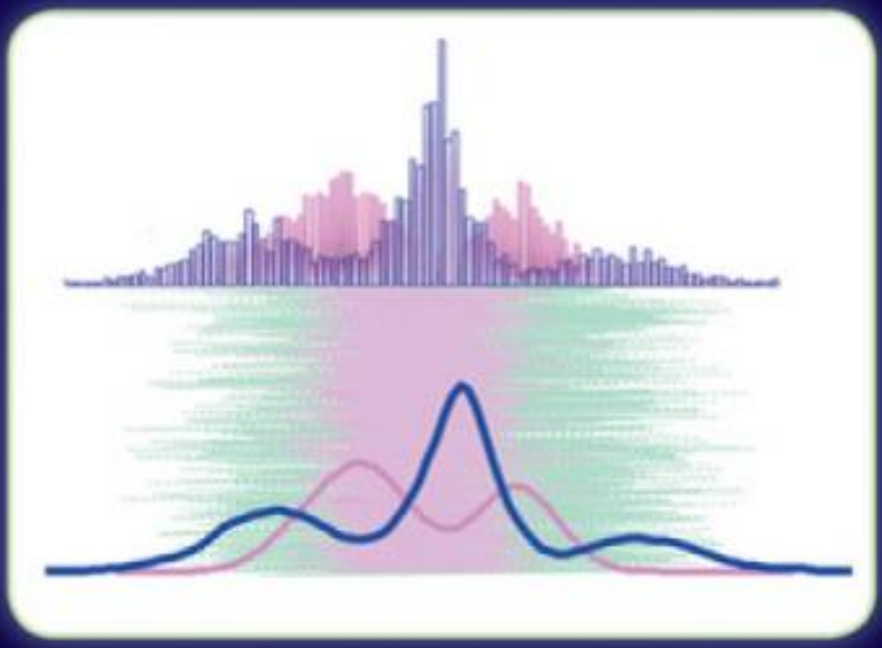
Wiley Series in Adaptive Learning Systems for Signal Processing, Communications and Control

Simon Haykin, Series Editor

Bayesian Signal Processing

Classical, Modern, and Particle Filtering Methods



James V. Candy

WILEY

IEEE
Celebrating 125 Years
of Engineering the Future

BAYESIAN SIGNAL PROCESSING

BAYESIAN SIGNAL PROCESSING

Classical, Modern, and Particle Filtering Methods

James V. Candy

Lawrence Livermore National Laboratory
University of California Santa Barbara

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:

Bayesian signal processing : classical, modern, and particle filtering methods / James V. Candy., p. cm.
Includes bibliographical references.
ISBN 978-0-470-18094-5 (cloth)
1. Signal processing—Mathematics. 2. Bayesian statistical decision theory. I. Title.
TK5102.9.C3187 2008
621.382′2–dc22

2008032184

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

Peace, real peace, can only be found through the
Lord, our Savior, Jesus Christ!

CONTENTS

PREFACE

In the real world, systems designed to extract signals from noisy measurements are plagued by errors evolving from constraints of the sensors employed, to random disturbances and noise and probably, most common, the lack of precise knowledge of the underlying physical phenomenology generating the process in the first place! Methods capable of extracting the desired signal from hostile environments require approaches that capture all of the *a priori* information available and incorporate them into a processing scheme. This approach is typically model-based [1] employing mathematical representations of the component processes involved. However, the actual implementation providing the algorithm evolves from the realm of statistical signal processing using a Bayesian approach based on Bayes' rule. Statistical signal processing is focused on the development of processors capable of extracting the desired information from noisy, uncertain measurement data. This is a text that develops the "Bayesian approach" to statistical signal processing for a variety of useful model sets. It features the next generation of processors which have recently been enabled with the advent of high speed/high throughput computers. The emphasis is on nonlinear/non-Gaussian problems, but classical techniques are included as special cases to enable the reader familiar with such methods to draw a parallel between the approaches. The common ground is the model sets. Here the state–space approach is emphasized because of its inherent applicability to a wide variety of problems both linear and nonlinear as well as time invariant and time-varying problems including what has become popularly termed "physics-based" models. This text brings the reader from the classical methods of model-based signal processing including Kalman filtering for linear, linearized and approximate nonlinear processors as well as the recently developed unscented or sigma-point filters to the next generation of processors that will clearly dominate the future of model-based signal processing for years to come. It presents a unique viewpoint of signal processing from the Bayesian perspective in contrast to the pure statistical approach found in many textbooks. Although designed primarily as a graduate textbook, it will prove very useful to the practicing signal processing professional or scientist, since a wide variety of applications are included to demonstrate the applicability of the Bayesian approach to real-world problems. The prerequisites for such a text is a melding of undergraduate

work in linear algebra, random processes, linear systems, and digital signal processing as well as a minimal background in model-based signal processing illustrated in the recent text [1]. It is unique in the sense that few texts cover the breadth of its topics, whereas, the underlying theme of this text is the Bayesian approach that is uniformly developed and followed throughout in the algorithms, examples, applications and case studies. It is this theme coupled with the hierarchy of physics-based models developed that contribute to its uniqueness. This text has evolved from three previous texts, Candy [1–3] coupled with a wealth of practical applications to real-world Bayesian problems.

The Bayesian approach has existed in statistical physics for a long time and can be traced back to the 1940s with the evolution of the Manhattan project and the work of such prominent scientists as Ulam, von Neumann, Metropolis, Fermi, Feynman, and Teller. Here the idea of Monte Carlo (*MC*) techniques to solve complex integrals evolved [4]. Since its birth, Monte Carlo related methods have been the mainstay of many complex statistical computations. Many applications have evolved from this method in such areas as physics, biology, chemistry, computer science, economics/finance, material science, statistics and more recently in engineering. Thus, statisticians have known for a long time about these methods, but their practicalities have not really evolved as a working tool until the advent of high speed super computers around the 1980s. In signal processing it is hard to pinpoint the actual initial starting point but clearly the work of Handschin and Mayne in the late 1960s and early 1970s [5, 6] was the initial evolution of Monte Carlo techniques for signal processing and control. However from the real-time perspective, it is probably the development of the sequential Bayesian processor made practical by the work of Gordon, Salmond and Smith in 1993 [7] enabling the evolution and the explosion of the Bayesian sequential processor that is currently being researched today. To put this text in perspective we must discuss the current signal processing texts available on Bayesian processing. Since its evolution much has been published in the statistical literature on Bayesian techniques for statistical estimation; however, the earliest texts are probably those of Harvey [8], Kitigawa and Gersch [9] and West [10] which emphasize the Bayesian model-based approach incorporating dynamic linear or nonlinear models into the processing scheme for additive Gaussian noise sources leading to the classical approximate (Kalman) filtering solutions. These works extend those results to non-Gaussian problems using Monte Carlo techniques for eventual solution laying the foundation for works to follow. Statistical MC techniques were also available, but not as accessible to the signal processor due to statistical jargon and abstractness of the discussions. Many of these texts have evolved during the 1990s such as Gilks [11], Robert [12], Tanner [13], Tanizaki [14], with the more up-to-date expositions evolving in the late 1990s and currently such as Liu [4], Ruanaidh [15], Haykin [16], Doucet [17], Ristic [18] and Cappe [19]. Also during the last period a sequence of tutorials and special IEEE issues evolved exposing the MC methods to the signal processing community such as Godsill [20], Arulampalam [21], Djuric [22], Haykin [23] and Doucet [24], Candy [25], as well as a wealth of signal processing papers (see references for details). Perhaps the most complete textbook from the statistical researcher's

perspective is that of Cappe [19]. In this text much of the statistical MC sampling theory is developed along with all of the detailed mathematics—ideal for an evolving researcher. But what about the entry level person—the engineer, the experimentalist, and the practitioner? This is what is lacking in all of this literature. Questions like, how do the MC methods relate to the usual approximate Kalman methods? How does one incorporate models (model-based methods) into a Bayesian processor? How does one judge performance compared with classical methods? These are all basically pragmatic questions that the proposed text will answer in a lucid manner through coupling the theory to real-world examples and applications. Thus, the goal of this text is to provide a bridge for the practitioners with enough theory and applications to provide the basic background to comprehend the Bayesian framework and enable the application of these powerful techniques to real-world problem solving. Next, let us discuss the structure of the proposed text in more detail to understand its composition and approach.

We first introduce the basic ideas and motivate the need for such processing while showing that they clearly represent the next generation of processors. We discuss potential application areas and motivate the requirement for such a generalization. That is, we discuss how the simulation-based approach to Bayesian processor design provides a much needed capability, while well known in the statistical community, not very well known (until recently) in the signal processing community. After introducing the basic concepts in Chapter 1, we begin with the basic Bayesian processors in Chapter 2. We start with the Bayesian "batch" processor and establish its construction by developing the fundamental mathematics required. Next we discuss the well-known maximum likelihood (*ML*) and minimum (error) variance (*MV*) or equivalently minimum mean-squared error (*MMSE*) processors. We illustrate the similarity and differences between the schemes. Next we launch into sequential Bayesian processing schemes which forms the foundation of the text. By examining the "full" posterior distribution in both dynamic variables of interest as well as the full data set, we are able to construct the sequential Bayesian approach and focus on the usual *filtered* or *filtering* distribution case of highest interest demonstrating the fundamental prediction/update recursions inherent in the sequential Bayesian structure. Once establishing the general Bayesian sequential processor (*BSP*) the schemes that follow are detailed depending on the assumed distribution with a variety of model sets.

We briefly review simulation-based methods starting with sampling methods, progressing to Monte Carlo approaches leading to the basic iterative methods of sampling using the Metropolis, Metropolis-Hastings, Gibb's and slice samplers. Since one of the major motivations of recursive or sequential Bayesian processing is to provide a real-time or pseudo real-time processor, we investigate the idea of *importance sampling* as well as *sequential importance sampling* techniques leading to the generic Bayesian sequential importance sampling algorithm. Here we show the solution can be applied, once the importance sampling distribution is defined.

In order to be useful, Bayesian processing techniques must be specified through a set of models that represent the underlying phenomenology driving the particular application. For example, in radar processing we must investigate the propagation

models, tracking models, geometric models, and so forth. In Chapter 4, we develop the state–space approach to signal modeling which forms the basis of many applications such as speech, radar, sonar, acoustics, geophysics, communications, control, etc. Here we investigate continuous, sampled-data and discrete state-space signals and systems. We also discuss the underlying systems theory and extend the model-set to include the stochastic case with noise driving both process and measurements leading the well-known Gauss-Markov (*GM*) representation which forms the starting point for the *classical* Bayesian processors to follow. We also discuss the equivalence of the state–space model to a variety of time series (*ARMA*, *AR*, *MA*, etc.) representations as well as the common engineering model sets (transfer functions, all-pole, all-zero, pole-zero, etc.). This discussion clearly demonstrates why the state–space model with its inherent generality is capable of capturing the essence of a broad variety of signal processing representations. Finally, we extend these ideas to *nonlinear* state–space models leading to "approximate" Gauss-Markov representation evolving from nonlinear, perturbed and linearized systems.

In the next chapter, we develop *classical* Bayesian processors by first motivating the Bayesian approach to the state–space where the required conditional distributions use the embedded state–space representation. Starting with the linear, time-varying, state–space models, we show that the "optimum" classical Bayesian processor under multivariate Gaussian assumptions leads to minimum (error) variance (*MV*) or equivalently minimum mean-squared error (*MMSE*), which is the much heralded *Kalman filter* of control theory [1]. That is, simply substituting the underlying Gauss-Markov model into the required conditional distributions leads directly to the *BSP* or Kalman filter in this case. These results are then extended to the *nonlinear* state–space representation which are linearized using a known reference trajectory through perturbation theory and Taylor-series expansions. Starting with the linearized or approximate *GM* model of Chapter 4, we again calculate the required Bayesian sequential processor from the conditionals which lead to the "linearized" *BSP* (or linearized Kalman filter) algorithm. Once this processor is developed, it is shown that the "extended" Bayesian processor follows directly by linearizing about the most currently available estimate rather than the reference trajectory. The extended Bayesian processor (*XBP*) or equivalently extended Kalman filter (*EKF*) of nonlinear processing theory evolves quite naturally from the Bayesian perspective, again following the usual development by defining the required conditionals, making nonlinear approximations and developing the posterior distributions under multivariate Gaussian assumptions. Next, we briefly investigate an iterative version of the *XBP* processor, again from the Bayesian perspective which leads directly to the iterative version of the extended Bayesian processor (*IX-BP*) algorithm—an effective tool when nonlinear measurements dominate the uncertain measurements required.

Chapter 6 focuses on *statistical linearization* methods leading to the *modern* unscented Bayesian processor (*UBP*) or equivalently sigma-point Bayesian processor (*SPBP*). Here we show how statistical linearization techniques can be used to transform the underlying probability distribution using the sigma-point or unscented nonlinear transformation technique (linear regression) leading to the *unscented*

Bayesian processor or equivalently the unscented Kalman filter (*UKF*). Besides developing the fundamental theory and algorithm, we demonstrate its performance on a variety of example problems. We also briefly discuss the Gaussian-Hermite quadrature (*G-H*) and Gaussian sum (*G-S*) techniques for completeness.

We reach the heart of the *particle filtering* methods in Chapter 7, where we discuss the Bayesian approach to the state–space. Here the ideas of Bayesian and model-based processors are combined through the development of Bayesian state–space particle filters. Initially, it is shown how the state–space models of Chapter 4 are incorporated into the conditional probability distributions required to construct the sequential Bayesian processors through importance sampling constructs. After investigating a variety of importance proposal distributions, the basic set of state-space particle filters (*SSPF*) are developed and illustrated through a set of example problems and simulations. The techniques including the Bootstrap, auxiliary, regularized *MCMC* and linearized particle filters are developed and investigated when applied to the set of example problems used to evaluate algorithm performance.

In Chapter 8 the important joint Bayesian *SSPF* are investigated by first developing the joint filter popularly known as the parametrically adaptive processor [1]. Here both states and static as well as dynamic parameters are developed as solutions to this joint estimation problem. The performance of these processors are compared to classical and modern processors through example problems.

In Chapter 9 the hidden Markov models (*HMM*) are developed for event related problems (e.g., Poisson point processes). This chapter is important in order to place purely discrete processes into perspective. *HMM* evolve for any type of memoryless, counting processes and become important in financial applications, communications, biometrics, as well as radiation detection. Here we briefly develop the fundamental ideas and discuss them in depth to develop a set of techniques used by the practicioner while applying them to engineering problems of interest.

In the final chapter, we investigate a set of physics-based applications focusing on the Bayesian approach to solving real-world problems. By progressing through a step-by-step development of the processors, we see explicitly how to develop and analyze the performance of such Bayesian processors. We start with a practical laser alignment problem followed by a broadband estimation problem in ocean acoustics. Next the solid-state microelectromechanical (*MEM*) sensor problem for biothreat detection is investigated followed by a discrete radiation detection problem based on counting statistics. All of these methods invoke Bayesian techniques to solve the particular problems of interest enabling the practitioner the opportunity to track "real-world" Bayesian model-based solutions.

The place of such a text in the signal processing textbook community can best be explained by tracing the technical ingredients that comprise 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 [26–28], Therrien [29], Brown [30] all provide the basic background information in much more detail than this text, so there is little overlap at the detailed level with them.

This text, however, possesses enough theory for the graduate or advanced graduate student to develop a fundamental basis to go onto more rigorous texts like Jazwinski [31], Sage [32], Gelb [33], Anderson [34], Maybeck [35], Bozic [36], Kailath [37, 38], and more recently, Mendel [39], Grewel [40], Bar-Shalom [41] and Simon [42]. These texts are rigorous and tend to focus on Kalman filtering techniques ranging from continuous to discrete with a wealth of detail on all of their variations. The Bayesian 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 of the more recent texts about the same technical level, but again, with a different focus: are Widrow [43], Orfanidis [44], Sharf [45], Haykin [46], Hayes [47], Brown [30] and Stoica [48]. Again the focus of these texts is not the Bayesian approach but more on narrow set of specific models and the development of a variety of algorithms to estimate these sets. The system identification literature and texts therein also provide some overlap with this text, but again the approach is focused on estimating a model from noisy data sets and not really aimed at developing a Bayesian solution to a particular signal processing problem. The texts in this area are Ljung [49, 50], Goodwin [51], Norton [52] and Soderstrom [53].

The recent particle filtering texts of Ristic [18] and Cappe [19] are useful as references to accompany this text, especially if more details are required on the tracking problem and the fundamental theorems governing statistical properties and convergence proofs. That is, Ristic's text provides a introduction that closely follows the 2002 tutorial paper by Arulampalam [21] but provides little of the foundational material necessary to comprehend this approach. It focuses primarily on the tracking problem. Cappe's text is at a much more detailed technical level and is written for researcher's in this area not specifically aimed at the practitioner's viewpoint. The proposed text combines the foundational material, some theory along with the practice and application of *PF* to real-world applications and examples.

The approach we take is to introduce the basic idea of Bayesian signal processing and show where it fits in terms of signal processing. It is argued that *BSP* 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, biomedical) and simple examples are incorporated to motivate the signal processor. Examples are discussed to motivate all of the models and prepare the reader for further developments in subsequent chapters. In each case the processor along with accompanying simulations are discussed and applied to various data sets demonstrating the applicability and power of the Bayesian approach. The proposed text is linked to the MATLAB (signal processing standard software) software package providing notes at the end of each chapter.

