistical tests are necessary to assess the overall performance of the processor. In these representative realizations each of the sensor innovations are deemed statistically *white*, that is, 95% of the sample correlation lie within or equivalently 5% fall outside the bounds (shown in the figure). They are also *zero-mean*, that is, the estimated mean is less than the calculated bound (see figure caption). Performing these tests on all of the sensor innovation outputs reveals that each individually satisfies these statistical tests indicating zero mean/white innovations and a near optimal processor. However, all that this indicates is that the generic dispersive wave model is robust enough to capture the dominant internal wave dynamics thereby indicating the potential for more practical use when applied to noisy measurement data. Thus these simulations show that the Gauss-Markov formulation enables us to capture various uncertainties of internal waves as well as its associated statistics in a completely consistent framework [8].

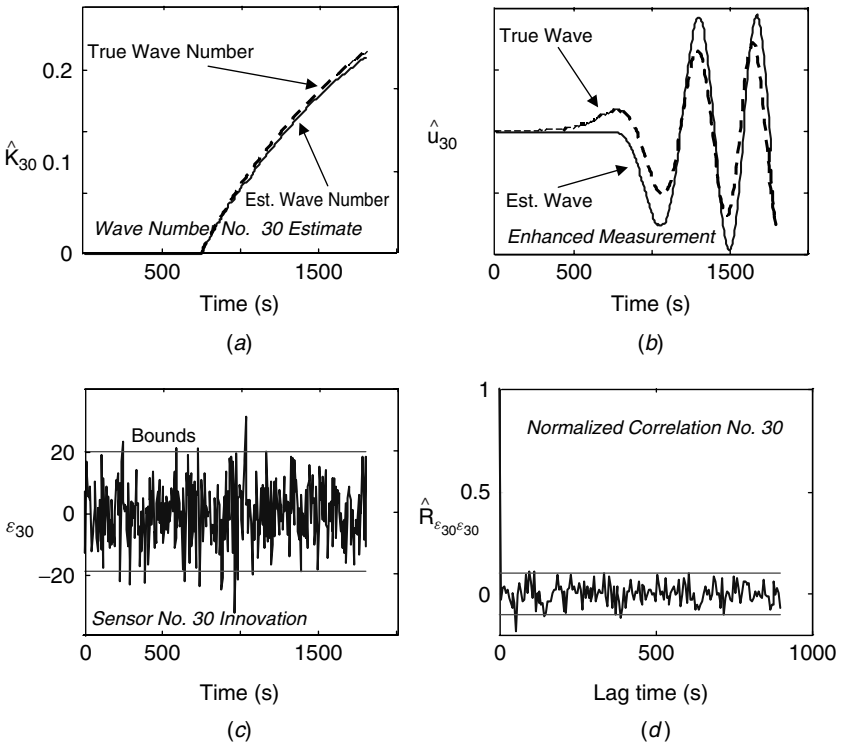



**Figure 9.41.** Model-based internal wave estimation: (*a*) Estimated (*solid line*) and true (*dashed line*) wavenumber 30. (*b*) Estimated (*solid line*) and true (*dashed line*) internal wave. (*c*) Residual/innovation (3.9% out). (*d*) Zero-mean/whiteness testing ($0.31 < 1.4$, 2.8% out).

To further assess the feasibility of this approach, we ran the enhancer on other synthetic data sets at 0 dB, −10 dB, and −13 dB with similar results, that is, the model-based approach enabled a near-optimal Bayesian solution with all sensor innovation sequences statistically testing as zero-mean/white. These preliminary results have led us to continue to pursue this approach.

We summarize the results of this application:

| | |
|---|---|
| Criterion: | $J = \text{trace}\ \ \tilde{P}(t\|t)$ |
| Models: | |
| Signal: | $\kappa(t_{k+1}) = \mathbf{a}[\kappa, t_k] + \triangle_{t_k}\mathbf{w}(t_k)$ |
| Measurements: | $\mathbf{u}(t_k) = \mathbf{c}[\kappa, t_k] + \mathbf{v}(t_k)$ |
| Noise: | $w \sim N(0, R_{ww})$ and $v \sim N(0, R_{vv})$ |
| Algorithm: | $\hat{\kappa}(t_{k+1\|k+1}) = \hat{\kappa}(t_{k+1\|k}) + \mathbf{K}(t_{k+1})\epsilon(t_{k+1})$ |
| Quality: | $\tilde{P}(t\|t) = (I - K(t)C(t))\tilde{P}(t\|t-1)$ |

This completes the application of *MBP* design to the generic dispersive wave estimation problem.

### 9.6.5 Summary

In this section we developed a generic dispersive wave processor. Starting with a general solution to the propagation of waves in a dispersive medium, we developed the approximate (due to nonlinear systems) maximum a posteriori (*MAP*) solution using a Bayesian formulation of the wave-field enhancement problem. The results are significant in that all that is required is the envelope or equivalently amplitude modulation function and dispersion relation to completely specify the underlying wave system. It is in fact the particular dispersion relation that enables the differentiation between acoustic and electromagnetic radiation, ocean surface waves and internal gravity waves or seismic waves or any wave system for that matter. The generality of this result enables the specification of a particular wave system by its underlying envelope and dispersion and then applying the algorithm to obtain the *MAP* solution.

## 9.7 *MBP* FOR GROUNDWATER FLOW

There exists a need for methods to simplify groundwater contaminant transport models in order to perform risk assessments for licensing and regulating long-term nuclear waste repositories. This application discusses the development of the model-based approach to the development of groundwater flow models and their application to demonstrate to subsurface hydrology. This investigation is aimed at applying the *MBP* to a synthesized a groundwater aquifer test in order to identify physical aquifer parameters ([49], [50]).

Suppose that a pump drawdown test is set up with a constant displacement pump discharging water from a single well that completely penetrates a confined homogeneous isotropic aquifer of constant thickness. Further assume that a single observation well is drilled some distance away from the discharging well and that all the information available is a set of noise corrupted measurements of the piezometric surface at the observation well. We would like to estimate the transmissivity and storativity for the aquifer.

The approach involves starting with the partial differential equation model for groundwater flow and transforming it to a system of first-order ordinary differential equations by finite differencing. Next, the corresponding Gauss-Markov representation of the system follows leading to the *MBP* design for this problem.

### 9.7.1 Groundwater Flow Model

In this section we discuss the development of a hydrogeological model used to determine the properties of a simulated aquifer. A pump drawdown test ([49]–[52]) is a fundamental technique used in aquifer tests to determine its hydrogeological

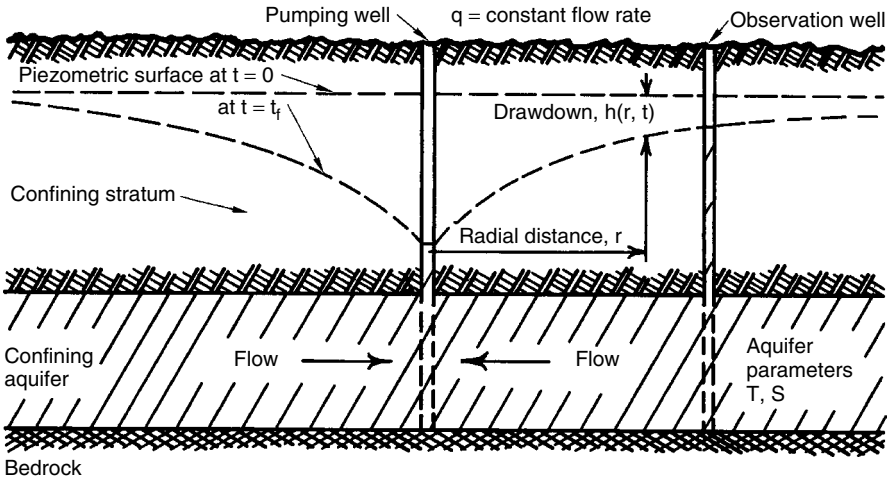



**Figure 9.42.** Aquifer pump drawdown test depicting the pumping as well as observation well along with the geology.

properties. The effect of pumping water from a well at a known rate is measured in distant observation wells penetrating the aquifer as depicted in Figure 9.42. The purpose of the test is to determine relationships among pumping rate, drawdown (lowering of the water table), and time in order to estimate the physical properties of the aquifer. We assume that the aquifer is (1) confined, (2) infinite areal extent, (3) homogeneous, (4) isotropic, and (5) uniform thickness. We also assume that the wells completely penetrate the aquifer, are of negligible diameter, and confined to radial flow with the pump rate assumed constant. When the pumping well is operated, water is continuously withdrawn from storage within the aquifer as the cone of depression in the piezometric surface progresses radially outward from the well. This water is released by the compaction of the aquifer and its associated beds and by the expansion of the water itself. Assume that the water is released instantaneously with a decline in head caused by decreasing pore pressure. The *groundwater flow* model (polar coordinates) is given by [52]

$$\left(\frac{T}{rS}\right)\nabla_r\left[r\nabla_r h(r;t)\right] = \nabla_t h(r;t) \tag{9.159}$$

or equivalently

$$\nabla_r^2 h(r;t) + \left(\frac{1}{r}\right)\nabla_r h(r;t) = \left(\frac{S}{T}\right)\nabla_t h(r;t) \tag{9.160}$$

where $h(\cdot)$ is the drawdown in meters ($m$), $r$ is the radial distance from the pumping well (m), $T$ is the transmissivity ($m^3$/Pa-s), $S$ is the storage coefficient and $t$ is time (s). The initial condition for $r \geq 0$ is $h(r,0) = h_o = 0,\ m\ \forall t < 0$, implying

a uniform hydraulic head before pumping. The boundary conditions for all $t \geq 0$ are $\lim_{r\to\infty} h(r;t) = 0$ and $\lim_{r\to 0} [r\nabla_r h(r;t)] = q/2\pi T$ for a constant discharge rate of $q$ ($m^3$/s).