In summary, this Bayesian signal processing text will provide a much needed "down-to-earth" exposition of modern *MC* techniques. It is coupled with well-known signal processing model sets along with examples and problems that can be used to solve many real-world problems by practicing engineers and scientists along

with entry level graduate students as well as advanced undergraduates and post-doctorates requiring a solid introduction to the "next generation" of model-based signal processing techniques.

JAMES V. CANDY

Danville, California
January 2009

REFERENCES TO THE PREFACE

1. J. Candy, *Model-Based Signal Processing* (Hoboken, NJ: John Wiley/IEEE Press, 2006).
2. J. Candy, *Signal Processing: The Model-Based Approach* (New York: McGraw-Hill, 1986).
3. J. Candy, *Signal Processing: The Modern Approach* (New York: McGraw-Hill, 1988).
4. J. Liu, *Monte Carlo Strategies in Scientific Computing* (New York: Springer-Verlag, 2001).
5. J. Handschin and D. Mayne, "Monte Carlo techniques to estimate the conditional expectation in multi-stage nonlinear filtering," *Intl. J. Control*, **9**, 547–559, 1969.
6. J. Handschin, "Monte Carlo techniques for prediction and filtering of nonlinear stochastic processes," *Automatica*, **6**, 555–563, 1970.
7. N. Gordon, D. Salmond and A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation, *IEE Proc.-F.*, **140**, 107–113, 1993.
8. A. Harvey, *Forcasting, Structural Time Series Models and the Kalman Filter* (Cambridge, UK: Cambridge University Press, 1989).
9. G. Kitagawa and W. Gersch, *Smoothness Priors Analysis of Time Series* (New York: Springer-Verlag, 1996).
10. M. West and J. Harrison, *Bayesian Forcasting and Dynamic Models*, 2^{nd} *Ed.* (New York: Springer-Verlag, 1997).
11. W. Gilks, S. Richardson and D. Spiegelhalter, *Markov Chain Monte Carlo in Practice* (New York: Chapman & Hall/CRC Press, 1996).
12. C. Robert and G. Casella, *Monte Carlo Statistical Methods* (New York: Springer, 1999).
13. M. Tanner, *Tools for Statistical Inference: Methods for the Exploration of Posterior Distributions and Likelihood Functions*, 2^{nd} *Ed.* (New York: Springer-Verlag, 1993).
14. H. Tanizaki, "Nonlinear filters: Estimation and Applications." *Lecture Notes in Economics and Math. Sys.* (New York: Springer, 1993).
15. J. Ruanaidh and W. Fitzgerald, *Numerical Bayesian Methods Applied to Signal Processing* (New York: Springer-Verlag, 1996).
16. S. Haykin, *Kalman Filtering and Neural Networks* (Hoboken, NJ: Wiley: 2001).
17. A. Doucet, N. de Freitas and N. Gordon, *Sequential Monte Carlo Methods in Practice*, (New York: Springer-Verlag, 2001).
18. B. Ristic, S. Arulampalam and N. Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Applications* (Boston: Artech House, 2004).
19. O. Cappe, E. Moulines and T. Ryden, *Inference in Hidden Markov Models* (New York: Springer-Verlag, 2005).

20. S. Godsill and P. Djuric, "Special Issue: Monte Carlo methods for statistical signal processing." *IEEE Trans. Signal Proc.*, **50**, 173–499, 2002.
21. M. Arulampalam, S. Maskell, N. Gordon and T. Clapp "A tutorial on particle filters for online nonlinear/non-gaussian Bayesian tracking." *IEEE Trans. Signal Proc.*, **50**, 2, 174–188, 2002.
22. 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.
23. S. Haykin and N. de Freitas, "Special Issue: Sequential state estimation: from Kalman filters to particle filters." *Proc. IEEE*, **92**, 3, 399–574, 2004.
24. A. Doucet and X. Wang, "Monte Carlo methods for signal processing," *IEEE Signal Proc. Mag.* **24**, 5, 152–170, 2005.
25. J. Candy, "Bootstrap particle filtering for passive synthetic aperture in an uncertain ocean environment." *IEEE Signal Proc. Mag.* **24**, 4, 63–75, 2007.
26. S. Kay, *Modern Spectral Estimation: Theory and Applications* (Englewood Cliffs, NJ: Prentice-Hall, 1988).
27. S. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory* (Englewood Cliffs, NJ: Prentice-Hall, 1993).
28. S. Kay, *Fundamentals of Statistical Signal Processing: Detection Theory* (Englewood Cliffs, NJ: Prentice-Hall, 1998).
29. C. Therrian, *Random Signal Processing: Detection Theory* (Englewood Cliffs, NJ: Prentice-Hall, 1991).
30. R. Brown and P. Hwang, *Introduction to Random Signals and Applied Kalman Filtering* (New York: Wiley, 1997).
31. A. Jazwinski, *Stochastic Processes and Filtering Theory* (New York: Academic Press, 1970).
32. A. Sage and J. Melsa, *Estimation Theory with Applications to Communcations and Control* (New York: McGraw-Hill, 1971).
33. A. Gelb, *Applied Optimal Estimation* (Cambridge, MA: M.I.T. Press, 1974).
34. B. Anderson and J. Moore, *Optimum Filtering* (Englewood Cliffs, NJ: Prentice-Hall, 1979).
35. P. Maybeck, *Stochastic Models, Estimation and Control* (New York: Academic Press, 1979).
36. M. S. Bozic, *Linear Estimation Theory* (Englewood Cliffs, NJ: Prentice-Hall, 1979).
37. T. Kailath, *Lectures on Kalman and Wiener Filtering Theory* (New York: Springer-Verlag, 1981).
38. T. Kailath, A. Sayed and B. Hassibi, *Linear Estimation* (Englewood Cliffs, NJ: Prentice-Hall, 2000).
39. J. Mendel, *Lessons in Estimation Theory for Signal Processing, Communications, and Control* (Englewood Cliffs, NJ: Prentice-Hall, 1995).
40. M. Grewal and A. Andrews, *Kalman Filtering: Theory and Practice* (Englewood Cliffs, NJ: Prentice-Hall, 1993).
41. Y. Bar-Shalom and X. Li, *Estimation and Tracking: Principles, Techniques and Software* (Boston: Artech House, 1993).
42. D. Simon, *Optimal State Estimation: Kalman, H_∞ and Nonlinear Approaches* (Hoboken, NJ: Wiley, 2006).

43. B. Widrow and S. Stearns, *Adaptive Signal Processing* (Englewood Cliffs, NJ: Prentice-Hall, 1985).
44. S. Orfanidis, *Optimal Signal Processing* (New York: MacMillan, 1988).
45. L. Sharf, *Statistical Signal Processing: Detection, Estimation and Time Series Analysis*, (Reading, MA: Addison-Wesley, 1991).
46. S. Haykin, *Adaptive Filter Theory* (Englewood Cliffs, NJ: Prentice-Hall, 1993).
47. M. Hayes, *Statistical Digital Signal Processing and Modeling* (Hoboken, NJ: Wiley, 1996).
48. P. Stoica and R. Moses, *Introduction to Spectral Analysis* (Englewood Cliffs, NJ: Prentice-Hall, 1997).
49. L. Ljung, *System Identification: Theory for the User* (Englewood Cliffs, NJ: Prentice-Hall, 1987).
50. L. Ljung and T.Soderstrom, *Theory and Practice of Recursive Identification* (Cambridge, MA: M.I.T. Press, 1983).
51. G. Goodwin and K. S. Sin, *Adaptive Filtering, Prediction and Control* (Englewood Cliffs, NJ: Prentice-Hall, 1984).
52. J. Norton, *An Introduction to Identification* (New York: Academic Press, 1986).
53. T. Soderstrom and P. Stoica, *System Identification* (New York: Academic Press, 1989).

ACKNOWLEDGMENTS

My wife, Patricia, was instrumental in suggesting that I write this text, so a big hearty thank you, with love to Patricia. My mother, Anne, who taught me the important lesson of tenacity, is an influence that will continue forever, thanks Mom. Of course, the constant support of my great colleagues and friends, especially Drs. E. Bond, G. Clark and M. Krnjajic, who carefully reviewed the manuscript and suggested many improvements, as well as Drs. D. Chambers, E. Sullivan, S. Godsill, N. Kingsbury and W. Ng cannot go without a strong acknowledgement. I wish to acknowledge the support and motivation from Dr. Simon Haykin, a colleague and friend.

1

INTRODUCTION

1.1 INTRODUCTION

In this chapter we motivate the philosophy of Bayesian processing from a probabilistic perspective. We show the coupling between model-based signal processing (*MBSP*) incorporating the *a priori* knowledge of the underlying processes and the Bayesian framework for specifying the distribution required to develop the processors. The idea of the sampling approach evolving from Monte Carlo (*MC*) and Markov chain Monte Carlo (*MCMC*) methods is introduced as a powerful methodology for simulating the behavior of complex dynamic processes and extracting the embedded information required. The main idea is to present the proper perspective for the subsequent chapters and construct a solid foundation for solving signal processing problems.

1.2 BAYESIAN SIGNAL PROCESSING

The development of Bayesian signal processing has evolved in a manner proportional to the evolution of high performance/high throughput computers. This evolution has led from theoretically appealing methods to pragmatic implementations capable of providing reasonable solutions for nonlinear and highly multi-modal (multiple distribution peaks) problems. In order to fully comprehend the Bayesian perspective, especially for signal processing applications, we must be able to separate our thinking and in a sense think more abstractly about probability distributions without worrying about how these representations can be "applied" to realistic processing problems. Our motivation is to first present the Bayesian approach from a statistical viewpoint and then couple it to useful signal processing implementations following the well-known model-based approach [1, 2]. Here we show that when we constrain the Bayesian

Bayesian Signal Processing. By James V. Candy

distributions in estimation to Markovian representations using primarily state–space models, we can construct sequential processors capable of "pseudo real-time" operations that are easily be utilized in many physical applications. Bayes' rule provides the foundation of all Bayesian estimation techniques. We show how it can be used to both theoretically develop processing techniques based on a specific distribution (e.g., Poisson, Gaussian, etc.) and then investigate properties of such processors relative to some of the most well-known approaches discussed throughout texts in the field.

Bayesian signal processing is concerned with the estimation of the underlying probability distribution of a random signal in order to perform statistical inferences [3]. These inferences enable the extraction of the signal from noisy uncertain measurement data. For instance, consider the problem of extracting the random variate, say X, from the noisy data, Y. The Bayesian approach is to first *estimate* the underlying conditional probability distribution, $\Pr(X|Y)$, and then perform the associated inferences to extract $\hat{X}$, that is,

$$\hat{\Pr}(X|Y) \Rightarrow \hat{X} = \arg\max_X \hat{\Pr}(X|Y)$$

where the caret, $\hat{X}$ denotes an estimate of X. This concept of estimating the underlying distribution and using it to extract a signal estimate provides the foundation of Bayesian signal processing developed in this text.

Let us investigate this idea in more detail. We start with the previous problem of trying to estimate the random parameter, X, from noisy data $Y = y$. Then the associated conditional distribution $\Pr(X|Y = y)$ is called the *posterior distribution* because the estimate is conditioned "after (*post*) the measurements" have been acquired. Estimators based on this *a posteriori* distribution are usually called *Bayesian* because they are constructed from *Bayes' rule*, since $\Pr(X|Y)$ is difficult to obtain directly. That is,

$$\Pr(X|Y) = \frac{\Pr(Y|X) \times \Pr(X)}{\Pr(Y)} \tag{1.1}$$

where $\Pr(X)$ is called the *prior distribution* (before measurement), $\Pr(Y|X)$ 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* distribution. As measurements are made, the *prior* is transformed to the *posterior distribution* function adjusting the parameter estimates. Thus, the result of increasing the number of measurements is to improve the *a posteriori* distribution resulting in a sharper peak closer to the true parameter as shown in Fig. 1.1.

When the variates of interest are *dynamic*, then they are functions of time and therefore, $X_t \to X$ and $Y_t \to Y$. Bayes' rule for the joint dynamic distribution is

$$\Pr(X_t|Y_t) = \frac{\Pr(Y_t|X_t) \times \Pr(X_t)}{\Pr(Y_t)} \tag{1.2}$$

In Bayesian theory, the *posterior* defined by $\Pr(X_t|Y_t)$ is decomposed in terms of the *prior* $\Pr(X_t)$, its *likelihood* $\Pr(Y_t|X_t)$ and the *evidence* or normalizing factor,

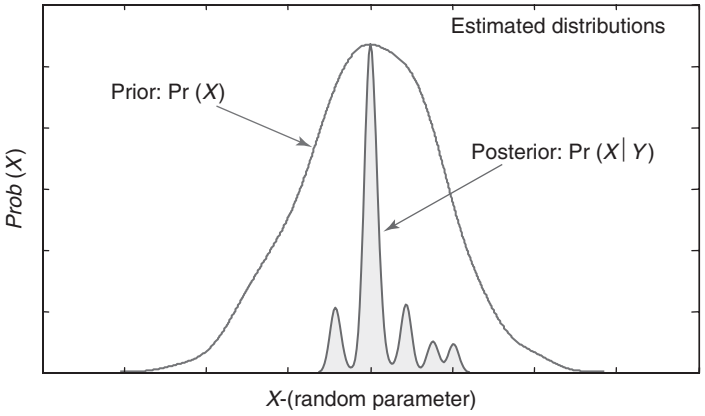


FIGURE 1.1 Bayesian estimation of the random variate X transforming the prior, Pr(X) to the posterior, Pr($X|Y$) using Bayes' rule.

$\Pr(Y_t)$. Bayesian signal processing in this dynamic case follows the identical path, that is,

$$\hat{\Pr}(X_t|Y_t) \Rightarrow \hat{X}_t = \arg\max_{X_t} \hat{\Pr}(X_t|Y_t)$$

So we begin to see the versatility of the Bayesian approach to random signal processing. Once the posterior distribution is determined, then all statistical inferences or estimates are made. For instance, suppose we would like to obtain the prediction distribution. Then it can be obtained as

$$\Pr(X_{t+1}|Y_t) = \int \Pr(X_{t+1}|X_t, Y_t) \times \Pr(X_t|Y_t)\, dX_t$$

and a point estimate might be the conditional mean of this distribution, that is,

$$E\{X_{t+1}|Y_t\} = \int X_{t+1} \Pr(X_{t+1}|Y_t)\, dX_{t+1}$$

This relation shows how information that can be estimated from the extracted distribution is applied in the estimation context by performing statistical inferences.

Again, even though the Bayesian signal processing concept is simple, conceptually, the real problem to be addressed is that of evaluating the integrals which is very difficult because they are only analytically tractable for a small class of priors and likelihood distributions. The large dimensionality of the integrals cause numerical integration techniques to break down, which leads to the approximations we discuss subsequently for stabilization. Next let us consider the various approaches taken to solve the probability distribution estimation problems using non-parametric or parametric representations. This will eventually lead to the model-based approach [4].

1.3 SIMULATION-BASED APPROACH TO BAYESIAN PROCESSING

The *simulation-based approach* to Bayesian processing is founded on Monte Carlo (*MC*) methods that are stochastic computational techniques capable of efficiently simulating highly complex systems. Historically motivated by games of chance and encouraged by the development of the first electronic computer (ENIAC), the *MC* approach was conceived by Ulam (1945), developed by Ulam, Metropolis and von Neumann (1947) and coined by Metropolis (1949) [5–9]. The method evolved in the mid-1940s during the Manhattan project by scientists investigating calculations for atomic weapon designs [10]. It evolved further from such areas as computational physics, biology, chemistry, mathematics, engineering, materials and finance to name a few. *Monte Carlo* methods offer an alternative approach to solving classical numerical integration and optimization problems. Inherently, as the dimensionality of the problem increases classical methods are prone to failure while *MC* methods tend to increase their efficiency by reducing the error—an extremely attractive property. For example, in the case of classical grid-based numerical integration or optimization problems as the number of grid points increase along with the number of problem defining vector components, there is an accompanying exponential increase in computational time [10–15]. The stochastic *MC* approach of selecting random samples and averaging over a large number of points actually reduces the computational error by the Law of Large Numbers irrespective of the problem dimensionality. It utilizes Markov chain theory as its underlying foundation establishing the concept that through random sampling the resulting "empirical" distribution converges to the desired posterior called the stationary or invariant distribution of the chain. Markov chain Monte Carlo (*MCMC*) techniques are based on sampling from probability distributions based on a Markov chain, which is a stochastic system governed by a transition probability, having the desired posterior distribution as its invariant distribution. Under certain assumptions the chain converges to the desired posterior through proper random sampling as the number of samples become large—a crucial property (see [10] for details). Thus, the Monte Carlo approach has evolved over a long time period and is well understood by scientists and statisticians, but it must evolve even further to be useful for signal processors to become an effective tool in their problem solving repertoire.