This groundwater flow equation can be converted into a set of differential-difference equations by approximating the spatial variable $r$ using finite (central) differences. The boundary conditions are included in the relations as well. As $r \to 0$, the pump is included as part of a step input of magnitude, $q/2\pi T$. As $r \to \infty$, the pressure is essentially constant at the boundary; therefore the last node $N$ is chosen to lie at a distance from the pumping well that is much greater than that of the observation well. The general set of space-time equations evolve as

$$\dot{h}(r_i;t) = \begin{cases} \left(\frac{T}{S}\right) C_{13} h(r_2;t) - \left(\frac{T}{S}\right) C_{12} h(r_1;t) - \left(\frac{1}{S}\right) C_{11} q\mu(t), & i = 1 \\ \left(\frac{T}{S}\right) C_{i3} h(r_{i+1};t) - \left(\frac{T}{S}\right) C_{i2} h(r_i;t) + \left(\frac{T}{S}\right) C_{i1} h(r_{i-1};t), & 1 < i < N \\ -\left(\frac{T}{S}\right) C_{N2} h(r_N;t) + \left(\frac{T}{S}\right) C_{N1} h(r_{N-1};t), & i = N \end{cases} \tag{9.161}$$

where

$$\begin{aligned} C_{13} &= \frac{1}{r_1(r_2 - r_1)};\ C_{12} = C_{13};\ C_{11} = \frac{1}{\pi r_1(r_2 + r_1)} \\ C_{N2} &= \frac{r_N + r_{N-1}}{\lambda r_N (r_N - r_{N-1})^2};\ C_{N1} = C_{N2};\ \lambda \text{ constant} \\ C_{i3} &= \frac{r_i + r_{i-1}}{r_i(r_{i+1} - r_i)(r_{i+1} - r_{i-1})};\ C_{i2} = \frac{2}{(r_{i+1} - r_i)(r_i - r_{i-1})}; \\ C_{i1} &= \frac{r_i + r_{i+1}}{r_i(r_{i+1} - r_{i-1})(r_i - r_{i-1})} \end{aligned}$$

These equations can now be cast into the Gauss-Markov framework by defining the state vector, $\mathbf{x}(t) := [h(r_1;t)\ h(r_2;t)\ \cdots\ h(r_N;t)]'$. The Gauss-Markov model for *groundwater flow* is in the general form

$$\begin{aligned} \dot{\mathbf{x}}(r;t) &= A(r)\mathbf{x}(r;t) + \mathbf{g}(r)\mu(t) + \mathbf{w}(t) \\ y(r_o;t_k) &= \mathbf{c}'\mathbf{x}(r;t_k) + \mathbf{v}(t_k) \qquad \text{for } t_k \text{ sampled time (s)} \end{aligned} \tag{9.162}$$

for $\mathbf{c}'$ a unit row vector, and $r_o$ is the observation well position and $\mathbf{w}$, $\mathbf{v}$ are zero-mean, gaussian with respective covariances, $\mathbf{R}_{ww}$ and $\mathbf{R}_{vv}$. The system matrix is

tridiagonal as

$$A(r) := \begin{bmatrix} -C_{12} & C_{13} & & & \mathbf{O} \\ C_{21} & C_{22} & C_{23} & & \\ & \ddots & \ddots & \ddots & \\ \mathbf{O} & & C_{(N-1)1} & -C_{(N-1)2} & C_{(N-1)3} \\ & & & -C_{N1} & -C_{N2} \end{bmatrix}$$

$$\mathbf{g}(r) = \begin{bmatrix} -C_{11}\frac{q}{S} \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \tag{9.163}$$

A Gauss-Markov simulation using this model was performed with the results shown in Figure 9.43. A 28-state model was used which gave reasonable solution of the original (noise-free) partial differential equation solution (1.5% error). The true model parameters for this run were: $T = 1 \times 10^{-8}$ m$^3$/Pa-s, $S = 5 \times 10^{-7}$ m/Pa, $q = 1 \times 10^{-2}$ m$^3/s$, $r_o = 100$ m and $N = 28$ with 10-nodes between pumping and

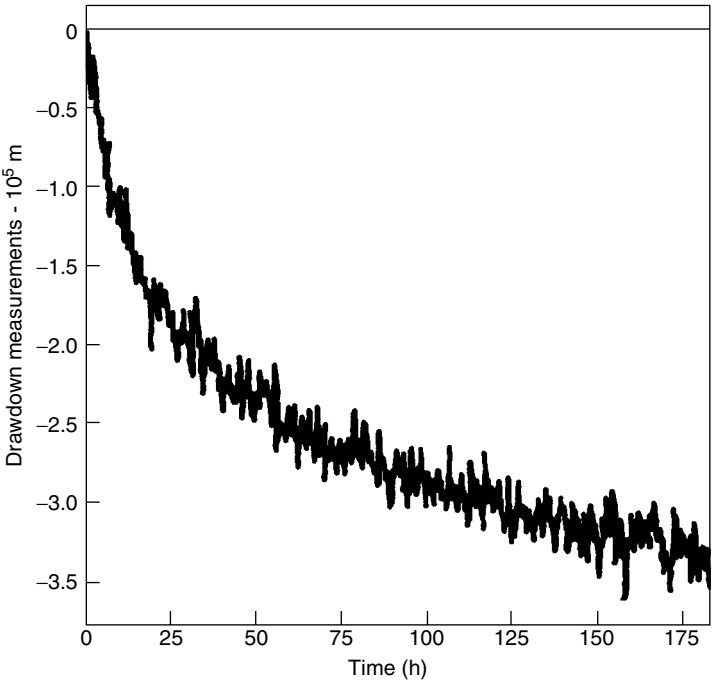



**Figure 9.43.** Noisy observation well measurement at a distance of 100 m from the pumping well with $R_{vv} = 10^8$m$^2$ corresponding to 10% measurement error at full scale.

observation wells. This completes the Gauss-Markov simulation, next we consider the *AMBP* design.

### 9.7.2 *AMBP* Design

In this subsection we consider the design and application of *AMBP* for the groundwater application. In order to solve the joint parameter/state estimation problem, we used the *AMBP* of Chapter 8. The processor was implemented using the necessary jacobians. The data were generated using the 28-state truth model and a reduced order, 21-state model was used in the processor. Initially the transmissivity and storage coefficient were guessed with large errors and the states and parameters estimated jointly. The initial errors were selected from a gaussian distribution with a true mean and 25% error variance, $x(t_o) \sim \mathcal{N}(x_{\text{true}}, (0.25\ x_{\text{true}})^2)$. An ensemble of 10 runs were generated with typical simulations shown in Figure 9.44 and Figure 9.45. In Figure 9.44 we see the reconstructed drawdown (state) estimates with associated estimation errors and bounds. We see that the estimates are in error due to the uncertainty in the parameters, but eventually track the actual drawdown

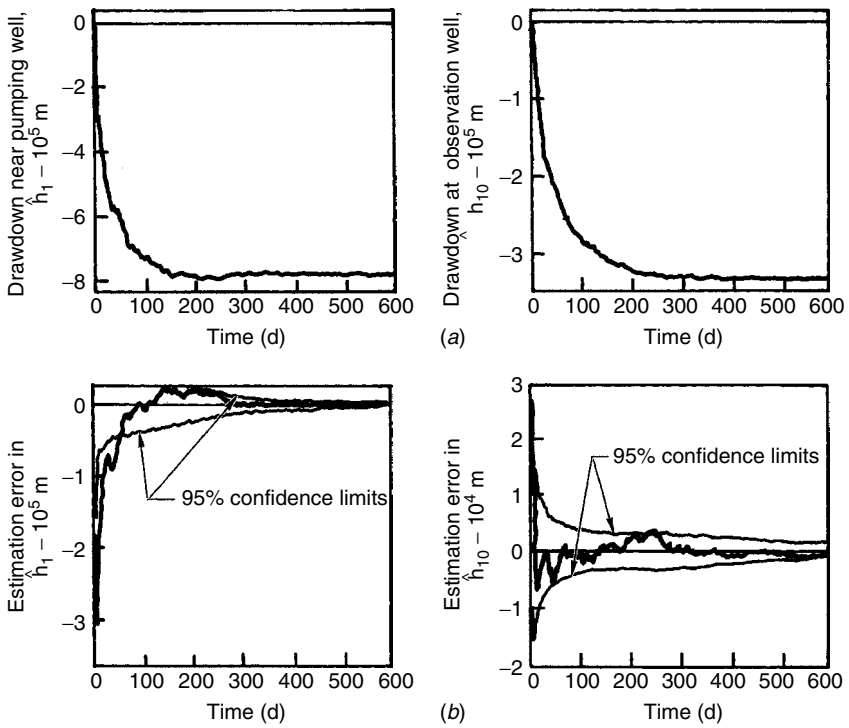



**Figure 9.44.** Drawdown (state) estimates using 21-state processor near pumping ($\hat{h}_1$) and observation ($\hat{h}_{10}$) wells using the *AMBP*: (*a*) Estimated drawdown. (*b*) Estimation errors.

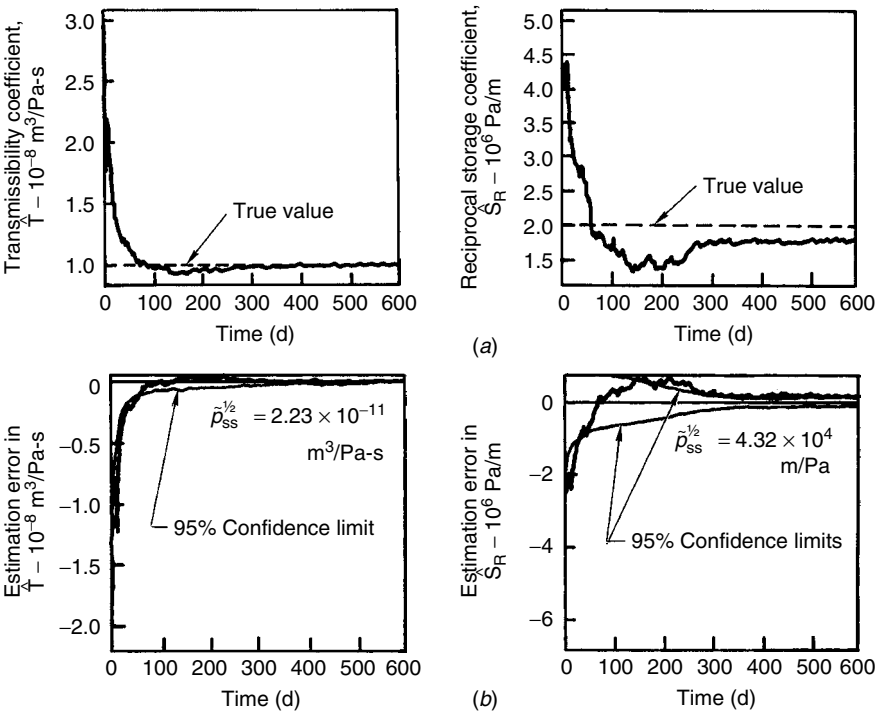



**Figure 9.45.** Transmissivity and storage coefficient (parameter) estimates: (*a*) Estimated parameters. (*b*) Estimation errors (1%, and 10%).

**Table 9.4. Ensemble Transmissivity and Storage Coefficient Ensemble (10 Runs) Statistics**

| Parameter | Truth | Estimated Mean | Percent Standard Deviation | Bias Error (%) | % Mean Error | Error S.D. | Maximum Error | Minimum Error |
|---|---|---|---|---|---|---|---|---|
| $T(\times 10^{-8})$ | 1.0 | 1.005 | 0.0007 | −0.0051 | 0.510 | 0.0007 | 0.0058 | 0.0038 |
| $S_R(\times 10^6)$ | 2.0 | 2.074 | 0.0032 | −0.0740 | 7.400 | 0.0032 | 0.076 | 0.0675 |

values: $\sqrt{\tilde{P}_{11}} = 2 \times 10^3$ m and $\sqrt{\tilde{P}_{22}} = 7 \times 10^2$ m. Thus the estimates are quite reasonable after the parameters converge. Joint parameter estimates of transmissivity and (reciprocal) storage coefficient are also shown for this run in Figure 9.45. For this realization the *AMBP* performed quite well with relative parameter estimation errors of approximately 0.3% and 8.5% respectively for $\hat{T}$ and $\hat{S}_R$. The ensemble results are shown in Table 9.4.

We summarize the results of this application:

| | |
|---|---|
| Criterion: | $J = \text{trace}\ \ \tilde{P}(t\|t)$ |
| Models: | |
| Signal: | $\dot{\mathbf{x}}(r;t) = A(r)\mathbf{x}(r;t) + \mathbf{g}(r)\mu(t) + \mathbf{w}(t)$ |
| Measurements: | $y(r_o;t_k) = \mathbf{c}'\mathbf{x}(r;t_k) + \mathbf{v}(t_k)$ |
| Noise: | $w \sim \mathcal{N}(0, R_{ww})$ and $v \sim \mathcal{N}(0, R_{vv})$ |
| Algorithm: | $\hat{\mathbf{x}}(t_{k+1\|t_{k+1}}) = \hat{\mathbf{x}}(t_{k+1\|t_k}) + \mathbf{K}(t_{k+1})\epsilon(t_{k+1})$ |
| Quality: | $\tilde{P}(t_{k+1}\|t_{k+1}) = (I - K(t_{k+1})C(t_{k+1}))\tilde{P}(t_{k+1}\|t_k)$ |

This completes the application of *AMBP* design to the groundwater flow estimation problem.

### 9.7.3 Summary

In this section we developed an *AMBP* applied to a hydrological drawdown test. Starting with a groundwater flow model using a finite difference solution to the partial differential equations, we produced a Gauss-Markov simulation based on a validated 28-state truth model. Using data synthesized from this model, we developed the *AMBP* and analyzed its performance based on an ensemble of data. The results were quite reasonable indicating the feasibility of using the model-based approach to solve groundwater flow estimation problems.

This completes the chapter on model-based physics applications.

## 9.8 SUMMARY

In this chapter we demonstrated the applicability of *MBP* through a variety of *physics-based* case studies. The major purpose was to start from a definition (sometimes vague) of a problem and cast it into the model-based framework as described in Chapter 1 of this text. We chose a variety of applications using a suite of the algorithms discussed herein. Starting with the first application of estimating parameters from reentry vehicle radar signatures, we demonstrated how the single input/single output *ARMAX* models can be used to perform a wide variety of model-based signal processing tasks ranging from simple parametric power spectral estimation to sophisticated recursive-in-time spectrogram estimation and event detection. The *FIR* class of Wiener *MBP* was developed to solve a nondestructive evaluation problem in laser ultrasound. Next the first multichannel application evolved from detecting and classifying structural failures using the state-space *MBP* design and innovations-based detectors. Here it was demonstrated just how effective this approach was in detecting failures or "changes from normal." Actual vibration data for nondestructive evaluation was used to apply the Condition Monitor and

demonstrate its effectiveness. Multichannel synthetic aperture array processing for sonar applications was investigated next. Here it was demonstrated just how plane and spherical wave processors could be implemented in the *MBP* framework to solve target identification and localization problems using a multichannel towed array creating a synthetic aperture. A more sophisticated shallow ocean acoustic application followed demonstrating the effectiveness of an *AMBP* employing a normal-mode propagation model. Here the performance of the processor coupled to a nonlinear optimizer proved to be an effective approach in solving the target localization problem. A generic model-based dispersive processor *MBP* was then developed and applied to estimate the internal wave enhancement problem. It was shown that this processor was applicable to any dispersive wave type problem. Finally, the application of the *AMBP* to a groundwater flow estimation problem was discussed and analyzed. Here it was shown how powerful the model-based approach can be even in this complex environment. Starting with the partial differential equations governing the flow, a multichannel state-space model was developed to estimate both the drawdown and aquifer parameters of high interest.

This completed the suite of physics-based applications of model-based processing. Coupled to the case studies and examples throughout the text, it should give the reader a pragmatic viewpoint of model-based processing.

## REFERENCES


1. Lincoln Laboratory, "Reduction, analysis, and reporting of measurements from KMR sensors." Lexington, MA: MIT Lincoln Labs., Mar. 1997.
2. H. Goldstein, *Classical Mechanics*, Reading, MA:Addison-Wesely, 1980.
3. J. M. Goodman and J. Arons. "Ionospheric Effects on Modern Electronic Systems," *Proc. IEEE.*, **78** (3), 512–527, 1990.
4. C. M. Rush. "Ionospheric radio propagation models and predictions—A mini-review," *IEEE Trans. Ant. Propag.*, **34** (9), 1163–1170, 1986.
5. J. V. Candy. "Processing of reentry vehicle signatures from radar tracking data," *LLNL Report*, UCRL-130433, 1998.
6. R. K. Crane, "Ionospheric scintillation," *IEEE Proc.*, **65** (2), 180–199, 1977.
7. J. V. Candy, *Signal Processing: The Modern Approach*, New York: McGraw-Hill, 1988.
8. J. V. Candy, *Signal Processing: The Model-Based Approach*, New York: McGraw-Hill, 1986.
9. L. Ljung. *System Identification: Theory for the User*, Englewood Cliffs, NJ: Prentice-Hall, 1987.
10. C. B. Scuby and L. E. Drain, *Laser Ultrasonics: Techniques and Applications*, Philadelphia: Hilger, 1990.
11. J. D. Aussel and J. P. Monchalin, "Precision laser-ultrasonic velocity measurement and elastic constant determination," *Ultrasonics*, **27**, 165–177, 1989.
12. J. B. Spicer and J. W. Wagner, "Comprehensive modelling of laser ultrasonic waveforms for materials characterization," in *Acousto-Optics and Acoustic Microscopy*, S. M. Gracewski and T. Kundu, eds., ASME AMD, **140** (American Society of Mechanical Engineers), 163–179, 1992.

13. J. V. Candy, G. H. Thomas, D. J. Chinn, and J. B. Spicer, "Laser ultrasonic signal processing: A model-reference approach." *J. Acoust. Soc. Am.*, **100** (1), 278–284, 1996.
14. E. Robinson and S. Treitel, *Geophysical Signal Analysis*, Englewood Cliffs, NJ: Prentice-Hall, 1980.
15. D. B. McCallen, "Ground motion estimation and nonlinear seismic analysis," *Lawrence Livermore National Laboratory Report* UCRL-JC-121667, 1995.
16. D. B. McCallen and A. Astaneh-Asl, "Computational simulation of the nonlinear response of suspension bridges," *Lawrence Livermore National Laboratory Report* UCRL-JC-128901, 1997.
17. J. V. Candy and R. B. Rozsa, "Safeguards design for a plutonium nitrate concentrator–an applied estimation approach," *Automatica*, **16**, 615–627 1980.
18. J. V. Candy and E. J. Sullivan, "Ocean acoustic signal processing: A model-based approach," *J. Acoust Soc. Am.*, **92** (6), 3185–3201, 1992.
19. V. Wowk, *Machinery Vibration: Measurement and Analysis*, New York: McGraw-Hill, 1991.
20. H. L. Van Trees, *Detection, Estimation and Modulation Theory*, New York: Wiley, 1968.
21. D. F. Specht, "Probabilistic neural networks and the polynomial Adaline as complimentary techniques for classification," IEEE Trans. *Neur. Networks*, **1**, 111–121, 1990.
22. J. V. Candy and H. E. Jones, "Processing of prosthetic heart valve sounds for single leg separation classification," *J. Acoust. Soc. Am.*, **65** (2), 180–199, 1994.
23. J. V. Candy and H. E. Jones, "Classification of prosthetic heart valve sounds: A parametric approach," *J. Acoust. Soc. Am.*, **65** (2), 180–199, 1994.
24. J. V. Candy and P. M. Candy, "SSPACK_PC: A model-based signal processing package on personal computers," *DSP Appl.*, **2**(3), 33–42, 1993 (see also **http://www.technisoft.net**).
25. Mathworks, *MATLAB Users Manual*, Natick, MA: Mathworks, 1990.
26. S. G. Azevedo, J. V. Candy, and D. L. Lager, "On-line failure detection of vibrating structures," *Proc. ASME Conf. Mechanical Vibrations and Noise*, Hartford, CT, 1980.
27. E. J. Sullivan, W. Carey, and S. Stergiopoulos, eds., "Special issue on acoustic synthetic aperture processing", *IEEE Trans. Ocean. Engr.*, **17**, 1993.
28. S. Stergiopoulos and E. J. Sullivan, "Extended towed array processing by an overlap correlator," *J. Acoust. Soc. Am.*, **86**, 158–171, 1989.
29. N. Yen and W. Carey, "Application of synthetic aperture processing to towed array data," *J. Acoust. Soc. Am.*, **86**, 754–765, 1989.
30. E. J. Sullivan and J. V. Candy, "Passive synthetic aperture processing as a Kalman filter problem," *J. Acoust Soc. Am.*, **95** (5, Prt. 2), 2953, 1994.
31. M. J. Hinich, "Maximum likelihood signal processing for a vertical array," *J. Acoust Soc. Am.*, **54**, 499–503, 1973.
32. E. J. Sullivan and J. V. Candy, "Space-time processing: The model-based approach," *J. Acoust Soc. Am.*, **102** (5), 2809–2820, 1997.
33. C. S. Clay, and H. Medwin, *Acoustical Oceanography*, New York: Wiley, 1977.
34. W. M. Carey, J. Doutt, R. Evans, and L. Dillman, "Shallow water transmission measurements taken on the New Jersey continental shelf," *IEEE J. Oceanic Engr.*, **20** (4), 321–336, 1995.