Perhaps the best way to visualize the *MC* methods follows directly from the example of Frenkel [11]. Suppose that a reasonable estimate of the depth of the Mississippi river is required. Using numerical quadrature techniques the integrand value is measured at prespecified grid points. We also note that the grid points may not be in regions of interest and, in fact, the integrand may vanish as shown in Fig. 1.2. On the other hand, the surveyor is in the Mississippi and performing a (random) walk within the river measuring the depth of the river directly. In this sampling approach measurements are accepted as long as the surveyor is in the river and rejected if outside. Here the "average" depth is simply the sample average of the measurements much the same as a sampling technique might perform. So we see that a refinement of the brute force integration approach is to use random points or samples that "most likely" come from regions of high contribution to the integral rather than from low regions.

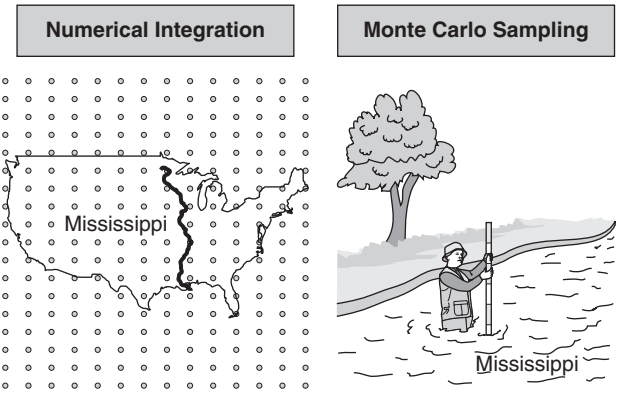


FIGURE 1.2 Monte Carlo sampling compared with numerical grid based integration for depth of Mississippi estimation.

Modern *MC* techniques such as in numerical integration seek to select random samples in high regions of concentration of the integrand by drawing samples from a proposed function very similar to the integrand. These methods lead to the well-known importance sampling approaches (see Chapter 3). Besides numerical integration problems that are very important in statistical signal processing for extracting signals/parameters of interest, numerical optimization techniques (e.g., genetic algorithms, simulated annealing, etc.) benefit directly from sampling technology. This important discovery has evolved ever since and become even more important with the recent development of high speed/high throughput computers.

Consider the following simple example of estimating the area of a circle to illustrate the *MC* approach.

Example 1.1

Define a sample space bounded by a square circumscribing (same center) a circle of radius r. Draw uniform random samples say $z := (X, Y)$ such that $z \sim \mathcal{U}(-r, +r)$; therefore, the number of random samples drawn from within the circle of radius r to the number of total samples drawn (bounded by the square) defines the probability

$$\Pr(Z = z) = \frac{\text{No. circle samples}}{\text{Total No. of (square) samples}}$$

From geometry we know that the probability is simply the ratio of the two areas (circle-to-square), that is,

$$\Pr(Z = z) = \frac{\pi r^2}{4r^2} = \pi/4$$

Let $r = 1$, then a simple computer code can be written that:

- Draws the X,Y-coordinates from $z \sim \mathcal{U}(-1, +1)$;
- Calculates the range function, $\rho = \sqrt{X^2 + Y^2}$;
- Counts the number of samples that are less than or equal to ρ;
- Estimates the probability, $\hat{\Pr}(Z = z)$.

The area is determined by multiplying the estimated probability by the area of the square. The resulting sample scatter plot is shown in Fig. 1.3 for a 10,000 sample realization resulting in $\pi \approx 3.130$. As the number of samples increase the estimate of the area (π) gets better and better demonstrating the *MC* approach. △△△

In signal processing, we are usually interested in some statistical measure of a random signal or parameter usually expressed in terms of its moments [16–23]. For example, suppose we have some signal function, say $f(X)$, with respect to some underlying probabilistic distribution, $\Pr(X)$. Then a typical measure to seek is its performance "on the average" which is characterized by the expectation

$$E_X\{f(X)\} = \int f(X) \Pr(X)\, dX \tag{1.3}$$

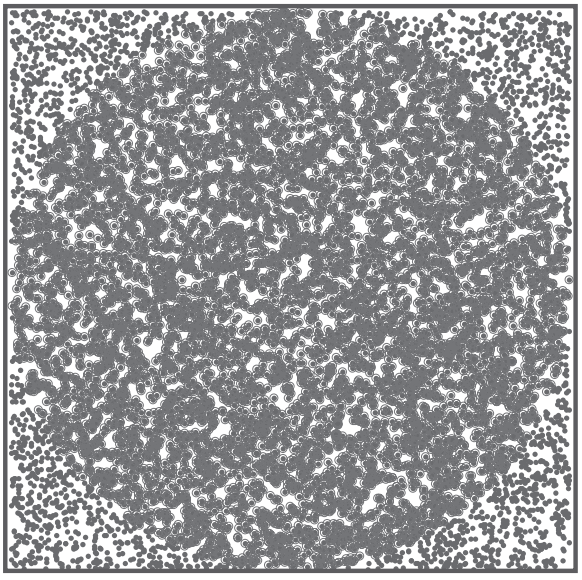


FIGURE 1.3 Area of a circle of unit radius using a Monte Carlo approach (area is estimated as 3.130 using 10,000 samples).

Instead of attempting to use direct numerical integration techniques, stochastic sampling techniques or *Monte Carlo integration* is an alternative. As mentioned, the *key idea* embedded in the *MC* approach is to represent the required distribution as a set of random *samples* rather than a specific analytic function (e.g., Gaussian). As the number of samples becomes large, they provide an equivalent (empirical) representation of the distribution enabling moments to be estimated directly (inference).

Monte Carlo integration draws samples from the required distribution and then forms sample averages to approximate the sought after distributions. That is, *MC* integration evaluates integrals by drawing samples, $\{X(i)\}$ from the designated distribution $\Pr(X)$. Assuming perfect sampling, this produces the estimated or *empirical distribution* given by

$$\hat{\Pr}(X) \approx \frac{1}{N}\sum_{i=1}^{N}\delta(X - X(i))$$

which is a probability mass distribution with weights, $\frac{1}{N}$ and random variable or sample, $X(i)$. Substituting the empirical distribution into the integral gives

$$E_X\{f(X)\} = \int f(X)\,\hat{\Pr}(X)\,dX \approx \frac{1}{N}\sum_{i=1}^{N} f(X(i)) \equiv \bar{f} \tag{1.4}$$

which follows directly from the sifting property of the delta or impulse function. Here $\bar{f}$ is said to be a *Monte Carlo estimate* of $E_X\{f(X)\}$.

As stated previously, scientists (Ulam, von Neumann, Metropolis, Fermi, Teller, etc. [7]) created statistical sampling-based or equivalently *simulation-based* methods for solving problems efficiently (e.g., neutron diffusion or eigenvalues of the Schrodinger relation). The *MC* approach to problem solving is a class of stochastic computations to simulate the dynamics of physical or mathematical systems capturing their inherent uncertainties. The *MC method* is a powerful means for generating random samples used in estimating conditional and marginal probability distributions required for statistical estimation and therefore signal processing. It offers an alternative numerical approach to find solutions to mathematical problems that cannot easily be solved by integral calculus or other numerical methods. As mentioned, the efficiency of the *MC* method increases (relative to other approaches) as the problem dimensionality increases. It is useful for investigating systems with a large number of degrees of freedom (e.g., energy transport, materials, cells, genetics) especially for systems with input uncertainty [5].

These concepts have recently evolved to the signal processing area and are of high interest in nonlinear estimation problems especially in model-based signal processing applications [16] as discussed next.

1.4 BAYESIAN MODEL-BASED SIGNAL PROCESSING

The estimation of probability distributions required to implement Bayesian processors is at the heart of this approach. How are these distributions obtained from data or simulations? Nonparametric methods of distribution estimation ranging from simple histogram estimators to sophisticated kernel smoothing techniques rooted in classification theory [3] offer reasonable approaches when data are available. However, these approaches usually do not take advantage of prior knowledge about the underlying physical phenomenology generating the data. An alternative is to parameterize the required distributions by prior knowledge of their actual form (e.g., exponential, Poisson, etc.) and fit their parameters from data using optimization techniques [3]. Perhaps the ideal realization is the parameterization of the evolution dynamics associated with the physical phenomenology using underlying mathematical representation of the process combined with the data samples. This idea provides the essence of the model-based approach to signal processing which (as we shall see) when combined with the Bayesian processors provide a formidable tool to attack a wide variety of complex processing problems in a unified manner. An alternative 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 signal processing [1].

Simply stated, 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 (*MBP*) enables the interpretation of results directly in terms of the problem physics. It is actually a modeler's tool enabling the incorporation of any *a priori* information about the problem to extract the desired information. 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, non-physical representation of the measurement data such as the Fourier or wavelet transforms to parametric black-box models used for data prediction, to lumped mathematical representation 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 non-physical 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. For instance, consider the example of

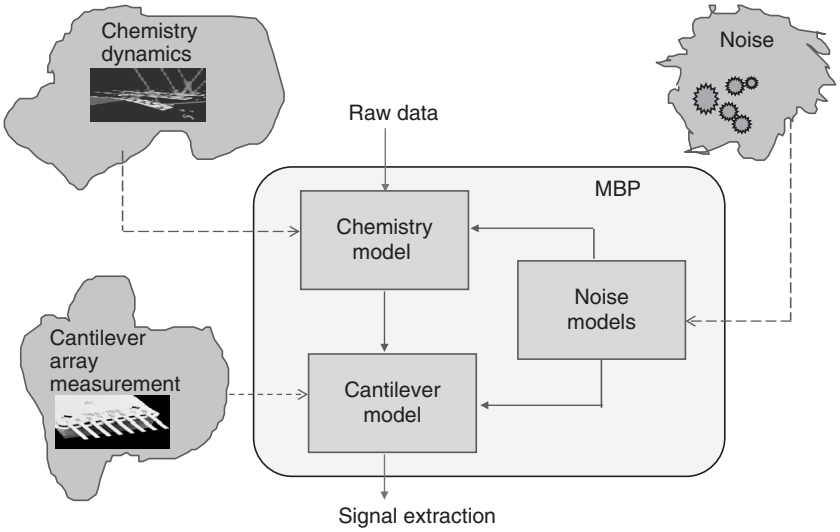


FIGURE 1.4 Model-based approach to signal processing: process (chemistry and physics), measurement (microcantilever sensor array) and noise (Gaussian) representations.

detecting the presence of a particular species in a test solution using a microcantilever sensor measurement system [4].

Example 1.2

The model-based processing problem is characterized in Fig. 1.4 representing the process of estimating the presence of a particular species of material in solution using the multichannel microcantilever sensor system. Here the microcantilever sensor is pre-conditioned by depositing attractor material on its levers to attract molecules of the target species. Once calibrated, the test solution flows along the levers with the target molecules attracted and deposited on each "tuned" microcantilever creating a deflection that is proportional to the concentration. This deflection is measured using a laser interferometric technique and digitized for processing. The process model is derived directly from the fluidics, while the measurement model evolves from the dynamics of the microcantilever structure. The resulting processor is depicted in Fig. 1.5, where we note the mathematical models of both the process dynamics and microcantilever measurement system. Since parameters, Θ, of the model are unknown *a priori* calibration data is used to estimate them directly and then they are employed in the *MBP* to provide the enhanced signal estimate shown in the figure. Even though nonlinear and non-Gaussian, the processor appears to yield reasonable estimates. See Sec. 10.3 [4] for details. $\triangle\triangle\triangle$

The above example demonstrates that incorporating reasonable mathematical models of the underlying phenomenology can lead to improved processing capability;

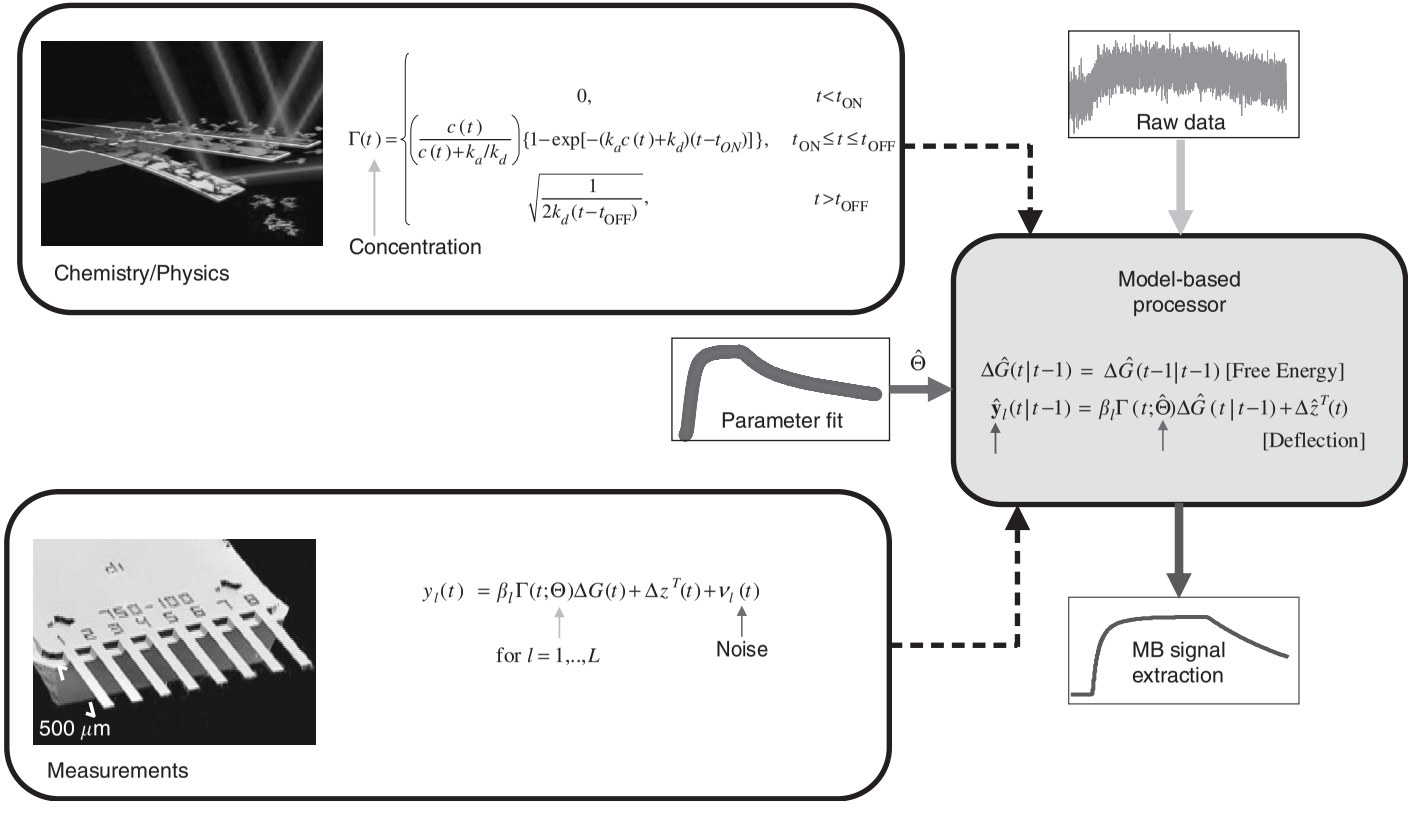


FIGURE 1.5 Model-based processor representation of species detection problem: process (concentration model), measurement (microcantilever sensor array), raw data, parameter estimator (coefficients) and model-based processor (enhancement).

however, even further advantages can be realized by combining the *MBP* concepts in conjunction with Bayesian constructs to generalize solutions.

Combining Bayesian and model-based signal processing can be considered a parametric representation of the required distributions using mathematical models of the underlying physical phenomenology and measurement (sensor) system. Certainly, if we assume the distribution is Gaussian and we further constrain the processes to be Markovian (only depending on the previous sample), then the multivariate Gaussian can be completely characterized using state–space models resulting in the well-known Kalman filter in the linear model case [2].

Since we are primarily concerned with pseudo real-time techniques in this text, we introduce the notion of a recursive form leading to the idea of sequential processing techniques. That is, we investigate "recursive" or equivalently "sequential" solutions to the estimation problem. Recursive estimation techniques evolved quite naturally during the advent of the digital computer in the late fifties, since both are sequential processes. It is important to realize the recursive solution is identical to the batch solution after it converges, so there is *no gain* in estimator performance properties; however, the number of computations is significantly less than the equivalent batch technique. It is also 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 [2]!

Many processors can be placed in a recursive form with various subtleties emerging in the calculation of the current estimate ($\hat{X}_{old}$). 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*:

$$\hat{X}_{new} = \hat{X}_{old} + KE_{new} \tag{1.5}$$

where

$$E_{new} = Y - \hat{Y}_{old} = Y - C\hat{X}_{old}$$

Here we see that the new estimate is obtained by correcting the old estimate with a K-weighted error. The error term E_{new} is the new information or innovation—the difference between the actual and the predicted measurement ($\hat{Y}_{old}$) based on the old estimate ($\hat{X}_{old}$). The computation of the weight matrix K depends on the criterion used (e.g., mean-squared error, absolute error, etc.).

Consider the following example, which shows how to recursively estimate the sample mean.