35. J. V. Candy and E. J. Sullivan. "Model-based processor design for a shallow water ocean acoustic experiment," *J. Acoust. Soc. Am.*, **95** (4), 2038–2051, 1994.
36. J. V. Candy and E. J. Sullivan "Model-based processing of a large aperture array," *IEEE Trans. Ocean Eng.*, **19** (4), 519–528, 1994.
37. P. E. Gill, W. Murray and M. H. Wright, *Practical Optimization*, New York: Academic Press, 1981.
38. J. A. Nelder and R. Mead, "A simplex method for function minimization," *Computer J.*, **7**, 308–313, 1965.
39. A. Grace, *Optimization toolbox for use with MATLAB*, Boston: MathWorks, 1992.
40. J. V. Candy and E. J. Sullivan, "Passive localization in ocean acoustics: A model-based approach," *J. Acoust Soc. Am.*, **98** (3), 1455–1471, 1995.
41. G. B. Whitham, *Linear and Nonlinear Waves*, New York: Wiley, 1974.
42. J. R. Apel, *Principles of Ocean Physics*, New York: Academic Press, 1987.
43. J. V. Candy and D. H. Chambers. "Internal wave processing: A model-based approach." *IEEE J. Oceanic Engr.*, **21** (1), 37–52, 1996.
44. J. V. Candy and D. H. Chambers. "Model-based dispersive wave processing: A recursive Bayesian solution." *J. Acoust Soc. Am.*, **105** (6), 3364–3374, 1999.
45. B. C. Barber, "On the dispersion relation for trapped internal waves," *J. Fluid Mechs.*, **252**, 31–49, 1993.
46. M. Milder, *Internal Waves Radiated by a Moving Source: Analytic Simulation*, RDA Report, RDA-TR-2702-007, 1974.
47. D. D. Mantrom, "Loch Linnhe '94: Test operations description and on-site analysis US activities," *LLNL Report.*,UCRL-ID-119197, 1994.
48. H. F. Robey and D. L. Ravizza, "Loch Linnhe experiment 1994: Background stratification and shear measurements. Part 1: Profile summary and dispersion relations," *LLNL Report.*, UCRL-ID-119352, 1994.
49. C. V. Theis, "The relation between the lowering of the piezometric surface and the rate and duration of discharge of a well using groundwater storage," *Trans. Am. Geophys. Union*, **16**, 519–524, 1935.
50. C. E. Jacob, "Flow of groundwater," in *Engineering Hydraulics*, H. Rouse, Ed., New York: Wiley, 1950.
51. R. J. DeWiest, "On the storage coefficient and the equations of groundwater flow," *J. Geophys. Res.*, **71** (4), 1117–1121, 1966.
52. S. N. Davis and R. J. DeWiest, *Hydrogeology*, New York: Wiley, 1966.
53. S. A. Azevedo, T. A. Doerr, J. V. Candy, and K. D. Pimental, "State-space models for state and parameter estimation in groundwater flow," *LLNL Report*, UCID-18575, 1980.
54. K. D. Pimental, J. V. Candy, S. A. Azevedo, and T. A. Doerr, "Simplified groundwater flow modeling: an application of Kalman filter based identification," *Math. Comput. Simul.*, **24**, 140–151, 1982.

# APPENDIX A

# PROBABILITY AND STATISTICS OVERVIEW

## A.1 PROBABILITY THEORY

Defining a sample space (outcomes), $\Omega$, a field (events), $B$, and a probability function (on a class of events), Pr, we can construct an *experiment* as the triple, $\{\Omega, B, \Pr\}$.

***Example A.1*** Consider the experiment $\{\Omega, B, \Pr\}$ of tossing a fair coin. We see the following construction:

Sample space: $\Omega = \{H, T\}$
Events: $B = \{0, \{H\}, \{T\}\}$
Probability: $\Pr(H) = p, \Pr(T) = 1 - p$

With the idea of a sample space, probability function, and experiment in mind, we can now start to define the concept of a discrete random signal more precisely. We define a discrete *random variable* as a real function whose value is determined by the outcome of an experiment. It assigns a real number to each point of a sample space $\Omega$ that consists of all the possible outcomes of the experiment. A random variable $X$ and its realization $x$ are written as

$$X(\omega) = x \qquad \text{for } \omega \epsilon \Omega \tag{A.1}$$

Consider the following example of a simple experiment.

*Model-Based Signal Processing*, by James V. Candy

***Example A.2*** We are asked to analyze the experiment of flipping a fair coin. Then the sample space consists of a head or tail as possible outcomes, that is,

$$\Omega = \{0 \quad H \quad T\} \Longrightarrow X(\omega) = x$$
$$\omega = \{H, T\}$$

If we assign a 1 for a head and 0 for a tail, then the random variable $X$ performs the mapping of

$$X(\omega = H) = x(H) = 1$$
$$X(\omega = T) = x(T) = 0$$

where $x(.)$ is called the sample value or realization of the random variable $X$.

A *probability mass function* defined in terms of the random variable, that is,

$$\mathrm{P}_X(x_i) = \Pr(X(\omega_i) = x_i) \tag{A.2}$$

and the *probability distribution function* is defined by

$$\mathrm{F}_X(x_i) = \Pr(X(\omega_i) \leq x_i) \tag{A.3}$$

These are related by

$$\mathrm{P}_X(x_i) = \sum_i \mathrm{F}_X(x_i)\delta(x - x_i) \tag{A.4}$$
$$\mathrm{F}_X(x_i) = \sum_i \mathrm{P}_X(x_i)\mu(x - x_i) \tag{A.5}$$

where $\delta$, and $\mu$ are the unit impulse and step functions, respectively.

It is easy to show that the distribution function is a monotonically increasing function (see Papoulis [1] for details) satisfying the following properties:

$$\lim_{x_i \to -\infty} \mathrm{F}_X(x_i) = 0$$
$$\lim_{x_i \to \infty} \mathrm{F}_X(x_i) = 1$$

These properties can be used to show that the mass function satisfies

$$\sum_i \mathrm{P}_X(x_i) = 1$$

Either the distribution or probability mass function completely describe the properties of a random variable. Given either of these functions, we can calculate probabilities that the random variable takes on values in any set of events on the

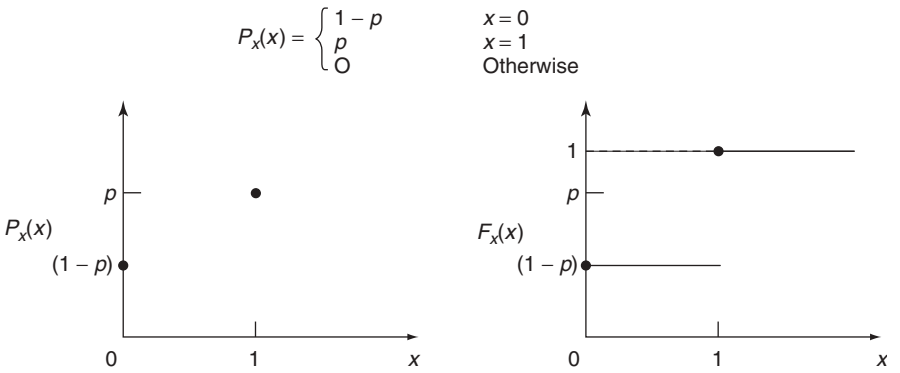



**Figure A.1.** Probability mass and distribution functions for coin tossing experiment.

real line. To complete our coin tossing example, if we define the probability of a head occurring as $p$, then we can calculate the distribution and mass functions as shown in the following example.

***Example A.3*** Consider the coin tossing experiment and calculate the corresponding mass and distribution functions. From the previous example, we have the following:

| | | | |
|---|---|---|---|
| Sample space: | $\Omega$ | $=$ | $\{H, T\}$ |
| Events: | $B$ | $=$ | $\{0, \{H\}, \{T\}\}$ |
| Probability: | $\mathrm{P}_X(x_1 = H)$ | $=$ | $p$ |
| | $\mathrm{P}_X(x_0 = T)$ | $=$ | $1 - p$ |
| Random variable: | $X(\omega_1 = H)$ | $=$ | $x_1 = 1$ |
| | $X(\omega_2 = T)$ | $=$ | $x_2 = 0$ |
| Distribution: | $\mathrm{F}_X(x_i)$ | $=$ | $\begin{cases} 1 & x_i \geq 1 \\ 1-p & 0 \leq x_i \leq 1 \\ 0 & x_i < 0 \end{cases}$ |

the mass and distribution functions for this example are shown in Figure A.1. Note that the sum of the mass function value must be 1 and that the maximum value of the distribution function is 1 satisfying the properties mentioned previously.

If we extend the idea that a random variable is now a function of time as well, then we can define a stochastic process as discussed in Chapter 2. More formerly, a random or *stochastic process* is a two-dimensional function of $t$ and $\omega$:

$$X(t, \omega), \quad \omega \epsilon \Omega, \quad t \epsilon T \tag{A.6}$$

where T is a set of index parameters (continuous or discrete) and $\Omega$ is the sample space.

We list some of the major theorems in probability theory and refer the reader to more detailed texts [1], [2].

| | |
|---|---|
| Univariate | $\Pr(X) = \mathrm{P}_X(x)$ |
| Bivariate | $\Pr(X, Y) = \mathrm{P}_{XY}(x, y)$ |
| Marginal | $\Pr(X) = \sum_y \mathrm{P}_{XY}(x, y)$ |
| Independent | $\Pr(X, Y) = \mathrm{P}_X(x) \times \mathrm{P}_Y(y)$ |
| Conditional | $\Pr(X\|Y) = \mathrm{P}_{XY}(x, y)/\mathrm{P}_Y(y)$ |
| Chain rule | $\Pr(X, Y, Z) = \Pr(X\|Y, Z) \times \Pr(Y\|Z) \times \Pr(Z)$ |

For a random variable, we can define basic statistics in terms of the probability mass function. The *expected value* or *mean* of a random variable $X$, is given by

$$m_x = E\{X\}$$

and is considered the typical or representative value of a given set of data. For this reason the mean is called a measure of central tendency. The degree to which numerical data tend to spread about the expected value is usually measured by the *variance* or equivalently, *autocovariance* given by

$$R_{xx} = E\{(X - m_x)^2\}$$

The basic statistical measures are called *ensemble statistics* because they are measured across the ensemble ($i = 1, 2, ...$) of data values; that is, the expectation is always assumed over an ensemble of realizations. We summarize these statistics in terms of their mass function as follows:

| | |
|---|---|
| Expected value | $m_x = E\{X\} = \sum_i X_i \mathrm{P}_X(x_i)$ |
| $N$th-moment | $E\{X^n\} = \sum_i X_i^n \mathrm{P}_X(x_i)$ |
| $N$th-moment about mean | $E\{(X - m_x)^n\} = \sum_i (X_i - m_x)^n \mathrm{P}_X(x_i)$ |
| Mean Squared ($N = 2$) | $E\{X^2\} = \sum_i X_i^2 \mathrm{P}_X(x_i)$ |
| Variance | $R_{xx} = E\left\{(X_i - m_x)^2\right\} = \sum_i (X_i - m_x)^2 \mathrm{P}_X(x_i)$ |
| Covariance | $R_{xy} = E\left\{(X_i - m_x)(Y_j - m_y)\right\}$ |
| Standard deviation | $\sigma_{xx} = \sqrt{R_{xx}}$ |
| Conditional mean | $E\{X\|Y\} = \sum_i X_i \mathrm{P}(X_i\|Y)$ |
| Joint conditional mean | $E\{X\|Y, Z\} = \sum_i X_i \mathrm{P}(X_i\|Y, Z)$ |
| Conditional variance | $R_{x\|y} = E\{(X - E\{X\|Y\})^2\|Y\}$ |

These basic statistics possess various properties that enable them to be useful for analyzing operations on random variables, some of the more important[1] are as follows:

[1]Recall that independence states that the joint mass function can be factored $\Pr(x, y) = \Pr(x) \times \Pr(y)$, which leads to these properties.

| | |
|---|---|
| Linearity | $E\{ax + b\} = aE\{x\} + b = am_x + b$ |
| Independence | $E\{xy\} = E\{x\}E\{y\}$ |
| Variance | $R_{xx}(ax + b) = a^2 R_{xx}$ |
| Covariance | |
| Uncorrelated | $E\{xy\} = E\{x\}E\{y\} \quad \{R_{xy} = 0\}$ |
| Orthogonal | $E\{xy\} = 0$ |

Note that the expected value operation implies that for stochastic processes these basic statistics are calculated *across* the ensemble. For example, if we want to calculate the mean of a process, that is,

$$m_x(t) = E\{X(t, \omega_i) = x_i(t)\}$$

we simply take the values of $t = 0, 1, \ldots$ and calculate the mean for each value of time across ($i = 1, 2, ...$) the ensemble. Dealing with stochastic processes is similar to dealing with random variables except that we must account for the time indexes (see Chapter 2 for more details).

Next let us define some concepts about the probabilistic information contained in a random variable. We define the (self) *information* contained in the occurrence of the random variable $X(\omega_i) = x_i$, as

$$I(x_i) = -\log_b \mathrm{P}_X(x_i) \tag{A.7}$$

where $b$ is the base of the logarithm which results in different units for information measures (base = 2 $\rightarrow$ bits) and the *entropy* or *average information* of $X(\omega_i)$ as

$$H(x_i) = -E\{I(x_i)\} = \sum_i \mathrm{P}_X(x_i) \log_b \mathrm{P}_X(x_i) \tag{A.8}$$

Consider the case where there is more than one random variable. Then we define the *joint* mass and distribution functions of an $N$-dimensional random variable as

$$\mathrm{P}_X(x_1, \ldots, x_N), \quad \mathrm{F}_X(x_1, \ldots, x_N)$$

All the basic statistical definitions remain as before, except that we replace the scalar with the joint functions. Clearly, if we think of a stochastic process as a sequence of ordered random variables, then we are dealing with joint probability functions, that is, a collection of time-indexed random variables. Suppose that we have two random variables, $x_1$, and $x_2$, and we know that the latter has already assumed a particular value. We can define the *conditional* probability mass function of $x_1$ given that $X(\omega_2) = x_2$ has occurred by

$$\Pr(x_1 \mid x_2) := \mathrm{P}_X(X(\omega_1) \mid X(\omega_2) = x_2) \tag{A.9}$$

It can be shown from basic probabilistic axioms (see Papoulis [1]) that

$$\Pr(x_1 \mid x_2) = \frac{\Pr(x_1, x_2)}{\Pr(x_2)} \tag{A.10}$$

Note also that this expression can also be written as

$$\Pr(x_1, x_2) = \Pr(x_2 \mid x_1)\Pr(x_1) \tag{A.11}$$

Substituting this equation into Eq. (A.10) gives *Bayes's rule*, that is,

$$\Pr(x_1 \mid x_2) = \Pr(x_2 \mid x_1)\frac{\Pr(x_1)}{\Pr(x_2)} \tag{A.12}$$

If we use the definition of joint mass function and substitute into the previous definitions, then we can obtain the *probabilistic chain rule* ([1], [2], [3]),

$$\Pr(x_1, \ldots, x_N) = \Pr(x_1 \mid x_2, \ldots, x_N)\Pr(x_2 \mid x_3, \ldots, x_N) \cdots \Pr(x_{N-1} \mid x_N)\Pr(x_N) \tag{A.13}$$

Along with these definitions follows the idea of conditional expectation, that is,

$$E\{x_i \mid x_j\} = \sum_i X_i \Pr(x_i \mid x_j) \tag{A.14}$$

With the conditional expectation defined, we list some of their basic properties:

1. $E_x\{X|Y\} = E\{X\}$ if $X$ and $Y$ are independent
2. $E\{X\} = E_y\{E\{X|Y\}\}$
3. $E_x\{g(y)X|Y\} = g(y)E\{X|Y\}$
4. $E_{x,y}\{g(Y)X\} = E_y\{g(Y)E\{X|Y\}\}$
5. $E_x\{c|Y\} = c$
6. $E_x\{g(Y)|Y\} = g(Y)$
7. $E_{x,y}\{cX + dY|Z\} = cE\{X|Z\} + dE\{Y|Z\}$

The concepts of information and entropy can also be extended to the case of more than one random variable. We define the *mutual information* between two random variables, $x_i$ and $x_j$ as

$$I(x_i; x_j) = \log_b \frac{\mathrm{P}_X(x_i \mid x_j)}{\mathrm{P}_X(x_i)} \tag{A.15}$$

and the *average mutual information* between $X(\omega_i)$ and $X(\omega_j)$ as

$$I(X_i; X_j) = E_{x_i x_j}\{I(x_i, x_j)\} = \sum_i \sum_j \mathrm{P}_X(x_i, x_j) I(x_i, x_j) \tag{A.16}$$

This leads to the definition of *joint entropy* as

$$H(X_i; X_j) = -\sum_i \sum_j \mathrm{P}_X(x_i, x_j) \log_b \mathrm{P}_X(x_i, x_j) \tag{A.17}$$

This completes the section on probability theory, next let us consider an important multivariable distribution and its properties.

## A.2 GAUSSIAN RANDOM VECTORS

In this section we consider the multivariable gaussian distribution used heavily in this text to characterize gaussian random vectors, that is, $\mathbf{z} \sim \mathcal{N}(\mathbf{m}_z, \mathbf{R}_{zz})$ where $\mathbf{z} \in \mathcal{R}^{N_z \times 1}$ and defined by

$$\Pr(\mathbf{z}) = (2\pi)^{-N_z/2} |\mathbf{R}_{zz}|^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{z} - \mathbf{m}_z)' \mathbf{R}_{zz}^{-1} (\mathbf{z} - \mathbf{m}_z)\right) \tag{A.18}$$

where the vector mean and covariance are defined by

$$\mathbf{m}_z := E\{\mathbf{z}\} \quad \text{and} \quad \mathbf{R}_{zz} = \mathrm{Cov}(\mathbf{z}) := E\{(\mathbf{z} - \mathbf{m}_z)(\mathbf{z} - \mathbf{m}_z)'\}$$

Certain properties of the gaussian vectors are useful:

- *Linear transformation.* Linear transformations of gaussian variables are gaussian; that is, if $\mathbf{z} \sim \mathcal{N}(\mathbf{m}_z, \mathbf{R}_{zz})$ and $\mathbf{y} = \mathbf{A}\mathbf{z} + \mathbf{b}$, then

$$\mathbf{y} \sim \mathcal{N}(\mathbf{A}\mathbf{m}_z + \mathbf{b}, \mathbf{A}\mathbf{R}_{zz}\mathbf{A}') \tag{A.19}$$

- *Uncorrelated gaussian vectors.* Uncorrelated gaussian vectors are independent.
- *Sums of gaussian variables.* Sums of independent gaussian vectors yield gaussian distributed vectors with mean and variance equal to the sums of the respective means and variances.
- *Conditional gaussian vectors.* Conditional gaussian vectors are gaussian distributed; that is, if $\mathbf{x}$ and $\mathbf{y}$ are jointly gaussian, with

$$\mathbf{m}_z = E\left\{\begin{array}{c}\mathbf{x}\\ \mathbf{y}\end{array}\right\} = \left[\begin{array}{c}\mathbf{m}_x\\ \mathbf{m}_y\end{array}\right]$$

$$\mathbf{R}_{zz} = \quad \mathrm{cov}(\mathbf{z}) = \left[\begin{array}{cc}\mathbf{R}_{xx} & \mathbf{R}_{xy}\\ \mathbf{R}_{yx} & \mathbf{R}_{yy}\end{array}\right]$$

then the conditional distribution for $\mathbf{x}$ and $\mathbf{y}$ is also gaussian with conditional mean and covariance given by

$$\mathbf{m}_{\mathbf{x}|\mathbf{y}} = \mathbf{m}_x + \mathbf{R}_{xy}\mathbf{R}_{yy}^{-1}(\mathbf{y} - \mathbf{m}_y)$$
$$\mathbf{R}_{\mathbf{x}|\mathbf{y}} = \mathbf{R}_{xx} - \mathbf{R}_{xy}\mathbf{R}_{yy}^{-1}\mathbf{R}_{yx}$$

and the vectors $\mathbf{x} - E\{\mathbf{x}|\mathbf{y}\}$ and $\mathbf{y}$ are independent.

- *Gaussian conditional means.* Let $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ be jointly distributed gaussian random vectors, and let $\mathbf{y}$ and $\mathbf{z}$ be *independent*. Then

$$E\{\mathbf{x}|\mathbf{y}, \mathbf{z}\} = E\{\mathbf{x}|\mathbf{y}\} + E\{\mathbf{y}|\mathbf{z}\} - \mathbf{m}_x$$

## A.3 UNCORRELATED TRANSFORMATION: GAUSSIAN RANDOM VECTORS

Suppose that we have a gaussian random vector, $\mathbf{x} \sim \mathcal{N}(\mathbf{m}_x, \mathbf{R}_{xx})$ and we would like to transform it to a normalized gaussian random vector with the mean, $\mathbf{m}_x$, removed so that, $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$. Assume that the mean has been removed ($\mathbf{z} \rightarrow \mathbf{z} - \mathbf{m}_x$), then there exists a nonsingular transformation, $\mathbf{T}$, such that $\mathbf{z} = \mathbf{T}\mathbf{x}$. Therefore

$$\mathbf{R}_{zz} = \text{Cov}(\mathbf{z}) = \text{Cov}\left((\mathbf{T}\mathbf{x})(\mathbf{T}\mathbf{x})'\right) = \mathbf{T}\mathbf{R}_{xx}\mathbf{T}' = \mathbf{I} \tag{A.20}$$

Thus we must find a transformation that satisfies the relation

$$\mathbf{R}_{zz} = \mathbf{I} = \mathbf{T}\mathbf{R}_{xx}\mathbf{T}' \tag{A.21}$$

Since $\mathbf{R}_{xx}$ is a positive semidefinite, symmetric matrix, it can always be factored into matrix square roots ($\mathbf{R} = \mathbf{U}\mathbf{U}' = \mathbf{R}_{xx}^{1/2}\mathbf{R}_{xx}^{T/2}$ ) using a Cholesky decomposition [4]. Therefore Eq. (A.21) implies

$$\mathbf{R}_{zz} = \mathbf{I} = (\mathbf{T}\mathbf{U})\,\mathbf{U}'\mathbf{T}' \tag{A.22}$$

or simply that

$$\mathbf{T} = \mathbf{U}^{-1} = \mathbf{R}_{xx}^{-1/2} \qquad \text{(Inverse matrix square root)} \tag{A.23}$$

Therefore

$$\mathbf{R}_{zz} = \left(\mathbf{R}_{xx}^{-1/2}\mathbf{U}\right)\mathbf{U}'\mathbf{R}_{xx}^{-T/2} = \mathbf{R}_{xx}^{-1/2}\mathbf{R}_{xx}\mathbf{R}_{xx}^{-T/2} = \mathbf{I} \tag{A.24}$$

the desired result.

This discussion completes the introductory concepts of probability and random variables which is extended to include stochastic processes in Chapter 2 and throughout the text.

## REFERENCES

1. A. Papoulis, *Probability, Random Variables and Stochastic Processes*, New York: McGraw-Hill, 1991.
2. R. Hogg and A. Craig, *Introduction to Mathematical Statistics*, New York: Macmillan, 1970.
3. A. Jazwinski, *Stochastic Processes and Filtering Theory*, New York: Academic Press, 1970.
4. G. Bierman, *Factorization Methods for Discrete Sequential Estimation*, New York: Academic Press, 1977.