Example 1.3

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,

$$\hat{X}(N) = \frac{1}{N} y(N) + \frac{N-1}{N} \hat{X}(N-1)$$

The recursive form is given by

$$\underbrace{\hat{X}(N)}_{NEW} = \underbrace{\hat{X}(N-1)}_{OLD} + \underbrace{\frac{1}{N}}_{WT} \underbrace{[y(N) - \hat{X}(N-1)]}_{ERROR}$$

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; and
3. Perform the algebra to determine the gain factor and place the estimator in the recursive form of Eq. 1.5 for a *scalar* measurement. △△△

1.5 NOTATION AND TERMINOLOGY

The notation used throughout this text is standard in the literature. Where necessary, vectors are represented by boldface, lowercase, $\mathbf{x}$, 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, such as $\hat{x}$. We also define partial derivatives at the component level by $\frac{\partial}{\partial \theta_i}$, the N_θ-gradient vector by ∇_θ and higher order partials by ∇_θ^2.

The most difficult notational problem will be with the "time" indices. Since this text is predominantly discrete-time, we will use the usual time symbol, t to mean a discrete-time index, that is, $t \in \mathcal{I}$ for $\mathcal{I}$ the set of integers. However, and hopefully not too confusing, t will also be used for continuous-time, that is, $t \in \mathcal{R}$ 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, that is, $x_{\mathbf{t}}$. 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 upon all of the previous data up to $t-1$. We will define these symbols prior to their use within the text to assure no misunderstanding of its meaning.

With a slight abuse of notation, we will use the terminology distribution of X, $\Pr(X)$ in general, so as not to have to differentiate between density for continuous random variables or processes and mass for discrete variates. It will be obvious from the context which is meant. In some cases, we will be required to make the distinction between cumulative distribution function (*CDF*) and density (*PDF*) or mass (*PMF*) functions. Here we use the uppercase notation, $\mathrm{P}_X(x)$ for the *CDF* and lower case $\mathrm{p}_X(x)$ for the *PDF* or *PMF*.

Subsequently we will also need to express a discrete *PMF* as a continuous *PDF* using impulse or delta functions as "samplers" much the same as in signal processing when we assume there exists an impulse sampler that leads to the well-known Nyquist sampling theorem [2]. Thus, corresponding to a discrete *PMF* we can define a continuous *PDF* through the concept of an *impulse sampler*, that is, given a discrete *PMF* defined by

$$\mathrm{p}_X(x) \approx \mathrm{p}(X = x_i) = \sum_i p_i \, \delta(x - x_i) \tag{1.6}$$

then we define the *equivalent continuous PDF* as $\mathrm{p}_X(x)$. Moments follow from the usual definitions associated with a continuous *PDF*, for instance, consider the definition of the expectation or mean. Substituting the equivalent *PDF* and utilizing the sifting property of the impulse function gives

$$E\{x\} = \int_{-\infty}^{\infty} x \, \mathrm{p}_X(x) \, dx = \int_{-\infty}^{\infty} x \left(\sum_i p_i \, \delta(x - x_i) \right) dx = \sum_i x_i p_i \tag{1.7}$$

which is precisely the mean of the discrete *PMF*.

Also, as mentioned, 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. We may also use the extended notation: $\mathcal{N}(x : m, v)$ to include the random variable x as well. When *sampling* we use the *non-conventional* right arrow "action" notation $\rightarrow$ to mean "draw a sample from" a particular distribution such as $x_i \rightarrow \Pr(x)$—this again will be clear from the context. When *resampling*, that is, replacing samples with new ones we use the "block" right arrow such as $x_j \Rightarrow x_i$ meaning new sample x_j replaces current sample x_i.

Finally in a discrete (finite) probabilistic representation, we define a purely discrete variate as $x_k(t) := \Pr(x(t) = \mathcal{X}_k)$ meaning that x can only take on values (integers) k from a known set $\mathcal{X} = \{\mathcal{X}_1, \ldots, \mathcal{X}_k, \ldots, \mathcal{X}_N\}$ at time t. We also use the symbol, $\triangle\triangle\triangle$ to mark the end of an example.

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. Candy, *Signal Processing: The Model-Based Approach* (New York: McGraw-Hill, 1986).
2. J. Candy, *Model-Based Signal Processing* (Hoboken, NJ: Wiley/IEEE Press, 2006).
3. R. Duda, P. Hart and D. Stork, *Pattern Classification* (Hoboken, NJ: Wiley/IEEE Press, 2001).
4. J. Tringe, D. Clague, J. Candy and C. Lee, "Model-based signal processing of multichannel cantilever arrays," *IEEE J. Micromech. Syst.*, **15**, 5, 1371–1391, 2006.
5. S. Ulam, R. Richtmyer and J. von Neumann, "Statistical methods in neutron diffusion," *Los Alamos Scientific Laboratory Report*, **LAMS-551**, 1947.
6. N. Metropolis and S. Ulam, "The Monte Carlo method," *J. American Stat. Assoc.*, **44**, 335–341, 1949.
7. N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller, "Equations of state calculations by fast computing," *J. Chemical Physics*, **21**, 6, 1087–1091, 1953.
8. W. Hastings, "Monte Carlo sampling methods using Markov chains and their applications," *Biometrika*, **57**, 1, 97–109, 1970.
9. N. Metropolis, "The beginning of the Monte Carlo method," *Los Alamos Science*, Special Issue, 125–130, 1987.
10. J. Liu, *Monte Carlo Strategies in Scientific Computing* (New York: Springer-Verlag, 2001).
11. D. Frenkel, "Introduction to Monte Carlo methods," in *Computational Soft Matter: From Synthetic Polymers to Proteins*, N. Attig, K. Binder, H. Grubmuller and K. Kremer (Eds.) J. von Neumann Instit. for Computing, Julich, NIC Series, Vol. 23, pp. 29–60, 2004.
12. C. Robert and G. Casella, *Monte Carlo Statistical Methods* (New York: Springer, 1999).
13. M. Tanner, *Tools for Statistical Inference: Methods for the Exploration of Posterior Distributions and Likelihood Functions, 2nd Ed.* (New York: Springer-Verlag, 1993).
14. J. Ruanaidh and W. Fitzgerald, *Numerical Bayesian Methods Applied to Signal Processing* (New York: Springer-Verlag, 1996).
15. W. Gilks, S. Richardson and D. Spiegelhalter, *Markov Chain Monte Carlo in Practice* (New York: Chapman & Hall/CRC Press, 1996).
16. A. Doucet, N. de Freitas and N. Gordon, *Sequential Monte Carlo Methods in Practice* (New York: Springer-Verlag, 2001).

17. B. Ristic, S. Arulampalam and N. Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Applications* (Boston: Artech House, 2004).
18. O. Cappe, E. Moulines and T. Ryden, *Inference in Hidden Markov Models* (New York: Springer-Verlag, 2005).
19. S. Godsill and P. Djuric, "Special Issue: Monte Carlo methods for statistical signal processing." *IEEE Trans. Signal Proc.*, **50**, 173–499, 2002.
20. 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.
21. S. Haykin and N. de Freitas, "Special Issue: Sequential state estimation: from Kalman filters to particle filters." *Proc. IEEE*, **92**, 3, 399–574, 2004.
22. A. Doucet and X. Wang, "Monte Carlo methods for signal processing," *IEEE Signal Proc. Mag.*, **24**, 5, 152–170, 2005.
23. J. Candy, "Bootstrap particle filtering for passive synthetic aperture in an uncertain ocean environment." *IEEE Signal Proc. Mag.*, **24**, 4, 73–85, 2007.

PROBLEMS

1.1 Estimate the number of times a needle when dropped between two parallel lines intersects a line. One was to accomplish this is experimentally by setting up the experiment and doing it—this is the famous Buffon's needle experiment performed in 1725.

(*a*) Set up the experiment and perform the measurements for 100 samples. Estimate the underlying probabilities.

(*b*) Analyze the experiment using a "closed form" approach.

(*c*) How do your answers compare?

Note that this is one of the first Monte Carlo approaches to problem solving.

1.2 Suppose we have three loaded dice with the following six "face" probabilities (each):

$$D1 = \left\{\frac{1}{12}, \frac{1}{6}, \frac{1}{12}, \frac{1}{3}, \frac{1}{6}, \frac{1}{6}\right\}$$

$$D2 = \left\{\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{12}, \frac{1}{12}, \frac{1}{3}\right\}$$

$$D3 = \left\{\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{12}, \frac{1}{12}, \frac{1}{3}\right\}$$

Applying Bayes' rule, answer the following questions:

(*a*) Selecting a die at random from the three, what is the probability of rolling a 6?

(*b*) What is the probability that die two ($D = D2$) was selected, if a six ($R = 6$) is rolled with the chosen die?

1.3 A binary communication transmitter (T) sends either a 0 or a 1 through a channel to a receiver (R) with the following probabilities for each as:

$$\begin{aligned} \Pr(T_1) &= 0.6 \quad \Pr(T_0) &&= 0.4 \\ \Pr(R_1|T_1) &= 0.9 \quad \Pr(R_0|T_1) &&= 0.1 \\ \Pr(R_1|T_0) &= 0.1 \quad \Pr(R_0|T_0) &&= 0.9 \end{aligned}$$

(a) What is the probability that R_1 is received?

(b) What is the probability that R_0 is received?

(c) What is the probability that the true transmitted signal was a 1, when a 1 was received?

(d) What is the probability that the true transmitted signal was a 0, when a 0 was received?

(e) What is the probability that the true transmitted signal was a 1, when a 0 was received?

(f) What is the probability that the true transmitted signal was a 0, when a 1 was received?

(g) Draw a probabilistic directed graph with nodes being the transmitters and receivers and links being the corresponding prior and conditional probabilities?

1.4 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, 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.

1.5 Think of measuring the temperature of a liquid in a beaker heated by a burner. Suppose 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 we model the heat transferred to the liquid from the burner as

$$Q(t) = CA\ \triangle T(t)$$

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.

1.6 We are given an *RLC* series circuit 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, $V_{out}(t)$ (measurement model).

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

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

1.7 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 θ is the angle of the satellite axis with respect to the inertial reference (no angular acceleration) A. Develop signal and noise models for this problem and identify each model-based processor component.

1.8 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

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

and the controller reference signal r is given by

$$r = e + c$$

Suppose 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.

1.9 The elevation of a tracking telescope is controlled by a DC motor. It 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 θ 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.

1.10 Suppose 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 $y = [7 \quad 21]'$?

(*b*) What is the batch weighted least-squares estimate of the parameter x with W selected for minimum variance estimation?

1.11 Calculate the batch and sequential least-squares estimate of the parameter vector x based on two measurements $y(1)$ and $y(2)$ where

$$\begin{aligned} y(1) &= C(1)x + v(1) = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \\ y(2) &= c'x + v(2) = 4 \\ C &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad c'(1) = [1 \quad 2], \quad W = I \end{aligned}$$

2

BAYESIAN ESTIMATION

2.1 INTRODUCTION

In this chapter we motivate the idea of Bayesian estimation from probabilistic perspective, that is, we perform the required estimation using the underlying densities or mass functions. We start with the "batch" approach and evolve to the Bayesian sequential techniques. We discuss the most popular formulations: maximum *a posteriori* (*MAP*), maximum likelihood (*ML*), minimum variance (*MV*) or equivalently minimum mean-squared error (*MMSE*) and least-squares (*LS*) methods. Bayesian sequential techniques are then developed. The main idea is to develop the proper perspective for the subsequent chapters and construct a solid foundation for the techniques to follow.

2.2 BATCH BAYESIAN ESTIMATION

Suppose we are trying to estimate a random parameter X from data $Y = y$. Then the associated conditional density $\Pr(X|Y = y)$ is called the *posterior density* because the estimate is conditioned "after (*post*) the measurements" have been acquired. Estimators based on this *a posteriori* density are usually called *Bayesian* because they are constructed from Bayes' theorem, since $\Pr(X|Y)$ is difficult to obtain directly. That is, *Bayes' rule* is defined

$$\Pr(X|Y) := \Pr(Y|X)\frac{\Pr(X)}{\Pr(Y)} \tag{2.1}$$

where $\Pr(X)$ is called the *prior density* (before measurement), $\Pr(Y|X)$ is called the likelihood (more likely to be true) and $\Pr(Y)$ is called the *evidence* (normalizes the

Bayesian Signal Processing. By James V. Candy

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 as depicted in Fig. 1.1.

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 $\Pr(X|Y)$—its *maximum* [1]. The *maximum a posteriori* (*MAP*) *estimate* is the value of x that maximizes the posterior density, that is,

$$\max_X \Pr(X|Y) \tag{2.2}$$

The optimization is carried out in the usual manner by differentiating, setting the result to zero and solving to obtain the *MAP equation*

$$\nabla_X \Pr(X|Y)\big|_{X=\hat{X}_{MAP}} = 0 \tag{2.3}$$

with the *gradient vector* $\nabla_X \in R^{N_X \times 1}$ defined by

$$\nabla_X := \left[\frac{\partial}{\partial X_1} \cdots \frac{\partial}{\partial X_{N_X}}\right]' \tag{2.4}$$

Because many problems are based on the exponential class of densities, the $\ln \Pr(X|Y)$ is considered instead. Since the logarithm is a monotonic function, the maximum of $\Pr(X|Y)$ and $\ln \Pr(X|Y)$ occur at the same value of X. Therefore, the logarithmic *MAP equation* is

$$\nabla_X \ln \Pr(X|Y)\big|_{X=\hat{X}_{MAP}} = 0 \tag{2.5}$$

Now if we apply Bayes' rule to Eq. 2.5, then

$$\ln \Pr(X|Y) = \ln \Pr(Y|X) + \ln \Pr(X) - \ln \Pr(Y) \tag{2.6}$$

Since $\Pr(Y)$ is not a function of the parameter X, the *MAP* equation can be written succinctly as

$$\nabla_X \ln \Pr(X|Y)\big|_{X=\hat{X}_{MAP}} = \nabla_X (\ln \Pr(Y|X) + \ln \Pr(X))\big|_{X=\hat{X}_{MAP}} = 0 \tag{2.7}$$

With a variety of estimators available, we must construct some way of ascertaining performance. The quality of an estimator is usually measured in terms of its *estimation error*,

$$\tilde{X} = X - \hat{X} \tag{2.8}$$

A common measure of estimator quality is called the Cramer-Rao lower bound (*CRLB*). The *CRLB* offers a means of assessing estimator quality prior to processing the measured data. We restrict discussion of the *CRLB* to the case of unbiased estimates, $\hat{X}$, of a "non-random" parameter X. The bound is easily extended to more complex cases for biased estimates as well as random parameters [2, 3]. The *Cramer-Rao lower bound*[1] for any unbiased estimate $\hat{X}$ of X based on the measurement, Y, is given by

$$R_{\tilde{X}|X} = \text{cov}(X - \hat{X}(Y)|X = x) \geq \mathcal{I}^{-1} \tag{2.9}$$

where $\mathcal{I}$ the $N_X \times N_X$ *information matrix* defined by

$$\mathcal{I} := -E_y\{\nabla_X(\nabla_X \ln \Pr(Y|X))'\} \tag{2.10}$$

with the gradient vector defined above. Any estimator satisfying the *CRLB* with equality is called *efficient*. The bound is easily calculated using the *chain rule* from vector calculus [5] defined by

$$\nabla_X(a'b) := (\nabla_X a')b + (\nabla_X b')a \quad a, b \in R^{N_X \times 1} \tag{2.11}$$

where a, b are functions of X. Consider the following example illustrating the calculation of the *CRLB*.

Example 2.1

Suppose we would like to estimate a nonrandom but unknown parameter, X, from a measurement y contaminated by additive Gaussian noise, that is,

$$y = X + v$$

where $v \sim \mathcal{N}(0, R_{vv})$ and X is unknown. Thus, we have that

$$E\{Y|X\} = E\{X + v|X\} = X$$

and

$$\text{var}(Y|X) = E\{(y - E\{Y|X\})^2|X\} = E\{v^2|X\} = R_{vv}$$

which gives

$$\Pr(Y|X) \sim \mathcal{N}(X, R_{vv})$$

and therefore

$$\ln \Pr(Y|X) = -\frac{1}{2}\ln(2\pi R_{vv}) - \frac{1}{2}\frac{(y - X)^2}{R_{vv}}$$

[1] We choose the matrix-vector version, since parameter estimators are typically vector estimates.

Differentiating according to the chain rule of Eq. 2.11 and taking the expectation we obtain

$$\mathcal{I} = -E\left\{\frac{\partial^2}{\partial X^2}\ln \Pr(Y|X)\right\} = -E\left\{\frac{\partial}{\partial X}\frac{(y-X)}{R_{vv}}\right\} = \frac{1}{R_{vv}}$$

and therefore the *CRLB* is

$$R_{\tilde{X}|X} \geq R_{vv} \qquad \triangle\triangle\triangle$$

The utility of the *CRLB* 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 a designed estimator is efficient, that is, any estimator achieving the *CRLB* with equality 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". Next we consider the case when the parameter X is not random leading to the maximum likelihood estimator.

2.3 BATCH MAXIMUM LIKELIHOOD ESTIMATION

In contrast to the Bayesian approach, the likelihood method views the parameter as deterministic but *unknown*. We include it in the Bayesian discussion because as we will show both estimators are in fact intimately linked. Maximum likelihood 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 $\mathbf{X}$ be a vector of unknown parameters, $\mathbf{X} \in R^{N_X \times 1}$ and the corresponding set of *N-conditionally independent* measurements, $Y(N) := \{\mathbf{y}(1) \cdots \mathbf{y}(N)\}$ for $\mathbf{y} \in R^{N_y \times 1}$. The *likelihood* of X given the measurements is defined to be proportional to the value of the probability density of the measurements given the parameters, that is,

$$\mathcal{L}(Y(N); X) \propto \Pr(Y(N)|X) = \Pr(\mathbf{y}(1)\ldots\mathbf{y}(N)|X) = \prod_{i=1}^{N} \Pr(\mathbf{y}(i)|X) \qquad (2.12)$$

where $\mathcal{L}$ is the likelihood function and $\Pr(Y|X)$ 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 X, say X_i, then we select the value of $\hat{X}$ that most likely specifies the underlying probability function that the measurement sample was drawn from, that is, suppose we have two estimates, $\hat{X}_i$ and $\hat{X}_j$ for which

$$\Pr(Y|X_i) > \Pr(Y|X_j) \tag{2.13}$$

Thus, it is "more likely" that the $Y(N)$ were drawn for parameter value $\hat{X}_i$ than $\hat{X}_j$, since, equivalently, $\mathcal{L}(Y;\hat{X}_i) > \mathcal{L}(Y;\hat{X}_j)$. Searching over all X and selecting that value of X that is maximum (most probable) leads to the maximum likelihood estimate (*ML*) given by

$$\hat{X}_{ML}(Y) = \arg\max_X \Pr(Y|X) \tag{2.14}$$

As noted previously, many problems are characterized by the class of exponential densities for the Bayesian estimator making it more convenient to use the natural logarithm function; therefore, we define the *log-likelihood function* as

$$\Lambda(Y(N)|X) := \ln \mathcal{L}(Y;X) = \ln \Pr(Y(N)|X) \tag{2.15}$$

Since the logarithm is monotonic, it preserves the maximum of the likelihood providing the identical result,

$$\hat{X}_{ML}(Y) = \arg\max_X \ \ln \Pr(Y|X) \tag{2.16}$$

What makes the *ML* estimator popular is the fact that it enjoys some very desirable properties that we list without proof (see [2] for details).

1. *ML* estimates are *consistent*.
2. *ML* estimates are *asymptotically efficient* with $R_{\tilde{X}|X} = \mathcal{I}^{-1}$.
3. *ML* estimates are *asymptotically Gaussian* with $\mathcal{N}(X, R_{\tilde{X}\tilde{X}})$.
4. *ML* estimates are *invariant*, that is, if $\hat{X}_{ML}$, then any function of the *ML* estimate is the *ML* estimate of the function, $\hat{f}_{ML} = f(\hat{X}_{ML})$.
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. 2.6 and ignore the *a priori* distribution $\Pr(X)$ (assume X 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 $\ln \Pr(X|Y)$ instead of $\Pr(X|Y)$.

We obtain the maximum likelihood estimate by solving the *log-likelihood equation* and checking for the existence of a maximum; that is,

$$\nabla_X \ln \Pr(X|Y)\big|_{X=\hat{X}_{ML}} = 0 \tag{2.17}$$

Of course, to check for a maximum we have that $\nabla_X(\nabla_X \ln \Pr(X|Y)') < 0$. Again applying Bayes' rule as in Eq. 2.1 and ignoring $\Pr(X)$, we have

$$\nabla_X \ln \Pr(X|Y) = \nabla_X \ln \Pr(Y|X)\big|_{X=\hat{X}_{ML}} = 0. \tag{2.18}$$

Consider the following example to demonstrate this relationship between *MAP* and *ML*.

Example 2.2

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 $v \sim N(0, R_{vv})$ and the measurement model is given by

$$y = X + v \tag{2.19}$$

First, we assume no *a priori* statistical knowledge of X just that it is an unknown, nonrandom constant. Thus, we require the maximum likelihood estimate, since no prior information is assumed about $\Pr(X)$. The associated conditional density is

$$\Pr(Y|X) = \frac{1}{\sqrt{2\pi R_{vv}}} e^{-\frac{1}{2}\frac{(y-X)^2}{R_{vv}}} \tag{2.20}$$

The maximum likelihood estimate of X is found by solving the log-likelihood equation:

$$\nabla_X \ln \Pr(Y|X)\big|_{X=\hat{X}_{ML}} = 0 \tag{2.21}$$

or

$$\begin{aligned} \frac{\partial}{\partial X} \ln \Pr(Y|X) &= \frac{\partial}{\partial X}\left\{-\frac{1}{2}\ln 2\pi R_{vv} - \frac{1}{2R_{vv}}(y-X)^2\right\} \\ &= \frac{1}{R_{vv}}(y-X) \end{aligned}$$

Setting this expression to zero and solving for X, we obtain

$$\hat{X}_{ML} = y \tag{2.22}$$

The best estimate of X in a maximum likelihood sense is the raw data y. The corresponding error variance is easily calculated (as before)

$$R_{\tilde{X}|X} = R_{vv} \tag{2.23}$$

Next we model X as a random variable with Gaussian prior, that is, $X \sim N(\overline{X}, R_{XX})$ and desire the maximum *a posteriori* estimate. The *MAP* equation is

$$\mathcal{J}_{MAP} = \nabla_X(\ln \Pr(Y|X) + \ln \Pr(X))$$

$$\mathcal{J}_{MAP} = \frac{\partial}{\partial X}\left\{-\frac{1}{2}\ln 2\pi R_{vv} - \frac{1}{2R_{vv}}(y-X)^2 - \frac{1}{2}\ln 2\pi R_{XX} - \frac{\frac{1}{2}(X-\overline{X})^2}{R_{XX}}\right\}$$

or

$$\mathcal{J}_{MAP} = \frac{1}{R_{vv}}(y-X) - \frac{1}{R_{XX}}(X-\overline{X})$$

Setting this expression to zero and solving for $X = \hat{X}_{MAP}$, we obtain

$$\hat{X}_{MAP} = \frac{y + \frac{R_{vv}}{R_{XX}}\overline{X}}{1 + \frac{R_{vv}}{R_{XX}}}$$

It can be shown from Eq. 2.9 that the corresponding error variance is

$$R_{\tilde{X}|X} = \frac{R_{vv}}{1 + \frac{R_{vv}}{R_{XX}}}$$

Examining the results of this example, we see that when the parameter variance is large ($R_{XX} \gg R_{vv}$), 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. △△△

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 the above example, if X were actually Gaussian, then the *ML* solution, which models X as an unknown parameter, is not an efficient estimator, while the *MAP* solution that incorporates this information by using the prior $\Pr(X)$ is efficient.

This completes the introduction to batch Bayesian estimation using the maximum *a posteriori* and maximum likelihood estimation. Next we consider a very popular approach to solving maximum likelihood estimation problems.

2.3.1 Expectation-Maximization Approach to Maximum Likelihood

Solving maximum likelihood parameter estimation problems is a formidable task especially when the underlying probability distribution functions are unknown. Therefore, we must resort to numerical approaches that will successfully converge to the parameters of interest. Expectation-Maximization (*EM*) is a general method of determining the maximum likelihood estimate of parameters of the underlying distribution

from a given set of data which is incomplete, that is, having "missing" (data) values [7]. Missing data could be considered a misnomer; however, if we include "hidden variables" (not directly measured) as missing, then a wide variety of state/parameter estimation problems can be incorporated into these problem classes. Probably the most popular applications of the *EM* technique occur in tomographic image reconstruction, pattern recognition, communications and the training of hidden Markov models (see Chapter 9) for speech recognition [8, 9].

The *EM* technique produces maximum-likelihood parameter estimates in two steps: an expectation-step followed by a maximization-step. The expectation-step with respect to the unknown parameters uses the most recently available parameter estimate conditioned on the measurements, while the maximization-step provides an updated estimate of the parameters.

To be more precise, we mathematically, formulate the general "missing data" problem by first defining the unknown parameters or variables to be estimated as $\theta \in \mathcal{R}^{N_\theta \times 1}$ with $\theta \in \Theta$, the parameter space. We further define three distinct spaces for our problem: (1) the *complete* data space, $\mathcal{Z}$; (2) the *incomplete* data space, $\mathcal{Y}$; and (3) the *missing* data space, $\mathcal{X}$, where the complete space is the union: $\mathcal{Z} = (\mathcal{X}, \mathcal{Y})$. Analogously, we define the corresponding complete, incomplete and missing/hidden vectors as: $\mathbf{z} \in \mathcal{R}^{N_z \times 1}$, $\mathbf{y} \in \mathcal{R}^{N_y \times 1}$ and $\mathbf{x} \in \mathcal{R}^{N_x \times 1}$, respectively.

With this in mind, we can now define the underlying distributions as the joint or *complete* distribution along with its Bayesian decompositions as

$$\Pr(\mathbf{z}|\theta) = \Pr(\mathbf{x}, \mathbf{y}|\theta) = \Pr(\mathbf{x}|\mathbf{y}, \theta) \times \Pr(\mathbf{y}|\theta) = \Pr(\mathbf{y}|\mathbf{x}, \theta) \times \Pr(\mathbf{x}|\theta) \tag{2.24}$$

or taking logarithms, we have the *complete* (data) log-likelihood

$$\Lambda_{CD}(\mathbf{z}|\theta) = \ln \Pr(\mathbf{z}|\theta) = \ln \Pr(\mathbf{x}, \mathbf{y}|\theta) = \ln \Pr(\mathbf{x}|\mathbf{y}, \theta) + \Lambda_{ID}(\mathbf{y}|\theta) \tag{2.25}$$

where $\Lambda_{ID}(\mathbf{y}|\theta) = \ln \Pr(\mathbf{y}|\theta)$ is the corresponding *incomplete* (data) log-likelihood and the missing data is *random* with $\mathbf{x} \sim \Pr(\mathbf{x})$. Since $\mathbf{x}$ is random, then so is $\Lambda_{CD}(\mathbf{z}|\theta)$ with $\mathbf{y}$ and θ assumed fixed (constant). Thus, the basic maximum likelihood parameter estimation problem for the complete data is to find the value of the parameter such that

$$\hat{\theta}_i = \arg\max_{\theta} \Lambda_{CD}(\mathbf{z}|\theta)$$

given the measured or observed (incomplete) data, $\mathbf{y}$, and the previous parameter estimate, $\theta = \hat{\theta}$. However, since Λ_{CD} is random, we must search for the parameter vector that maximizes its expectation (over $\mathbf{x}$), that is, we seek

$$\hat{\theta}_i = \arg\max_{\theta} E_{\mathbf{x}}\{\Lambda_{CD}(\mathbf{z}|\theta)\} \tag{2.26}$$

given the measured data, $\mathbf{y}$, and the current parameter estimate, θ. Multiplying both sides of Eq. 2.25 by the underlying marginal posterior distribution of the missing

data, $\Pr(\mathbf{x}|\mathbf{y},\theta)$ and summing over $\mathbf{x}$, we obtain

$$\begin{aligned}\sum_x \Lambda_{CD}(\mathbf{z}|\theta) \times \Pr(\mathbf{x}|\mathbf{y},\theta) &= \sum_x \ln \Pr(\mathbf{z}|\theta) \times \Pr(\mathbf{x}|\mathbf{y},\theta) \\ &= \sum_x \ln \Pr(\mathbf{x}|\mathbf{y},\theta) \times \Pr(\mathbf{x}|\mathbf{y},\theta) \\ &\quad + \sum_x \Lambda_{ID}(\mathbf{y}|\theta) \times \Pr(\mathbf{x}|\mathbf{y},\theta)\end{aligned}$$

Using the definition of the conditional expectation and recognizing that the last term is not a function of the random vector $\mathbf{x}$, we obtain

$$E_{\mathbf{x}}\{\Lambda_{CD}(\mathbf{z}|\theta)|\mathbf{y},\hat{\theta}\} = E_{\mathbf{x}}\{\ln \Pr(\mathbf{x}|\mathbf{y},\hat{\theta})\} + \Lambda_{ID}(\mathbf{y}|\hat{\theta}) \tag{2.27}$$

Since we do not know the complete data, we cannot calculate the exact log-likelihood for this problem. But, given the measured data $\mathbf{y}$, we can estimate the posterior probability for the missing (data) variables, $\mathbf{x}$. For each $\mathbf{x}$, there exists a $\hat{\theta}$, and therefore we can calculate an expected value of the complete log-likelihood.

The basic *EM* principle is to find the θ that maximizes $\Pr(\mathbf{z}|\theta)$ using the available data $\mathbf{y}$ and current parameter estimate. Let $\hat{\theta}_{i-1}$ be the current parameter estimate, then the complete log-likelihood is given by the expectation-step:

$$\text{E-step:} \quad Q(\theta,\hat{\theta}_{i-1}) := E_x\{\Lambda_{CD}(\mathbf{z}|\theta)|\mathbf{y},\hat{\theta}_{i-1}\} \tag{2.28}$$

for θ the new parameter vector to be optimized in the next step. In this expression the $\mathbf{y}$ and $\hat{\theta}$ are assumed fixed (constant) and $\mathbf{x}$ is the random vector such that $\mathbf{x} \sim \Pr(\mathbf{x}|\mathbf{y},\hat{\theta}_{i-1})$ so that

$$E_x\{\Lambda_{CD}(\mathbf{z}|\theta)|\mathbf{y},\hat{\theta}_{i-1}\} = \sum_{\mathbf{x}} \ln \Pr(\mathbf{x}|\mathbf{y},\theta)\Pr(\mathbf{x}|\mathbf{y},\hat{\theta}_{i-1}) \tag{2.29}$$

where $\Pr(\mathbf{x}|\mathbf{y},\hat{\theta}_{i-1})$ is the marginal of the missing data (hidden variable) based on the measured data and current parameter estimate (as shown).

The maximization-step is to find the parameter vector update, $\hat{\theta}_i$ that will maximize the computed expectation, that is,

$$\text{M-step:} \quad \hat{\theta}_i = \arg\max_{\theta} Q(\theta,\hat{\theta}_{i-1}) \tag{2.30}$$

Each iteration is guaranteed to *increase* the log-likelihood eventually converging to a local maximum at an exponential rate [10, 11]. Different forms of the *EM* have evolved with the "generalized" *EM* (*GEM*) method by finding an alternative (simpler) expectation function and using the updated parameter vector such that $Q(\hat{\theta}_i,\hat{\theta}_{i-1}) > Q(\theta,\hat{\theta}_{i-1})$ which is also guaranteed to converge. The Monte Carlo *EM* (*MCEM*) is another form that is used when a closed form distribution for the

E-step is replaced by simulation-based methods (sampling) [12]. Consider the following example from Snyder [16].

Example 2.3

Suppose we would like to estimate the rate or intensity parameter, λ_s, of a signal contaminated in Poisson noise with known mean, λ_v. We measure the counts from a photodetector characterized by

$$y_n = s_n + v_n$$

where y_n is the observation with Poisson distribution

$$y_n \sim \mathcal{P}(\lambda_y) = (\lambda_y)^{y_n} e^{-\lambda_y}/y_n!$$

The respective signal and noise counts during the measurement period are independent of each other with $s_n \sim \mathcal{P}(\lambda_s)$ and $v_n \sim \mathcal{P}(\lambda_v)$. We have that the log-likelihood of the incomplete data is

$$\Lambda_{ID}(y_n|\lambda_y) = \ln \mathcal{P}(y_n|\lambda_y) = y_n \ln \lambda_y - \lambda_y - \ln y_n!$$

Differentiating Λ_{ID} with respect to λ_s, setting to the result to zero and solving for the rate parameter, that is,

$$\frac{d}{d\lambda_s}(y_n \ln(\lambda_s + \lambda_v) - (\lambda_s + \lambda_v) - \ln y_n!) = \frac{y_n}{\lambda_s + \lambda_v} - 1 = 0$$

which yields the maximum likelihood parameter estimate

$$\hat{\lambda}_s = y_n - \lambda_v > 0$$

Here we have used the fact that the superposition of Poisson processes is Poisson with intensity parameter, $\lambda_y = \lambda_s + \lambda_v$. Next, let us investigate the *EM* approach to this problem. Here the complete data space is $z_n = (s_n, v_n)$ and y_n is the incomplete (observed) data. Therefore, we have that the complete log-likelihood is

$$\Lambda_{CD}(z_n|\lambda_s) = -(\lambda_s + \lambda_v) + s_n \ln \lambda_s + v_n \ln \lambda_v - \ln s_n! - \ln v_n!$$

because s_n and v_n are independent. Thus, the expectation-step is given by

$$\text{E-step:} \quad Q(\lambda_s|\hat{\lambda}_s(i-1)) = -\lambda_s + \hat{s}_n(i-1) \ln \lambda_s - \lambda_v + \hat{v}_n(i-1) \ln \lambda_v$$

with

$$\hat{s}_n(i-1) = E\{s_n|y_n, \hat{\lambda}_s(i-1)\} = \frac{\hat{\lambda}_s(i-1)}{\hat{\lambda}_s(i-1) + \lambda_v} y_n$$

and

$$\hat{v}_n(i-1) = E\{v_n|y_n, \hat{\lambda}_v(i-1)\} = y_n - \hat{s}_n(i-1)$$

Since the maximization-step does not depend on λ_v or $\hat{v}_n(i-1)$, we have

$$\hat{\lambda}_s(i) = \arg\max_{\lambda_s}(-\lambda_s + \hat{s}_n(i-1)\ln\lambda_s)$$

giving

$$\text{M-step:} \quad \hat{\lambda}_s(i) = \hat{s}_n(i-1)$$

which completes the *EM* algorithm.