# APPENDIX B

# SEQUENTIAL *MBP* AND *UD*-FACTORIZATION

## B.1 SEQUENTIAL *MBP*

In this appendix we briefly review the preferred method of implementing *MBP* algorithms using the state-space models of this text, that is, we discuss the sequential processing of vector measurements, its implication in terms of the optimal state estimate and then outline the *UD* algorithm developed by Bierman [1]. *Sequential processing* refers to the technique of processing the measurement vector, $\mathbf{y}(t)$, *one component* at a time, that is, $y_i(t)$; for $i = 1, \ldots, N_y$. The major reason for processing data this way is that besides avoiding the inversion of the $N_y \times N_y$ innovations covariance matrix resulting in up to 50% computational savings, it provides a numerically stable solution, which is critical especially when dealing with large-scale systems. The only restriction is that the measurement noise must be uncorrelated or equivalently $R_{vv}$ *must* be diagonal. The latter does not present a problem, since $R_{vv} \geq 0$, it can *always* be transformed to an uncorrelated covariance using matrix square roots (e.g., Cholesky decomposition) as shown in Appendix A. The *MBP* algorithm (see Chapter 5), when implemented sequentially, performs an orthogonal decomposition of the measurement or equivalently, the innovations sequence such that it becomes "time uncorrelated," or equivalently $R_{ee}$ is diagonal. In order to see this, we first develop the algorithm under the diagonal innovations covariance assumption and then show the structure of the transformation, $\mathbf{T}$, that accomplishes this diagonalization, *implicitly* during the sequential processing.

*Model-Based Signal Processing*, by James V. Candy

From this perspective we can consider sequential processing as performing a transformation of coordinates in the measurement or output space from time correlated to uncorrelated measurements or innovations.

Sequential processing occurs only in the correction portion of the *MBP* algorithm (see Table 5.1). The correction is accomplished (using simplified notation) by

$$\begin{aligned} e(t) &= y(t) - C\hat{x}(t|t-1) \\ R_{ee} &= C\tilde{P}(t|t-1)C' + R_{vv} \\ K &= \tilde{P}(t|t-1)C'R_{ee}^{-1} \\ \hat{x}(t|t) &= \hat{x}(t|t-1) + Ke(t) \\ \tilde{P}(t|t) &= (I - KC)\tilde{P}(t|t-1) \end{aligned}$$

Let us partition the $N_y \times N_x$ measurement matrix $C$ into $1 \times N_x$ row vectors, $\mathbf{c}_i'$ and the $N_x \times N_y$ gain matrix into $N_x \times 1$ column vectors, $\mathbf{k}_i$, so that

$$C = \begin{bmatrix} \mathbf{c}_1' \\ \vdots \\ \mathbf{c}_{N_y}' \end{bmatrix} \quad \text{and} \quad K = \begin{bmatrix} \mathbf{k}_1 & \cdots & \mathbf{k}_{N_y} \end{bmatrix}, \quad \text{where } \mathbf{c}, \mathbf{k} \in \mathcal{R}^{N_x \times 1} \tag{B.1}$$

and rewrite the "correction" equations above component-wise, assuming that $R_{vv}$ and $R_{ee}$ are diagonal. Therefore we have

$$\begin{bmatrix} e_1(t) \\ \vdots \\ e_{Ny}(t) \end{bmatrix} = \begin{bmatrix} y_1(t) \\ \vdots \\ y_{Ny}(t) \end{bmatrix} - \begin{bmatrix} \mathbf{c}_1' \\ \vdots \\ \mathbf{c}_{N_y}' \end{bmatrix} \hat{x}(t|t-1)$$

$$\begin{bmatrix} R_{ee}(1) & & O \\ & \ddots & \\ O & & R_{ee}(N_y) \end{bmatrix} = \begin{bmatrix} \mathbf{c}_1' \\ \vdots \\ \mathbf{c}_{N_y}' \end{bmatrix} \tilde{P}(t|t-1)[\mathbf{c}_1 \ \cdots \ \mathbf{c}_{N_y}] + \begin{bmatrix} R_{vv}(1) & & O \\ & \ddots & \\ O & & R_{vv}(N_y) \end{bmatrix}$$

$$[\mathbf{k}_1 \ \cdots \ \mathbf{k}_{N_y}] = \tilde{P}(t|t-1)[\mathbf{c}_1 \ \cdots \ \mathbf{c}_{N_y}] \begin{bmatrix} 1/R_{ee}(1) & & O \\ & \ddots & \\ O & & 1/R_{ee}(N_y) \end{bmatrix} \tag{B.2}$$

$$\hat{x}(t|t) = \hat{x}(t|t-1) + [\mathbf{k}_1 \ \cdots \ \mathbf{k}_{N_y}] \begin{bmatrix} e_1(t) \\ \vdots \\ e_{Ny}(t) \end{bmatrix}$$

$$\tilde{P}(t|t) = \left( \mathbf{I} - [\mathbf{k}_1 \ \cdots \ \mathbf{k}_{N_y}] \begin{bmatrix} \mathbf{c}_1'(t) \\ \vdots \\ \mathbf{c}_{Ny}'(t) \end{bmatrix} \right) \tilde{P}(t|t-1)$$

If we extract the $i$th equation from each relation in Eq. (B.2) and sequentially process the measurement from $i = 1, \ldots, N_y$ with

$$\hat{\mathbf{x}}_i = \begin{cases} \hat{x}(t|t-1), & i = 0 \\ \\ \hat{x}(t|t), & i = N_y \end{cases}$$

$$\tilde{\mathbf{P}}_i = \begin{cases} \tilde{P}(t|t-1), & i = 0 \\ \\ \tilde{P}(t|t), & i = N_y \end{cases}$$

we obtain the sequential implementation.

Here the state estimate takes on different values during the sequential processing. *Prior* to processing ($i = 0$), the state is the predicted estimate, $\hat{\mathbf{x}}(t|t-1)$, *during* the processing the state takes on intermediate values, $\hat{\mathbf{x}}_{i-1}$ and *after* ($i = N_y$) processing the "corrected" state, $\hat{\mathbf{x}}(t|t)$, evolves from the final calculation. This form is very efficient when combined with the *UD*-factorization techniques [1]. By selecting the $i$th row of these relations, we obtain the sequential implementation of the *MBP* algorithm shown below.

### *Sequential Measurement Processing for MBP*

$$\begin{aligned} e_i &= y_i - \mathbf{c}_i' \hat{\mathbf{x}}_{i-1} \\ R_{ee}(i) &= \mathbf{c}_i' \tilde{P}_{i-1} \mathbf{c}_i + R_{vv}(i) \\ \mathbf{k}_i &= \frac{\tilde{P}_{i-1} \mathbf{c}_i}{R_{ee}(i)}, \qquad i = 1, \ldots, N_y \\ \hat{\mathbf{x}}_i &= \hat{\mathbf{x}}_{i-1} + \mathbf{k}_i e_i \\ \tilde{P}_i &= (I - \mathbf{k}_i \mathbf{c}_i') \tilde{P}_{i-1} \end{aligned} \tag{B.3}$$

It is interesting to note that the transformation, $\mathbf{T}$, implicit to the sequential algorithm follows from the orthogonalization property of the optimal *MBP* (see Chapter 5). Recall from the derivation that, by construction from the Gram-Schmidt procedure [3], the innovations is the uncorrelated process that evolves such that

$$\nu := \mathbf{T}\mathbf{e} \quad \text{and} \quad R_{\nu\nu} = \mathbf{T} R_{ee} \mathbf{T}'$$

for $R_{\nu\nu} = \text{diag}[R_{11}^{\nu} \cdots R_{N_yN_y}^{\nu}]$ and $R_{ij} = \text{cov}(e_i \nu_j)$. Therefore

$$\mathbf{T} = \begin{bmatrix} 1 & & O \\ R_{21}(R_{11}^{\nu})^{-1} & 1 & \\ \vdots & \vdots & \ddots \\ R_{N_y1}(R_{11}^{\nu})^{-1} & \cdots & R_{N_yN_y-1}(R_{N_yN_y}^{\nu})^{-1} \quad 1 \end{bmatrix}$$

Thus we can think of sequential measurement processing as a special case of orthogonal decomposition where the *MBP* performs a Gram-Schmidt procedure on each component of the measurement or equivalently the innovation. Next we see how this technique can be used in conjunction with the square root methods.

## B.2 *UD*-FACTORIZATION ALGORITHM FOR *MBP*

In this section we develop the *UD*-factorization algorithm for correcting the error covariance matrix in the *MBP* algorithm. The approach requires the sequential processing implementation of the previous section and therefore assumes the measurement (innovations) is uncorrelated in time. The underlying idea in covariance factorization techniques evolves from the early work on square root processors [1]. Since the covariance matrix is positive semi-definite it always admits a factorization or square root (e.g., Cholesky decomposition) of the form

$$\tilde{P} = SS' \tag{B.4}$$

Square root processing by its very nature forces $\tilde{P}$ to be positive semidefinite by the preceding constraint, thereby eliminating many numerical problems caused by roundoff and truncation in the correction and prediction equations [1]. The *UD*-factorization method is based on the fact that

$$\tilde{P}(t-1) = U(t-1)D(t-1)U'(t-1) \tag{B.5}$$

where $U$ is upper triangular with unity diagonals and $D$ is diagonal. We would like the error covariance correction equation

$$\tilde{P}(t) = [I - K(t)C(t)]\tilde{P}(t-1) \tag{B.6}$$

to be placed in a *UD*-factored form. Before we develop the equations, recall the sequential processing algorithm of Eq. (B.4). The error covariance equation of Eq. (B.6) can be rewritten in *UD*-form as

$$\begin{aligned} U(t)D(t)U'(t) = (&U(t-1)D(t-1)U'(t-1) \\ &- \mathbf{k}_i\mathbf{c}_i')[U(t-1)D(t-1)U'(t-1)] \end{aligned} \tag{B.7}$$

where

$$\mathbf{k}_i = \frac{\tilde{P}(t-1)\mathbf{c}_i}{R_{ee}(i)} = \frac{U(t-1)D(t-1)U'(t-1)\mathbf{c}_i}{R_{ee}(i)}$$

Substituting for $\mathbf{k}_i$ in Eq. (B.7) gives

$$U(t)D(t)U'(t) = U(t-1)D(t-1)U'(t-1) - \frac{U(t-1)D(t-1)U'(t-1)\mathbf{c}_i\mathbf{c}_i'U(t-1)D(t-1)U'(t-1)}{R_{ee}(i)}$$

Then factoring out $U(t-1)$ on the left and $U'(t-1)$ on the right gives

$$U(t)D(t)U'(t) = U(t-1) \times \left[D(t-1) - \frac{D(t-1)U'(t-1)\mathbf{c}_i\mathbf{c}_i'U(t-1)D(t-1)}{R_{ee}(i)}\right]U'(t-1) \tag{B.8}$$