We simulated a sequence of Poisson counts for 500 samples composed of the additive signal ($\lambda_s = 14$) and noise ($\lambda_N = 3.5$). The estimated signal intensity at each measurement is shown in Fig. 2.1a along with the estimated probability mass

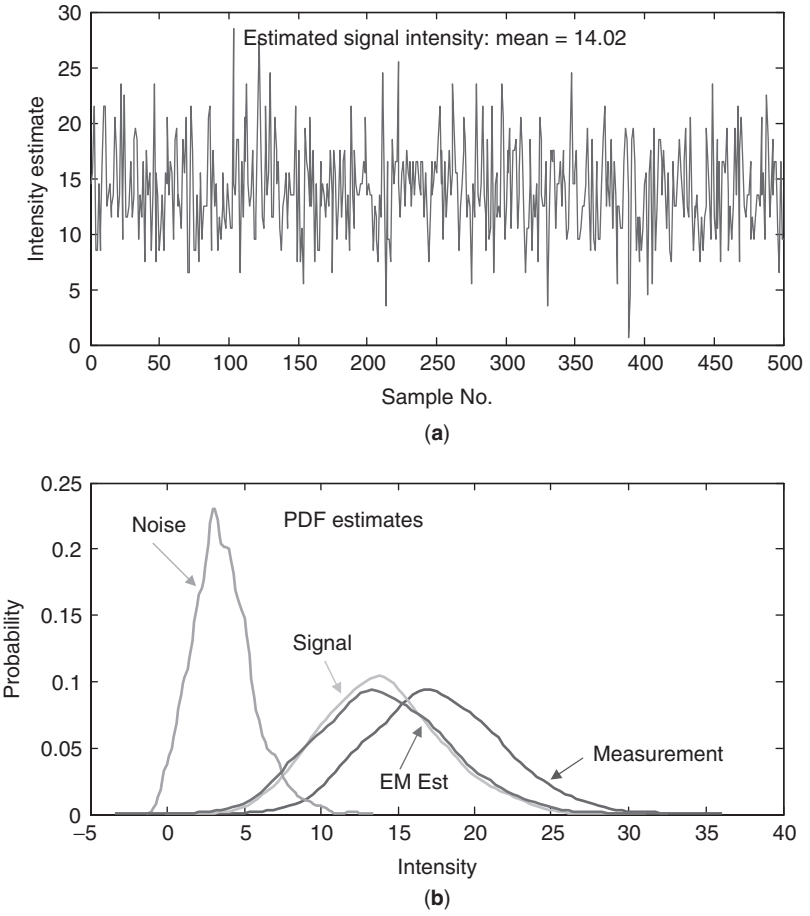


FIGURE 2.1 EM estimation for Poisson Intensity: (a) Signal intensity estimate (Mean = 14.02). (b) PDF estimation of Poisson processes: Noise, Signal, EM Estimate, Measurement.

functions in b of the True signal, EM estimated signal, measurement and noise. Here we see the estimated signal PDF closely matches the true PDF and the average intensity is very close to the true signal intensity at $\lambda_s = 14.02$. △△△

In summary, the *EM* approach is a two-step, iterative technique to solve the maximum likelihood parameter estimation problem when missing data or hidden variables (states) are present. Clearly, we have just discussed the generic form of the *EM* approach, the actual algorithm is problem specific based on the knowledge of the underlying distributions; however, we can consider a class of distributions to develop a more specific algorithm such as the so-called *exponential family* [13].

2.3.2 EM for Exponential Family of Distributions

In this subsection we apply the *EM* technique to the exponential class of distribution functions many of which are well-known forms like the Gaussian, Poisson, Exponential, Raleigh, Binomial and more. The exponential family is defined by the generic form:

$$\Pr(\mathbf{z}|\theta) = \mathbf{b}(\mathbf{z})\exp(\mathbf{c}'(\theta)\mathbf{s}(\mathbf{z}))/a(\theta) \tag{2.31}$$

with $\theta \in \mathcal{R}^{N_\theta}$, the parameters defining the class [7, 14, 15] and $\mathbf{s}(\mathbf{z})$ the *sufficient* statistic providing all of the information necessary about the underlying process for estimation with $\mathbf{s}, \mathbf{c} \in \mathcal{R}^{N_s \times 1}$. Since the complete log-likelihood can be written as

$$\Lambda_{CD}(\mathbf{z}|\theta) = \ln \Pr(\mathbf{z}|\theta) = \ln \mathbf{b}(\mathbf{z}) + \mathbf{c}'(\theta)\mathbf{s}(\mathbf{z}) - \ln a(\theta) \tag{2.32}$$

then taking the conditional expectations, we obtain the expectation-step as:

$$\text{E-step:} \quad Q(\theta, \hat{\theta}_{i-1}) = E_{\mathbf{x}}\{\ln \mathbf{b}(\mathbf{z})|\mathbf{y}, \theta) + \mathbf{c}'(\theta)E\{\mathbf{s}(\mathbf{z})|\mathbf{y}, \hat{\theta}_{i-1}\} - \ln a(\theta) \tag{2.33}$$

Defining $\hat{\mathbf{s}}_i := E\{\mathbf{s}(\mathbf{z})|\mathbf{y}, \hat{\theta}_{i-1}\}$, the maximization-step with respect to θ, is performed on

$$E\{\ln \mathbf{b}(\mathbf{z})|\mathbf{y}, \theta) + \mathbf{c}'(\theta)\hat{\mathbf{s}}_i - \ln a(\theta) \tag{2.34}$$

Since the first term is not a function of θ, it need not be included. The *EM* technique for the exponential family is:

$$\begin{aligned} &\text{E-step:} \quad \hat{\mathbf{s}}_i \\ &\text{M-step:} \quad \hat{\theta}_i = \arg\max_{\theta} \mathbf{c}'(\theta)\hat{\mathbf{s}}_i - \ln a(\theta) \end{aligned} \tag{2.35}$$

completing the method.

For instance, in the previous example, we have $\mathbf{b}(\mathbf{z}) = (\lambda_y)^{y_n}$; $\exp(\mathbf{c}'\mathbf{s}) = e^{-\lambda_y}$ and $a(\theta) = y_n!$. Consider the following example of producing an image by estimating the intensity of a Poisson process discussed in McLachlan [6].

Example 2.4

Photons emitted from a radioactive tracer are injected into tissue in order to create a single photon emission computed tomography (SPECT) image for medical diagnosis [6, 14–16]. The basic idea is to create an image that represents the photon emitted, counted by photon detectors and displayed as an image based on pixel counts. For simplicity we assume all photons are counted by the instrument. The emissions are assumed to be Poisson distributed with unknown intensities or *emission densities* corresponding to the Poisson rate parameters, λ. Thus, the basic problem is:

GIVEN a set of incomplete measurements (counts), $\{y_n\}$, **FIND** the maximum likelihood estimate of the unknown emission densities, λ, (rate parameter vector) assuming that each of these component intensities is constant within a pixel, that is, λ_m is constant for $m = 1, \ldots, M_p$.

Let y_n; $n = 1, \ldots, N_d$ be the number of counts measured by the n^{th}-detector, so mathematically the problem is to estimate the M_p-intensity vector of emission densities, $\hat{\lambda}$, from the N_d-measurement vector, $\mathbf{y}$, assuming that the counts are conditionally independent such that $\Pr(y_n|\lambda_y(n))$ is Poisson, that is,

$$\mathbf{y}_n \sim \mathcal{P}(\lambda_y(n)) = \frac{(\lambda_y(n))^{y_n}\, e^{-\lambda_y(n)}}{y_n!}$$

For imaging considerations, it is assumed that individual photons are emitted within a given pixel and detected; therefore, we define x_{nm} as the unobservable counts (missing data) assuming λ is known and conditionally independent of x, such that,

$$x_{mn} \sim \mathcal{P}(\lambda_x(m,n)) = \frac{(\lambda_x(m,n))^{x_{nm}}\, e^{-\lambda_x(m,n)}}{x_{nm}!}$$

where $\lambda_x(m,n) = \lambda_m p_{mn}$, the emission density corresponding to the number of photons emitted in the m^{th}-pixel detected by the n^{th}-detector with emission probability, p_{mn}—a marked Poisson process [16, 18].

The photon counter measurement at the n^{th}-detector is the superposition of the photons emitted by the radionuclide at the m^{th}-pixel; therefore, we have that

$$y_n = \sum_{m=1}^{M_p} x_{mn}$$

which is the sum of Poisson variates, so that

$$\lambda_y(n) = \sum_{m=1}^{M_p} \lambda_x(m,n) = \sum_{m=1}^{M_p} \lambda_m p_{mn}$$

It has been shown [6] that the conditional probability of the missing data is binomial with parameter, y_n and probability $\Pr(\lambda_m p_{mn})$, that is,

$$x_{mn} \sim \mathcal{B}(y_n, \Pr(\lambda_m p_{mn})) \quad \text{with } \Pr(\lambda_m p_{mn}) = \lambda_m p_{mn} \Big/ \sum_{j=1}^{M_p} \lambda_i p_{jn}$$

The *EM* approach for this problem is straightforward, since the underlying distributions are all in the exponential family [6]. The sufficient statistic is the data, x_{mn}, which can be estimated. That is, since x_{mn} is binomial, the conditional mean (Q-step) is given by

$$\hat{x}_{mn}(i-1) := E\{x_{mn}|y_n, \hat{\lambda}_m(i-1)\} = y_n \left(\hat{\lambda}_x(m,n) \Big/ \sum_{j=1}^{M_p} \hat{\lambda}_x(j,n) \right)$$

where $\hat{\lambda}_x(m,n) = \hat{\lambda}_m(i-1)p_{mn}$. Next define the likelihood based on $\mathbf{z} = (\mathbf{x}, \mathbf{y})$ assuming λ is known and conditionally independent. Therefore, the complete likelihood is given by

$$\mathcal{L}(\mathbf{x}, \mathbf{y}|\lambda) = \prod_{m,n} e^{-\lambda_x(m,n)} (\lambda_x(m,n))^{x_{mn}} / x_{mn}!$$

and

$$\Lambda(\mathbf{x}, \mathbf{y}|\lambda) = \sum_{m,n} -\lambda_x(m,n) + x_{nm} \ln \lambda_x(m,n) - \ln x_{nm}!$$

or substituting for $\lambda_x(m,n)$ and expanding, we have

$$\Lambda(\mathbf{x}, \mathbf{y}|\lambda) = \sum_{m=1}^{M_p} \sum_{n=1}^{N_d} -\lambda_m p_{nm} + x_{nm} \ln(\lambda_m p_{nm}) - \ln x_{nm}!$$

Differentiating this expression with respect to λ_m, setting to the result zero and solving gives the maximization-step as:

$$\hat{\lambda}_m(i) = \sum_{n=1}^{N_d} \hat{x}_{mn}(i-1)$$

Summarizing the E and M-steps for this problem are given by:

$$\begin{aligned} \text{E-step:} \quad Q(\lambda_m, \hat{\lambda}_m(i-1) &= E_x\{\Lambda(\mathbf{x}, \mathbf{y})|\mathbf{y}, \hat{\lambda}_m(i-1))\} \\ &= y_n \hat{\lambda}_m(i-1) p_{mn} \Big/ \sum_{j=1}^{M_p} \hat{\lambda}_j(i-1) p_{jn} \end{aligned}$$

$$\text{M-step:} \quad \hat{\lambda}_m(i) = \hat{\lambda}_m(i-1) \sum_{n=1}^{N_d} \left(\frac{y_n p_{mn}}{\sum_{j=1}^{M_p} \hat{\lambda}_j(i-1) p_{jn}} \right)$$

More details can be found in [6, 14, 16]. △△△

This completes our discussion of the *EM* approach to parameter estimation problems with incomplete data/parameters. Next we briefly describe one of the most popular approaches to the estimation problem—minimum variance (*MV*) or equivalently minimum mean-squared error (*MMSE*) estimation.

2.4 BATCH MINIMUM VARIANCE ESTIMATION

To complete this discussion evolving from the batch Bayesian perspective, we discuss the development of the minimum (error) variance (*MV*) or equivalently minimum mean-squared error (*MMSE*) estimator. The general techniques developed in this section can be applied to develop various model-based processors [1].

Suppose we are asked to obtain an estimate of a N_x-parameter vector $\mathbf{X}$ 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

$$J(X) = E_X\{[X - \hat{X}(Y)]'[X - \hat{X}(Y)]|Y\} \tag{2.36}$$

where

X is the true random N_x-vector
Y is the measured random N_y-vector (data) and
$\hat{X}$ is the estimate of X given Y

whose minimization leads to the *minimum variance estimator* [1]. Thus, if we minimize $J(X)$ using the chain rule of Eq. 2.11, then we have that

$$\begin{aligned} \nabla_{\hat{X}} J(X) &= E_x\{\nabla_{\hat{X}}(X - \hat{X}(Y))'(X - \hat{X}(Y))|Y\} \\ &= E_x\{-(X - \hat{X}(Y)) - (X - \hat{X}(Y))|Y\} \\ &= -2[E_x\{X - \hat{X}(Y)|Y\}] \end{aligned}$$

performing the conditional expectation operation gives

$$\nabla_{\hat{X}} J(X) = -2[E_x\{X|Y\} - \hat{X}(Y)] \tag{2.37}$$

and setting this equation to zero and solving yields the minimum variance estimate as

$$\hat{X}_{MV} = \hat{X}(Y) = E_x\{X|Y\} \tag{2.38}$$

We check for the global minimum by examining the sign (positive) of the second derivative, and since

$$\nabla_{\hat{X}}(\nabla_{\hat{X}}J(X))' = 2I \tag{2.39}$$

a unique minimum exists. Thus, the minimum variance estimator is the *conditional mean*. The associated error variance is determined by substituting for $\hat{X}(Y)$ into Eq. 2.36 giving

$$\begin{aligned} J_{MV} &= R_{\tilde{X}|Y} = E_x\{(X - E\{X|Y\})'(X - E\{X|Y\})|Y\} \\ &= E_x\{X'X|Y\} - E_x^2\{X|Y\} \end{aligned} \tag{2.40}$$

This estimator is linear, unconditionally and conditionally unbiased and possesses general orthogonality properties. To see this we investigate the estimation error defined by

$$\tilde{X} = X - \hat{X}(Y) \tag{2.41}$$

Taking expectations, we have the fact that the estimator is unconditionally unbiased

$$E_x\{\tilde{X}\} = E_x\{X - \hat{X}(Y)\} = E_x\{X\} - E_x\{E_x\{X|Y\}\} = E_x\{X\} - E_x\{X\} = 0 \tag{2.42}$$

as well as conditionally unbiased, since

$$\begin{aligned} E_x\{\tilde{X}|Y\} &= E_x\{X - \hat{X}(Y)|Y\} = E_x\{X|Y\} - E_x\{E_x\{X|Y\}|Y\} \\ &= E_x\{X|Y\} - E_x\{X|Y\} = 0 \end{aligned} \tag{2.43}$$

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 [4], that is,

$$E_{xY}\{f(Y)\tilde{X}'\} = 0 \tag{2.44}$$

Also,

$$E_x\{f(Y)\tilde{X}'|Y\} = 0 \tag{2.45}$$

This is the well-known orthogonality condition. To see this we substitute for the error

$$\begin{aligned} E_x\{f(Y)\tilde{X}'|Y\} &= E_x\{f(Y)(X - \hat{X}(Y))'|Y\} \\ &= f(Y)E_x\{(X - \hat{X}(Y))'|Y\} \\ &= f(Y)(E_x\{X|Y\} - \hat{X}(Y)') = 0 \end{aligned}$$

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,

$$E_{XY}\{Y\tilde{X}'\} = 0 \tag{2.46}$$

or as before

$$E_X\{Y\tilde{X}'|Y\} = 0 \tag{2.47}$$

This is the well-known minimum variance estimator results in the linear case [1]. For a linear function of the parameter, we have that

$$y = CX + v \tag{2.48}$$

where $y, v, \in R^{N_y \times 1}, X \in R^{N_x \times 1}, C \in R^{N_y \times N_x}$ and v is zero-mean, white with R_{vv}. The mean-squared error criterion

$$J(X) = E\{\tilde{X}'\tilde{X}\} \tag{2.49}$$

is minimized to determine the estimate. The minimization results in the orthogonality condition of Eq. 2.47 which we write as

$$E\{y\tilde{X}'\} = E\{yX'\} - E\{y\hat{X}'_{MV}\} = 0 \tag{2.50}$$

for $\hat{X}_{MV} = K_{MV}y$, a linear function of the data vector. Substituting for y and $\hat{X}$ in this equation gives

$$K_{MV} = R_{XX}C'(CR_{XX}C' + R_{vv})^{-1} \tag{2.51}$$

where R_{XX} is the covariance matrix of X. The corresponding quality is obtained as

$$R_{\tilde{X}\tilde{X}} = (R_{XX}^{-1} + C'R_{vv}^{-1}C)^{-1} \tag{2.52}$$

It is also interesting to note that the fundamental Wiener result [1] is easily obtained from the orthogonality condition of Eq. 2.47, that is,

$$E\{y\tilde{X}'\} = E\{yX'\} - E\{yy'\}K'_{MV} = R_{yX} - R_{yy}K'_{MV} = 0 \tag{2.53}$$

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,

$$K_{MV} = R_{Xy}R_{yy}^{-1} \tag{2.54}$$

Note that *least-squares estimation* is similar to that of minimum variance except that no statistical information (expectation removed) is assumed known about the process, that is, the least-squares estimator, $\hat{y}_{LS}$ minimizes the sum-squared error criterion

$$\min_{\hat{y}} J = \tilde{y}'\tilde{y} = \sum_i \tilde{y}_i^2 \tag{2.55}$$

for $\tilde{y} = y - \hat{y}_{LS}$.

This completes the introduction to *batch* minimum variance, maximum *a posteriori* and maximum likelihood estimation. Next we consider the sequential problem.

2.5 SEQUENTIAL BAYESIAN ESTIMATION

Modern statistical signal processing techniques evolve directly from a Bayesian perspective, that is, they are cast into a probabilistic framework using Bayes' theorem as the fundamental construct. More specifically, the information about the random signal, $x(t)$, required to solve a vast majority of estimation/processing problems is incorporated in the underlying probability distribution generating the process. For instance, the usual signal enhancement problem is concerned with providing the "best" (in some sense) estimate of the signal at time t based on all of the data available at that time. The filtering distribution provides that information directly in terms of its underlying statistics. That is, by calculating the statistics of the process directly from the filtering distribution the enhanced signal can be extracted using a variety of estimators like *MAP*, *ML* or *MMSE* accompanied by a set of performance statistics such as error covariances and bounds. Sequential methods to calculate these distributions become extremely important in pragmatic problems where implementation and speed are an issue. Therefore from an engineering perspective, they are our primary focus.

The roots of this theory are based on Bayesian estimation and in fact sequential Bayesian estimation. We will see that many of our well-known techniques are easily cast into this unifying framework especially in the nonlinear signal processing area. The Bayesian algorithms that provide posterior distribution estimates are optimal; however, they are impossible to implement directly because they require integrations or summations with an infinite number of terms. We will develop the optimal Bayesian algorithms mathematically and perform the required calculations in a sequential manner, but it must be realized that only under certain restricted circumstances can these actually be realized (e.g., the linear Gaussian case). Starting with Bayes' theorem it is possible to show how this leads directly to a recursive or sequential estimation framework that is the foundation of these new approaches.

We cast this discussion into a dynamic variable/parameter structure by defining the "unobserved" signal or equivalently "hidden" variables as the set of N_x-vectors, $\{x(t)\}, t = 0, \ldots, N$. On the other hand, we define the observables or equivalently measurements as the set of N_y-vectors, $\{y(t)\}, t = 0, \ldots, N$ considered to be *conditionally independent* of the signal variables. The goal in recursive Bayesian estimation is to sequentially (in-time) estimate the joint *posterior* distribution, $\Pr(x(0), \ldots, x(N); y(0), \ldots, y(N))$. Once the posterior is estimated than many of the interesting statistics characterizing the process under investigation can be exploited to extract meaningful information.

We start by defining two sets of random (vector) processes: $X_t := \{x(0), \ldots, x(t)\}$ and $Y_t := \{y(0), \ldots, y(t)\}$, as before. Here we can consider X_t to be the set of dynamic

random variables or parameters of interest and Y_t as the set of measurements or observations of the desired process as before.[2] In any case we start with Bayes' theorem for the joint posterior distribution as

$$\Pr(X_t|Y_t) = \Pr(Y_t|X_t) \times \frac{\Pr(X_t)}{\Pr(Y_t)} \tag{2.56}$$

In Bayesian theory (as before), the *posterior* defined by $\Pr(X_t|Y_t)$ is decomposed in terms of the *prior* $\Pr(X_t)$, its *likelihood* $\Pr(Y_t|X_t)$ and the *evidence* or normalizing factor, $\Pr(Y_t)$. Each has a particular significance in this construct which we shall discuss subsequently.

We can replace the evidence by realizing that it is the *total* probability given by

$$\Pr(Y_t) = \int \Pr(Y_t|X_t)\Pr(X_t)\, dX_t \tag{2.57}$$

which is integrated over the entire X_t-dimensional parameter space. Substituting this into Eq. 2.56 we obtain

$$\Pr(X_t|Y_t) = \frac{\Pr(Y_t|X_t) \times \Pr(X_t)}{\int \Pr(Y_t|X_t)\Pr(X_t)\, dX_t} \tag{2.58}$$

Once the posterior distribution is determined, then all statistical inferences or estimates are made by integration or equivalently summation. For example, suppose we would like to obtain a prediction distribution, then it is obtained as

$$\Pr(X_{t+1}|Y_t) = \int \Pr(X_{t+1}|X_t, Y_t) \times \Pr(X_t|Y_t)\, dX_t$$

and a point estimate might be the conditional mean of this distribution, that is,

$$E\{X_{t+1}|Y_t\} = \int X_{t+1}\Pr(X_{t+1}|Y_t)\, dX_{t+1}$$

while the variance of the distribution can be calculated similarly and used for performance evaluation. This calculation illustrates how information that can be estimated from the extracted distribution is applied in the estimation context.

Again, even though simple conceptually, the real problem to be addressed is that of evaluating these integrals which is very difficult because they are only analytically tractable for a small class of priors and likelihood distributions. The large dimensionality of the integrals cause numerical integration techniques to break down which

[2] In *Kalman filtering theory*, the X_t are considered the states or hidden variables not necessarily observable directly, while the Y_t are observed or measured directly.

leads to the approximations we discuss subsequently for stabilization. Let us investigate further to consider posing problems of signal processing interest. First, we review some of the basics.

Many of the derivations and algorithms to follow are based on the simple, but sometimes not so obvious, manipulations of Bayes' theorem; therefore, we develop these variants here first. Starting with a joint distribution of variables indexed by t, we apply Bayes' rule[3] in steps to expose some of the manipulations:

$$\Pr(y(t), y(t-1), y(t-2)) = \Pr(y(t), y(t-1)|y(t-2)) \times \Pr(y(t-2)) \tag{2.59}$$

Applying the rule again to the first term in this expression gives

$$\Pr(y(t), y(t-1)|y(t-2)) = \Pr(y(t)|y(t-1), y(t-2)) \times \Pr(y(t-1)|y(t-2)) \tag{2.60}$$

Combining these results we have

$$\begin{aligned}\Pr(y(t), y(t-1), y(t-2)) = {} & \Pr(y(t)|y(t-1), y(t-2)) \\ & \times \Pr(y(t-1)|y(t-2)) \times \Pr(y(t-2))\end{aligned} \tag{2.61}$$

Additionally, if $\{y(t)\}$ is considered first order Markov, then Eq. 2.61 simplifies even further to

$$\begin{aligned}\Pr(y(t), y(t-1), y(t-2)) = {} & \Pr(y(t)|y(t-1)) \times \Pr(y(t-1)|y(t-2)) \\ & \times \Pr(y(t-2))\end{aligned} \tag{2.62}$$

Generalizing these expansions, we can obtain the *chain rule of probability* which states that the joint distribution Y_t can be decomposed using Bayes' rule as

$$\Pr(Y_t) = \Pr(y(t)|Y_{t-1}) \times \Pr(Y_{t-1}) = \Pr(y(t)|Y_{t-1}) \times \Pr(y(t-1)|Y_{t-2}) \times \Pr(Y_{t-2})$$

or expanding this expression, we obtain

$$\Pr(Y_t) = \prod_{k=0}^{t} \Pr(y(t-k)|Y_{t-k-1}) = \Pr(y(t)|Y_{t-1}) \times \cdots \times \Pr(y(1)|Y_0) \times \Pr(y(0)) \tag{2.63}$$

Further assuming that the process is *Markov*, then

$$\Pr(y(t)|Y_{t-1}) = \Pr(y(t)|y(t-1))$$

[3] In set notation, we have (simply) that

$$\Pr(ABC) = \Pr(AB|C) \times \Pr(C) = [\Pr(A|BC) \times \Pr(B|C)] \times \Pr(C)$$

and the chain rule simplifies even further to

$$\Pr(Y_t) = \prod_{k=0}^{t} \Pr(y(t-k)|Y_{t-k-1}) = \prod_{k=0}^{t} \Pr(y(t-k)|y(t-k-1)) \tag{2.64}$$

Marginal distributions are used throughout these derivations and they follow directly from the *Chapman-Kolmogorov* evolution equation [18], that is,

$$\Pr(y(t)|y(t-2)) = \int \Pr(y(t)|y(t-1)) \times \Pr(y(t-1)|y(t-2))\, dy(t-1) \tag{2.65}$$

Before we close this section, we must mention that the derivations to follow rely on the following two fundamental *assumptions*:

- the dynamic variable $x(t)$ is Markov; and
- the data $y(t)$ are conditionally independent of the *past* dynamic variables, $\{x(t-k)\}\,\forall\, k > 0$.

Next we proceed with the development of generic Bayesian processors.

2.5.1 Joint Posterior Estimation

With this information in mind, we develop the sequential Bayesian solution to the joint posterior estimation problem. Starting with Eq. 2.56, we decompose this expression by first extracting the t^{th} term from the joint distributions, that is,

$$\Pr(Y_t|X_t) = \Pr(y(t), Y_{t-1}|x(t), X_{t-1})$$

and applying Bayes' rule to obtain

$$\Pr(Y_t|X_t) = \Pr(y(t)|Y_{t-1}, x(t), X_{t-1}) \times \Pr(Y_{t-1}|x(t), X_{t-1}) \tag{2.66}$$

The data at time t, $y(t)$, is assumed independent of X_{t-1} and Y_{t-1}; therefore, the first term of Eq. 2.66 simplifies to

$$\Pr(y(t)|x(t), Y_{t-1}, X_{t-1}) \longrightarrow \Pr(y(t)|x(t)) \tag{2.67}$$

The second term also simplifies based on the independence of the past data and $x(t)$ to give

$$\Pr(Y_{t-1}|x(t), X_{t-1}) \longrightarrow \Pr(Y_{t-1}|X_{t-1}) \tag{2.68}$$

We now have the final expression for the *likelihood* as

$$\Pr(Y_t|X_t) = \Pr(y(t)|x(t)) \times \Pr(Y_{t-1}|X_{t-1}) \tag{2.69}$$

Similarly, for the *prior*, extracting the t^{th} term and applying the rule, we obtain

$$\Pr(X_t) = \Pr(x(t), X_{t-1}) = \Pr(x(t)|X_{t-1}) \times \Pr(X_{t-1})$$

Assuming $x(t)$ is a first order Markov process, this expression simplifies to

$$\Pr(X_t) = \Pr(x(t)|x(t-1)) \times \Pr(X_{t-1}) \tag{2.70}$$

Finally the *evidence* is obtained in a similar manner to give

$$\Pr(Y_t) = \Pr(y(t), Y_{t-1}) = \Pr(y(t)|Y_{t-1}) \times \Pr(Y_{t-1}) \tag{2.71}$$

Therefore, substituting these results into Eq. 2.56, we obtain

$$\Pr(X_t|Y_t) = \frac{\overbrace{[\Pr(Y_{t-1}|X_{t-1}) \times \Pr(y(t)|x(t))]}^{\Pr(Y_t|X_t)} \overbrace{[\Pr(x(t)|x(t-1)) \times \Pr(X_{t-1})]}^{\Pr(X_t)}}{\underbrace{\Pr(y(t)|Y_{t-1}) \times \Pr(Y_{t-1})}_{\Pr(Y_t)}} \tag{2.72}$$

but the *posterior* at the previous time, $t-1$, is given by

$$\Pr(X_{t-1}|Y_{t-1}) = \frac{\Pr(Y_{t-1}|X_{t-1}) \times \Pr(X_{t-1})}{\Pr(Y_{t-1})} \tag{2.73}$$

Identifying the terms on the right-hand side of Eq. 2.72 and grouping them together enables the joint *sequential Bayesian posterior estimator* to be expressed as

$$\Pr(X_t|Y_t) = \left[\frac{\Pr(y(t)|x(t)) \times \Pr(x(t)|x(t-1))}{\Pr(y(t)|Y_{t-1})}\right] \times \Pr(X_{t-1}|Y_{t-1}) \tag{2.74}$$

This result is satisfying in the sense that we need only know the *joint* posterior distribution at the previous stage, $t-1$, scaled by a weighting function to sequentially propagate the posterior to the next stage, that is,

$$\overbrace{\Pr(X_t|Y_t)}^{NEW} = \overbrace{\mathcal{W}(t, t-1)}^{WEIGHT} \times \overbrace{\Pr(X_{t-1}|Y_{t-1})}^{OLD} \tag{2.75}$$

where the *weight* is defined by

$$\mathcal{W}(t, t-1) := \left[\frac{\Pr(y(t)|x(t)) \times \Pr(x(t)|x(t-1))}{\Pr(y(t)|Y_{t-1})}\right]$$

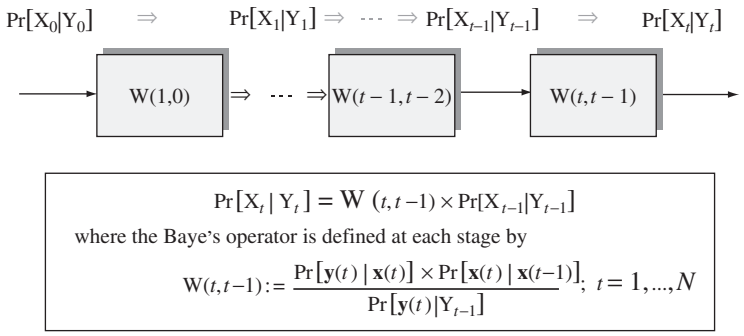


FIGURE 2.2 Sequential Bayesian processor for joint posterior distribution.

The sequential Bayesian processor is shown diagrammatically in Fig. 2.2. Even though this expression provides the full joint posterior solution, it is not physically realizable unless the distributions are known in closed form and the underlying multiple integrals or sums can be analytically determined. In fact, a more useful solution is the *marginal* posterior distribution.

2.5.2 Filtering Posterior Estimation

In this subsection we develop a more realizable Bayesian processor for the posterior distribution of the random $x(t)$. Instead of requiring that the posterior possess the entire set of dynamic variables, X_t, we need only restrict our attention to the current variable at time t. That is, for signal enhancement, we would like to estimate $\Pr(x(t)|Y_t)$ where we restrict our attention to the value of the parameter based on all of the available measurements at time t. We start with the prediction recursion "marginalizing" the joint posterior distribution

$$\Pr(x(t)|Y_{t-1}) = \int \Pr(x(t), x(t-1)|Y_{t-1})\, dx(t-1)$$

Applying Bayes' rule to the integrand yields

$$\Pr(x(t), x(t-1)|Y_{t-1}) = \Pr(x(t)|x(t-1), Y_{t-1}) \times \Pr(x(t-1)|Y_{t-1})$$

or

$$\Pr(x(t), x(t-1)|Y_{t-1}) = \Pr(x(t)|x(t-1)) \times \Pr(x(t-1)|Y_{t-1}) \tag{2.76}$$

where the final expression evolves from the first order Markovian assumption. Applying the *Chapman-Kolmogorov* equation, the *prediction recursion* can be

written as

$$\Pr(x(t)|Y_{t-1}) = \int \Pr(x(t)|x(t-1)) \times \Pr(x(t-1)|Y_{t-1})\, dx(t-1) \tag{2.77}$$

Examining the update or correction equation based on the filtering relation and applying Bayes' rule, we have

$$\Pr(x(t)|Y_t) = \frac{\Pr(x(t), Y_t)}{\Pr(Y_t)} = \frac{\Pr(x(t), y(t), Y_{t-1})}{\Pr(y(t), Y_{t-1})}$$

Again applying the rule to this relation gives

$$\Pr(x(t)|Y_t) = \frac{\Pr(y(t)|x(t), Y_{t-1}) \times \Pr(x(t)|Y_{t-1}) \times \Pr(Y_{t-1})}{\Pr(y(t)|Y_{t-1}) \times \Pr(Y_{t-1})} \tag{2.78}$$

Canceling like terms in numerator and denominator and applying the Markov assumption, we obtain the final expression for the *update recursion*[4] as

$$\overbrace{\Pr(x(t)|Y_t)}^{Posterior} = \frac{\overbrace{\Pr(y(t)|x(t))}^{Likelihood} \times \overbrace{\Pr(x(t)|Y_{t-1})}^{Prior}}{\underbrace{\Pr(y(t)|Y_{t-1})}_{Evidence}} \tag{2.79}$$

where we can consider the update or filtering distribution as a weighting of the prediction distribution as in the full joint case above, that is,

$$\overbrace{\Pr(x(t)|Y_t)}^{UPDATE} = \overbrace{\mathcal{W}_c(t, t-1)}^{WEIGHT} \times \overbrace{\Pr(x(t)|Y_{t-1})}^{PREDICTION} \tag{2.80}$$

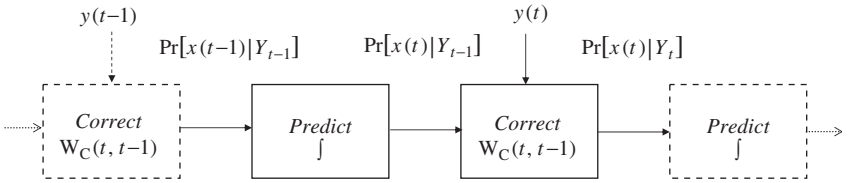
where the *weight* in this case is defined by

$$\mathcal{W}_c(t, t-1) := \frac{\Pr(y(t)|x(t))}{\Pr(y(t)|Y_{t-1})}$$

The resulting processor is shown diagrammatically in Fig. 2.3.