Define $\mathbf{s} := D(t-1)U'(t-1)\mathbf{c}_i$. Then substituting into Eq. (B.8) gives

$$U(t)D(t)U'(t) = U(t-1)\left[D(t-1) - \frac{\mathbf{s}\mathbf{s}'}{R_{ee}(i)}\right]U'(t-1) \tag{B.9}$$

We also re-define

$$\mathbf{s} := D(t-1)\mathbf{h} = [s(1)\cdots s(N_x)]'$$
$$\text{for} \quad \mathbf{h} := U'(t-1)\mathbf{c}_i = [h(1)\cdots h(N_x)]'$$

Using this notation, we have the Bierman *UD*-factorization algorithm as follows:

### *UD-Factorization Algorithm for MBP*

For $j = 2, \ldots, N_x$ recursively,

$$\begin{aligned}
\alpha_j &= \alpha_{j-1} + s(j)h(j) \\
d_j(t) &= \frac{d_j(t-1)\alpha_{j-1}}{\alpha_j} \\
\mathbf{k}_j &= \mathbf{s}(j) \\
\tilde{P}_j &= \frac{-h(j)}{\alpha_{j-1}} \\
U_{ij}(t) &= U_{ij}(t-1) + \mathbf{k}_i\tilde{P}_j \qquad \text{for} \quad i = 1, \ldots, j-1 \\
\mathbf{k}_{i+1} &= \mathbf{k}_i + U_{ij}(t-1)s(j) \\
R_{ee}(i) &= \alpha_{N_x} \\
\mathbf{k}_{i+1} &= \frac{\mathbf{k}_{i+1}}{R_{ee}(i)}
\end{aligned} \tag{B.10}$$

for $D(t) = \text{diag}[d_1(t) \cdots d_{N_x}(t)]$ and $U(t) = [U_{ij}(t)]$ with initial conditions given by

$$\begin{aligned} \alpha_1 &= s(1)h(1) + R_{vv}(1) \\ d_1(t) &= \frac{d_1(t-1)R_{vv}(1)}{\alpha_1} \\ \mathbf{k}_1 &= \mathbf{s}(1) \end{aligned}$$

This completes the development of the *UD*-factorization algorithm applied to the error covariance correction equation. Note well that Bierman [1] has also shown how to *UD*-factor the prediction error covariance as well. This is the *standard* implementation of the *MBP* algorithm applied to data. It can also applied in nonlinear filtering (e.g., *XMBP*) in both prediction and correction equations to provide a numerically stable solution to the model-based processing problem [2]. In a sense the simple *MBP* algorithm of Table 5.1 has been transformed to this complex, but numerically stable, implementation. This situation is analogous to performing a brute force gaussian elimination inversion of a matrix or replacing that unstable inversion with the stable singular value decomposition. This is one of the primary reasons for the evolution of special software packages (e.g., *SSPACK_PC* [4]) using *UD*-methods.

## REFERENCES

1. G. Bierman, *Factorization Methods for Discrete Sequential Estimation*, New York: Academic Press, 1977.
2. A. Jazwinski, *Stochastic Processes and Filtering Theory*, New York: Academic Press, 1970.
3. D. Gavel, Personal communication on sequential measurement processing, 1982;
4. J. Candy and P. Candy, "SSPACK_PC: A model-based signal processing package on personal computers," *DSP Applications*, **2** (3), 33–42 1993 (see **http://www.technisoft.net** for more details).

# APPENDIX C

# *SSPACK_PC*: AN INTERACTIVE MODEL-BASED PROCESSING SOFTWARE PACKAGE

The design and analysis of model-based processors is often a formidable task, especially when models are complex. In this appendix we introduce the Model-Based Signal Processing/State-Space Systems Software Package (*SSPACK_PC*), an interactive software package for the analysis, design, and display of model-based processors and state-space systems on personal computers [1]. The package contains algorithms for model-based signal processing simulation, estimation, and identification integrated into an easy-to-use package. It enables the user to quickly access model-based signal processing algorithms and analyze their performance. A preprocessor designed to simplify model specifications allows even complex systems to easily be defined and processed. *SSPACK_PC* enables easy interfacing to powerful model-based algorithms with display and analysis capability.

## C.1 INTRODUCTION

*SSPACK_PC* offers an integrated approach to various model-based signal processing problems. It eases the burden of a priori knowledge by utilizing a graphical user interface (GUI)-oriented Supervisor so the casual or new user can immediately begin operation. The package is designed to enable the serious user to run individual processors and algorithms separately as well as interface his or her own particular algorithms. *SSPACK_PC* utilizes step-by-step procedures that enable the user to monitor various operations in full detail.

*Model-Based Signal Processing*, by James V. Candy

The package consists of a Supervisor, which controls the operation of the software, preprocessors and postprocessors, and sets of individual algorithms (or modules) that communicate using disk files. It controls the package by enabling the user to perform various operations ranging from specifying a problem title to preparing and running a particular problem. The user need not leave the Supervisor to accomplish any of the required operations in *SSPACK_PC*. However, since it exists with the *MATLAB* environment all of the usual *MATLAB* are readily available at any time for supplementary analysis.

The processors aid the user in entering the problem of interest as well as displaying the results. The state-space preprocessor (SSPREP) algorithm program aids the user in preparing files for the individual algorithm programs, while the state-space postprocessor (SSPOST) program displays and analyzes the output from the algorithms. SSPREP prompts the user with a series of questions in a GUI format. The initial GUI lets the user choose the desired algorithm. Subsequent GUIs help specify all the necessary parameters for the selected algorithm. SSPOST is an interactive, command-driven processor that is embellished in the package, but may be executed independently as well. It is designed to interpret the output of the various *SSPACK_PC* algorithms and display the results.

Besides the Supervisor and processors, *SSPACK_PC* consists of the following modules: a linear discrete-time state estimator (SSDEST) and simulator (SSDSIM), a linear discrete-time varying state estimator (SSTEST) and simulator (SSTSIM), a linear discrete-time state estimator using innovations models (SSIEST) and simulator (SSISIM), a linearized discrete-time state estimator (SSLZEST), a nonlinear discrete-time state estimator (SSNEST) and simulator (SSNSIM), continuous-to-discrete-time conversion (SSCTOD), innovations to Gauss-Markov conversion (INVTOGM), a Gauss-Markov to innovations conversion (GMTOINV) and a discrete-time parameter estimator/identifier (SSDID). All the estimation algorithms are based on the Kalman filter/estimator [2] implemented using efficient *UD*-factorized matrix techniques [3].

## C.2 SUPERVISOR

The focus of *SSPACK_PC* interactivity lies in its systems Supervisor. It is designed to enable the user to perform all the necessary operations from simulation to performance analysis without the burden of understanding the detailed linkage between individual modules that comprise package. Owing to its modular design, the user can interface his or her own particular algorithms to the package easily.

The control flow in the *SSPACK_PC* Supervisor is best described by the diagram shown in Figure C.1. The Supervisor enables the user to select the desired operation (simulation, estimation, or identification) through a GUI format and then perform the various steps necessary to accomplish it. Once selected, the user's choices are identical for each operation; that is, RENAME the problem title, CREATE the specification file for the particular algorithm, MODIFY the problem models, EDIT the specification file, RUN the selected algorithm, and ANALYZE and DISPLAY the

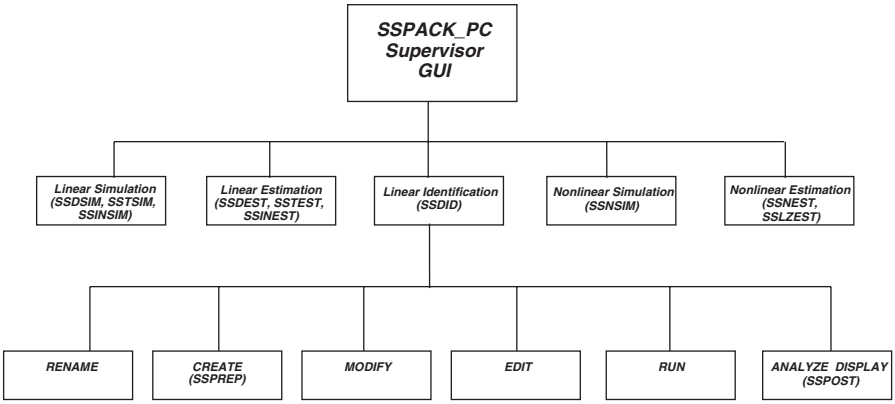



**Figure C.1.** *SSPACK_PC* interaction diagram.

results. Most of these operations are performed by specific *SSPACK_PC* algorithms discussed in detail in the user's manual [3]. Note that for various algorithms some GUI choices operate differently. For example, the MODIFY selection for the linear-discrete simulation enables the user to convert from a continuous-time model to a discrete-time model, while this selection for the nonlinear-discrete simulation actually enables the user to enter in new set of nonlinear difference equations. These tasks are all accomplished within the *SSPACK_PC* Supervisor, thus simplifying the design procedure.

The GUI format of the Supervisor relieves much of the typing burden from the user. For example, consider the *SSPACK_PC* interaction for a linear discrete-time simulation. With the choice (CREATE) the user begins to prepare a problem specification file for the particular problem under investigation. Through a series of questions (from SSPREP), the model parameters and problem parameters are written into a file for eventual algorithm use.

## C.3 PREPROCESSOR

The preprocessor creates a problem specification file incorporating all of the necessary information to execute the desired program (SSDSIM in this example). This file is a simple text file in free format, making it easy for the user to read, print, or edit. Once the file is created, alterations in problem parameters are easily modified by selecting the EDIT operation, in which the actual problem specification file is displayed for editing. Note that the preprocessor uses a question-and-answer format to enter parameters and a GUI processor to alter these parameters after a particular category or model has been completely entered.

Once the specification file is completed, the user selects the RUN option of the Supervisor to actually perform the simulation using SSDSIM. After optionally echoing the problem specifications back to the user's terminal, SSDSIM executes

the problem and creates a postprocessor file for display and performance analysis. Following the execution of SSDSIM, the user selects the ANALYZE/DISPLAY option in the Supervisor menu, which executes the postprocessor.

## C.4 POSTPROCESSOR

SSPOST is a command-driven processor that displays the outputs of the various simulation and estimation algorithms in *SSPACK_PC* and performs statistical tests to evaluate the performance of each particular algorithm. It reads commands and operates on a postprocessor (*pp*) file created by other *SSPACK_PC* routines. The user issues commands to SSPOST interactively from a terminal or from a command file for automatic execution. The commands are directives to the SSPOST program for reading data, setting scales, producing plots, and so forth. It also has the capability of selecting single or multiple plots on the display device by choosing various "viewports" available in *MATLAB* (e.g., four plots per screen). Along with this viewporting capability, SSPOST also enables the user to display data from multiple ppfiles as families of curves. This feature is particularly useful when attempting to analyze the effect of a particular adjustment on the performance of an algorithm, such as tuning an algorithm. SSPOST also offers the user an on-line help package that includes a description of each command for easy reference.

The following list summarizes the capability and important features of SSPOST:

1. *Displays* the time history of various *SSPACK_PC* algorithm variables.
2. *Tests* statistically the performance of *SSPACK_PC* algorithms.
3. *Executes* interactively from the keyboard or command files.
4. Displays *viewport*.
5. *Helps* through an on-line list of commands.
6. *Plots* multiple ppfiles for comparison.

All the SSPOST commands are succinctly summarized in Table C.1.

## C.5 ALGORITHMS

*SSPACK_PC* is primarily aimed at discrete-time model-based processing, since the majority of real-time applications are discrete-time problems implemented on personal computers. Problems formulated in continuous time are converted to discrete-time and then executed by the various algorithms.

The conversion from continuous to discrete time is accomplished by first using the Paynter algorithm to determine the number of terms required in the matrix exponential series and then calculating the series using numerical methods [3].

There are four state-space simulators available in *SSPACK_PC* three linear and a nonlinear algorithm. The simulators are stochastic and produce Gauss-Markov, *time-varying* Gauss-Markov, Innovations and approximately Gauss-Markov outputs

**Table C.1. *SSPACK_PC*: Postprocessor Command Summary**

| | | |
|---|---|---|
| | Display commands | |
| EP | I J | Plot innovations sequence |
| GK | I J K L | Plot gain |
| PC | I J K L | Plot corrected covariance |
| PP | I J K L | Plot predicted covariance |
| PS | I J K L | Plot simulated covariance |
| RE | I J K L | Plot innovations covariance |
| RZ | I J K L | Plot measurement covariance |
| U | I J | Plot forcing function |
| XC | I J | Plot corrected state |
| XP | I J | Plot predicted state |
| XT | I J | Plot true (mean) state |
| Z | I J | Plot measurement |
| ZEST | I J | Plot predicted measurement |
| ZT | I J | Plot the true (mean) measurement |
| | Performance commands | |
| PSDEP | I J ISTART | Plot innovations power spectrum |
| WTEST | I J ISTART | Plot innovations whiteness/zero-mean |
| WSSR | N | Plot the WSSR with window *N* |
| XERR | I J | Plot estimation error |
| XS | I J | Plot simulated state |
| ZS | I J | Plot simulated measurement |
| | Input-output commands | |
| COMFILE | NAME | Read next command from file NAME |
| END or EXIT | | User is done–terminate |
| | Parameter commands | |
| HELP | | Obtain on-line help |

for the state and noisy measurements ([4], [5], [6]. The associated statistics are also calculated (mean and variance) and are used to determine confidence limits as well as evaluate the performance of the algorithm. Of course, deterministic state-space models are special cases of these stochastic models and are easily simulated as well.

Four state-space estimators are available in *SSPACK_PC*. The linear discrete-time estimator utilizes the *UD*-factorized matrix or so called *UD*-form of the recursive *MBP* estimation algorithm, a superior numerical technique for small word-length machines or large-scale problems [2]. The linear time-varying estimator also uses the *UD*-form. The innovations processor is basically the steady-state form of the

linear estimator. The linearized discrete-time estimator *LZ-MBP* is available especially when a reference trajectory can be predefined for the problem. The nonlinear discrete-time algorithm is also the *UD*-form of the *XMBP* algorithm as well as the *IX-MBP*. Both algorithms produce postprocessor files with estimates as well as associated statistics to enable the overall statistical performance to be analyzed and displayed. For instance, such features as whiteness testing, spectral estimation, tracking errors, etc. are provided as some of the available analysis procedures (see Chapter 5).

There are also two identification algorithms [7] available. The first is a linear *MBP* or, equivalently, recursive LS identifier capable of performing parameter estimation for a *scalar* discrete-time transfer function or state-space model in observer canonical form [5]. Nonlinear or multivariable models can be identified using the parametrically adaptive *XMBP* discussed in Chapter 8 by augmenting the unknown parameters into the state-space models.

All the algorithms communicate with the preprocessors and postprocessors through text files. These files are readable by the user. This completes discussion

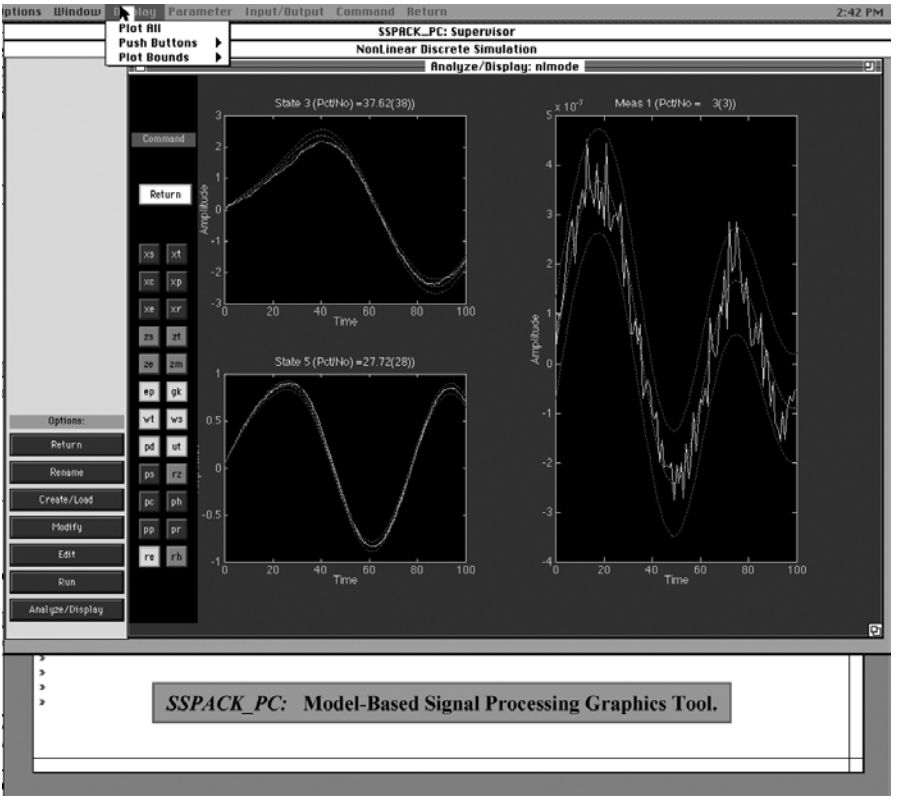



**Figure C.2.** *SSPACK_PC* GUI.

of the current algorithms available in *SSPACK_PC* (see [8], [9]). A typical display of a model-based problem using the GUI is shown in Figure C.2. Note the upper pull-down menu (Supervisor), push-button problem controls (Simulator/Estimator) and the push-button implementation of the postprocessor with graphics.

## C.6 AVAILABILITY

*SSPACK_PC* is a production code and is available for purchase from Techni-Soft(Technical Software Systems), P. O. Box 2525, Livermore, CA 94551; Phone (925) 443-7213 or (925) 743-1145. Technical details are available in the user's manual [3] or on the website:**http://www.techni-soft.net**.

## REFERENCES

1. J. Candy and P. Candy, "SSPACK_PC: A model-based signal processing package on personal computers," *DSP Applications*, **2** (3), 33–42 1993 (see **http://www.techni-soft.net** for more details).
2. G. Bierman, *Factorization Methods for Discrete Sequential Estimation*, New York: Academic Press, 1977.
3. Technical Software Systems, "SSPACK_PC: State-space systems software package," *Techni-Soft*, User's Manual, 1993.
4. A. Gelb, *Optimal Estimation*, Cambridge: MIT Press, 1975.
5. J. Candy, *Signal Processing: The Model-Based Approach*, New York: McGraw-Hill, 1986.
6. B. Anderson and J. Moore, *Optimal Filtering*, Englewood Cliffs, NJ: Prentice-Hall, 1979.
7. L. Ljung and T. Soderstrom, *Theory and Practice of Recursive Identification*, Cambridge: MIT Press, 1983.
8. S. Azevedo, J. Candy and D. Lager, "SSPACK_PC: An interactive state-space systems software package," *Proc. CACSD'85 Conf.*, Santa Barbara, CA, 1985.
9. S. Azevedo, J. Candy, and D. Lager, "SSPACK: A multi-channel signal processing package," *Proc. IEEE Circ. Syst. Conf.*, San Jose, CA, 1986.

# INDEX



*Model-Based Signal Processing*, by James V. Candy

**Adaptive and Learning Systems for Signal Processing, Communications, and Control**

***Editor: Simon Haykin***

Beckerman / ADAPTIVE COOPERATIVE SYSTEMS

Candy / MODEL-BASED SIGNAL PROCESSING

Chen and Gu / CONTROL-ORIENTED SYSTEM IDENTIFICATION: An $\mathcal{H}_\infty$ Approach

Cherkassky and Mulier / LEARNING FROM DATA: Concepts, Theory, and Methods

Diamantaras and Kung / PRINCIPAL COMPONENT NEURAL NETWORKS: Theory and Applications

Hänsler and Schmidt / ACOUSTIC ECHO AND NOISE CONTROL: A Practical Approach

Haykin / UNSUPERVISED ADAPTIVE FILTERING: Blind Source Separation

Haykin / UNSUPERVISED ADAPTIVE FILTERING: Blind Deconvolution

Haykin and Puthussarypady / CHAOTIC DYNAMICS OF SEA CLUTTER

Haykin and Widrow / LEAST-MEAN-SQUARE ADAPTIVE FILTERS

Hrycej / NEUROCONTROL: Towards an Industrial Control Methodology

Hyvärinen, Karhunen, and Oja / INDEPENDENT COMPONENT ANALYSIS

Kristić, Kanellakopoulos, and Kokotović / NONLINEAR AND ADAPTIVE CONTROL DESIGN

Mann / INTELLIGENT IMAGE PROCESSING

Nikias and Shao / SIGNAL PROCESSING WITH ALPHA-STABLE DISTRIBUTIONS AND APPLICATIONS

Passino and Burgess / STABILITY ANALYSIS OF DISCRETE EVENT SYSTEMS

Sánchez-Peña and Sznaier / ROBUST SYSTEMS THEORY AND APPLICATIONS

Sandberg, Lo, Fancourt, Principe, Katagiri, and Haykin / NONLINEAR DYNAMICAL SYSTEMS: Feedforward Neural Network Perspectives

Spooner, Maggiore, Ordóñez, and Passino / STABLE ADAPTIVE CONTROL AND ESTIMATION FOR NONLINEAR SYSTEMS: Neural and Fuzzy Approximator Techniques

Tao / ADAPTIVE CONTROL DESIGN AND ANALYSIS

Tao and Kokotović / ADAPTIVE CONTROL OF SYSTEMS WITH ACTUATOR AND SENSOR NONLINEARITIES

Tsoukalas and Uhrig / FUZZY AND NEURAL APPROACHES IN ENGINEERING

Van Hulle / FAITHFUL REPRESENTATIONS AND TOPOGRAPHIC MAPS: From Distortion- to Information-Based Self-Organization

Vapnik / STATISTICAL LEARNING THEORY

Werbos / THE ROOTS OF BACKPROPAGATION: From Ordered Derivatives to Neural Networks and Political Forecasting

Yee and Haykin / REGULARIZED RADIAL BIAS FUNCTION NETWORKS: Theory and Applications