We summarize the sequential Bayesian processor in Table 2.1.

[4] Note that this expression precisely satisfies Bayes' rule as illustrated in the equation.



$$\text{Predict:} \quad \Pr[x(t)|Y_{t-1}] = \int \Pr[x(t)|x(t-1)]\Pr[x(t-1)|Y_{t-1}]dx(t-1)$$

$$\text{Correct:} \quad \Pr[x(t)|Y_t] = \mathrm{W_C}(t,t-1) \times \Pr[x(t)|Y_{t-1}]$$

$$\mathrm{W_C}(t,t-1) := \left[\frac{\Pr[y(t)\,|\,x(t)]}{\Pr[y(t)|Y_{t-1}]}\right]$$

FIGURE 2.3 Sequential Bayesian processor for filtering posterior distribution.

TABLE 2.1 Sequential Bayesian (Filtering) Processor

Prediction

$$\Pr(x(t)|Y_{t-1}) = \int \Pr(x(t)|x(t-1)) \times \Pr(x(t-1)|Y_{t-1})\,dx(t-1) \qquad \text{(prediction)}$$

where $\Pr(x(t)|x(t-1))$ (transition probabliity)

Correction/Update

$$\Pr(x(t)|Y_t) = \Pr(y(t)|x(t)) \times \Pr(x(t)|Y_{t-1})/\Pr(y(t)|Y_{t-1}) \qquad \text{(posterior)}$$

where $\Pr(y(t)|x(t))$ (likelihood)

Initial Conditions

$$\bar{x}(0) \quad \overline{P}(0) \quad \Pr(x(0)|Y_0)$$

2.6 SUMMARY

In this chapter we have developed the idea of statistical signal processing from the Bayesian perspective. We started with the foundations of Bayesian processing by developing the "batch" solutions to the dynamic variable (state) or parameter estimation problem. These discussions led directly to the generic Bayesian processor—the maximum *a posteriori* solution evolving from Bayes' theorem. We showed the relationship of *MAP* to the popular maximum likelihood estimator demonstrating the *ML* is actually a special case when the dynamic variable is assumed deterministic but unknown and therefore not random. We also include the minimum variance estimator (*MV*, *MMSE*) and mentioned the least-squares (*LS*) approach. After discussing some of the basic statistical operations required (chain rule, performance (*CRLB*), etc.), we developed the idea of Bayesian sequential or recursive processing. We started

with the full or joint posterior distribution assuming all of the information about the dynamic variable and observations was available. The solution to this problem led to a sequential Bayesian processor. Lastly we followed with the development of a solution to the more practical marginal posterior or "filtering" distribution and illustrated the similarity to the previous processor.

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 statistical operations and graphics. Least-squares problems are solved with the pseudo-inverse (**pinv**). When the covariance is known (minimum variance) the (**lscov**) command can be applied. Individual linear algebraic techniques including the singular-value decomposition, qr-decomposition (Gram-Schmidt) and eigen-decomposition techniques (**svd**, **qr**, **eig**, etc.) are available. The *Statistics* toolbox offers a wide variety of commands to perform estimation. For instance, "fit" tools are available to perform parameter estimation for a variety of distributions: exponential (**expfit**), Gaussian or normal (**normfit**), Poisson (**poissfit**), etc. as well as the generic maximum likelihood estimation (**mle**) as well as specific likelihood estimator for negative Gaussian/normal (**normlike**), negative exponential (**explike**).

REFERENCES

1. A. Sage and J. Melsa, *Estimation Theory with Applications to Communications and Control* (New York: McGraw-Hill, 1971).
2. H. Van Trees, *Detection, Estimation and Modulation Theory* (New York: Wiley, 1968).
3. J. Candy, "The Cramer-Rao bound: A measure of estimator quality," *Lawrence Livermore National Laboratory Report*, UCID-17660, 1977.
4. I. Rhodes, "A Tutorial Introduction to Estimation and Filtering," *IEEE Trans. Autom. Contr.*, **AC-16**, 1971.
5. J. Candy, *Model-Based Signal Processing* (Hoboken, NJ: John Wiley/IEEE Press, 2006).
6. G. McLachlan and T. Krishnam, *The EM Algorithm and Extensions* (New York: Wiley, 1997).
7. A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. R. Statist. Soc.*, **B**, **39**, 1, 1–38, 1977.
8. R. Duda and P. Hart, *Pattern Classsification* (New York: Wiley, 2000).
9. I. Nabney, *NETLAB: Algorithms for Pattern Recognition* (New York: Springer, 2002).
10. C. Wu, "On the convergence properties of the EM algorithm," *Ann. Statist.*, **11**, 1, 95–103, 1983.

11. R. Redner and H. Walker, "Mixture densities, maximum likelihood and the EM algorithm," *SIAM Rev.*, **26**, 2, 195–239, 1984.
12. W. Gilks, S. Richardson and D. Spiegelhalter, *Markov Chain Monte Carlo in Practice* (New York: Chapman & Hall, 1996).
13. M. Tanner, *Tools for Statistical Inference*, *2nd Ed.* (New York: Springer-Verlag, 1993).
14. T. Moon, "The expectation-maximization algorithm," *IEEE Sig. Proc. Mag.*, **13**, 6, 47–60, 1996.
15. J. Bilmes, "A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models," *U. C. Berkeley Report*, TR-97-021, 1998.
16. D. Snyder and M. Miller, *Random Point Processes in Time and Space*, *2nd Ed.* (New York: Springer, 1991).
17. A. Gelman, J. Carlin, H. Stern and D. Rubin, *Bayesian Data Analysis*, *2nd Ed.* (New York: Chapman & Hall, 2004).
18. A. Papoulis and S. Pillai, *Probability, Random Variables, and Stochastic Processes*, *4th Ed.* (New York: McGraw-Hill, 2002).

PROBLEMS

2.1 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*) $E_{xy}\{cX + dY|Z\} = cE\{X|Z\} + dE\{Y|Z\}$.

2.2 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.

2.3 Show that for any unbiased estimate of $\hat{\mathbf{x}}(y)$ of the parameter vector $\mathbf{x}$ the *Cramer Rao bound* is

$$\text{Cov}(\tilde{x}|x) \geq \mathcal{I}^{-1} \quad \text{for } \tilde{x} = x = \hat{x}(y) \text{ and } \mathbf{x} \in \mathcal{R}^n, \mathcal{I} \in \mathcal{R}^{n\times n}$$

where the *information matrix* is defined by

$$\mathcal{I} := -E_y\{\nabla_x(\nabla_x \ln Pr(Y|X))'\}$$

2.4 The sample mean, $\overline{\theta} = 1/N \sum_{t=1}^{N} \theta(t)$, is a very important statistic in data processing.

(*a*) Show that the sample mean is an *unbiased* estimator of $\theta \sim \mathcal{E}xp(1/\theta)$.

(*b*) Estimate the corresponding variance of $\overline{\theta}$.

(*c*) Show that the sample mean is a *consistent* estimator of θ.

(*d*) Construct the two standard deviation *confidence interval* about the sample mean estimator. (*Hint*: Let $\overline{\theta} \sim N(\theta, \sigma^2)$, σ^2 known.)

(*e*) Show that $\overline{\theta}$ is a *minimum variance* estimator. (*Hint*: Show it satisfies the *Cramer Rao bound* with equality).

(*f*) If $\hat{\theta}$ is an estimator or θ, calculate the corresponding *mean-squared error*.

2.5 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 $y, v \in \mathcal{R}^p$ and x is governed by the state transition mechanism

$$x(t+1) = \Phi(t+1, t)\,x(t) \quad \text{for } \Phi \in \mathcal{R}^{n \times n}$$

Calculate the Cramer Rao bound for the case of estimating the initial state $x(0)$.

2.6 Suppose we have the following signal in additive Gaussian noise:

$$y = x + n \quad \text{with } n \sim N(0, R_{nn})$$

(*a*) Find the Cramer-Rao bound if x is assumed unknown.

(*b*) Find the Cramer-Rao bound if $p(x) = xe^{-x^2/R_{nn}},\ x \geq 0$.

2.7 Suppose 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 $\operatorname{cov} \tilde{x}$, $\tilde{x} = x - \hat{x}$.

2.8 We would like to estimate a vector of unknown parameters $p \epsilon R^n$ from a sequence of N-noisy measurements given by

$$y = Cp + n$$

where $C \epsilon R^{p \times n}$, $y, n \epsilon R^p$, and v is zero-mean and white covariance R_{nn}.

(*a*) Find the minimum variance estimate $\hat{p}_{MV}$.

(*b*) Find the corresponding quality $\operatorname{cov}(p - \hat{p}_{MV})$.

2.9 Suppose we have N samples $\{x(0) \ldots x(N-1)\}$ of a process $x(t)$ to be estimated by N complex sinusoids of arbitrary frequencies $\{f_0, \ldots, f_{n-1}\}$. Then

$$x(k\,\triangle T) = \sum_{m=0}^{N-1} a_m \exp(j2\pi f_m k\,\triangle T) \quad \text{for } k = 0, \ldots, N-1$$

Find the least-squares estimate $\hat{a}_{LS}$ of $\{a_m\}$.

2.10 Suppose 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

$$J = E\{(s - \hat{s})^2\}$$

2.11 Find the maximum likelihood and maximum a posteriori estimates of the parameter x

$$p(x) = \alpha e^{-\alpha x} \quad x \geq 0, \;\; \alpha \geq 0$$

and

$$p(y|x) = xe^{-xy} \quad x \geq 0, \;\; y \geq 0$$

2.12 Suppose we have two classes of objects, black and white with shape subclasses, circle and square and we define the random variables as:

$$\begin{aligned} X_1 &= \text{number of black circular objects} \\ X_2 &= \text{number of black square objects} \\ X_3 &= \text{number of white objects} \end{aligned}$$

Assume that these objects are trinomially distributed such that:

$$\Pr(\mathbf{x}|\Theta) = \left(\frac{(x_1 + x_2 + x_3)!}{x_1! x_2! x_3!}\right)\left(\frac{1}{4}\right)^{x_1}\left(\frac{1}{4} + \frac{\Theta}{4}\right)^{x_2}\left(\frac{1}{2} - \frac{\Theta}{4}\right)^{x_3}$$

Suppose a person with blurred vision cannot distinguish shape (circle or square), but can distinguish between black and white objects. In a given batch of objects, the number of objects detected by this person is: $\mathbf{y} = [y_1\ y_2]'$ with

$$\begin{aligned} y_1 &= x_1 + x_2 \\ y_2 &= x_3 \end{aligned}$$

(*a*) What is the probability, $\Pr(y_1|\Theta)$?

(*b*) What is the expression for the E-step of the *EM*-algorithm assuming $\hat{\Theta}_k$ is the current parameter estimate, that is, find the expression for $E\{x_1|y_1, \hat{\Theta}_k\}$ with x_3 known?

(*c*) What is the corresponding M-step?

(*d*) Take the *EM* solution (a)–(c) above based on 100 samples with $y_1 = 100$ and start iterating with $\Theta_0 = 0$ for 10-steps, what is your final estimate of the parameter, $\hat{\Theta}_{10}$? ($\Theta_{true} = 0.5$) for simulation $x_1 = 25, x_2 = 38$.

2.13 Suppose we have a bimodal distribution consisting of a Gaussian mixture with respective means, variances and mixing coefficients: $\{\mu_1, \sigma_1^2, p_1\}$ and $\{\mu_2, \sigma_2^2, p_2\}$ such that $\mathrm{p}_X(x) = \sum_{i=1}^{2} p_i \mathcal{N}(\mu_i, \sigma_i^2)$ with $p_2 = 1 - p_1$. We would like to fit the parameters, $\Theta = (p_1, \mu_1, \sigma_1^2, \mu_2, \sigma_2^2)$ to data. Develop the *EM* algorithm for this problem.

2.14 Suppose we are given a measurement system

$$y(t) = x(t) + v(t) \quad t = 1, \ldots, N$$

where $v(t) \sim N(0, 1)$.

(*a*) Find the maximum likelihood estimate of $x(t)$, that is, $\hat{x}_{ML}(t)$, for $t = 1, \ldots, N$.

(*b*) Find the maximum *a posteriori* estimate of $x(t)$ that is, $\hat{x}_{MAP}(t)$, if $p(x) = e^{-x}$.

2.15 Suppose we have a simple AM receiver with signal

$$y(t) = \Theta s(t) + v(t) \quad t = 1, \ldots, N$$

where Θ is a random amplitude, s is the known carrier, and $v \sim N(0, R_{vv})$.

(*a*) Find the maximum likelihood estimate $\hat{\Theta}_{ML}$.

(*b*) Assume $\Theta \sim N(\Theta_0, R_{\Theta\Theta})$. Find the maximum *a posteriori* estimate $\hat{\Theta}_{MAP}$.

(*c*) Assume Θ is Rayleigh-distributed (a common assumption). Find $\hat{\Theta}_{MAP}$.

2.16 We would like to estimate a signal from a noisy measurement

$$y = s + v$$

where $v \sim N(0, 3)$ and s is Rayleigh-distributed

$$p(s) = se^{-s^2/2}$$

with

$$p(y|s) = \frac{1}{\sqrt{6\pi}} e^{-\frac{(y-s)^2}{6}}$$

(*a*) Find the maximum likelihood estimate.

(*b*) Find the maximum *a posteriori* estimate.

(*c*) Calculate the Cramer-Rao bound (ignoring $p(s)$).

2.17 Assume that a planet travels an elliptical orbit centered about a known point. Suppose 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 $\hat{\beta} = [\hat{\beta}_1 \ \hat{\beta}_2]'$.

(*b*) Suppose we are given the following measurements: $(x, y) = \{(2, 2), (0, 2), (-1, 1), \text{and } (-1, 2)\}$, find $\hat{\beta}$ and $\hat{J}$, the cost function.

2.18 Find the parameters, β_1, β_2 and β_3 such that

$$f(t) = \beta_1 t + \beta_2 t^2 + \beta_3 \sin t$$

fits $(t, f(t)) := \{(0, 1), (\pi/4, 2), (\pi/2, 3), \text{and } (\pi, 4)\}$ with corresponding quality estimate, $\hat{J}$.

3

SIMULATION-BASED BAYESIAN METHODS

3.1 INTRODUCTION

In this chapter we investigate the idea of Bayesian estimation [1–13] using approximate sampling methods to obtain the desired solutions. We first motivate the simulation-based Bayesian processors and then review much of the basics required for comprehension of this powerful methodology. Next we develop the idea of simulation-based solutions using the *Monte Carlo* (*MC*) approach [14–21] and introduce *importance sampling* as a mechanism to implement this methodology from a generic perspective [22–28]. Finally, we consider the class of iterative processors founded on Markov chain concepts leading to efficient techniques such as the foundational Metropolis-Hastings approach and the Gibbs sampler [29–37].

Starting from Bayes' rule and making assertions about the underlying probability distributions enables us to develop reasonable approaches to design approximate Bayesian processors. Given "explicit" distributions, it is possible to develop analytic expressions for the desired posterior distribution. Once the posterior is estimated, then the Bayesian approach allows us to make inferences based on this distribution and its associated statistics (e.g., mode, mean, median, etc.). For instance, in the case of a *linear* Gauss-Markov (*GM*) model, calculation of the posterior distribution leads to the optimal minimum variance solution [11]. But again this was based completely on the assertion that the dynamic processes were strictly constrained to Gaussian distributions and a *linear GM* model therefore leading to a Gaussian posterior. When the dynamics become nonlinear, then approximate methods evolve based on "linearization" techniques either model-based (Taylor series transformations) or statistical (sigma-point transformations). In both cases, these are fundamentally approximations that are constrained to unimodal distributions.

Bayesian Signal Processing. By James V. Candy

What happens when both the dynamics and statistics are nonstationary and non-Gaussian? Clearly, these approaches can be applied, but with little hope of success under most conditions. Therefore, we must resort to other less conventional (in signal processing) ways of attacking this class of problems that have dominated the science and engineering literature for a long time [38–61]. This question then leads us directly to statistical simulation-based techniques invoking random sampling theory and the *Monte Carlo* (*MC*) method—well-known in statistics for a long time [14]. This method is essentially a collection of techniques to estimate statistics based on random sampling and simulation. It can be thought of simply as "performing estimation through sampling". The *goal* of Bayesian techniques using *MC* methods is to generate a set of independent samples from the target posterior distribution with enough samples to perform accurate inferences [21]. Monte Carlo techniques have been applied to a large variety of applications in science and engineering [56, 57, 61]. In this chapter, we start out with the simple idea of random sampling with the underlying motive of developing Monte Carlo simulation techniques to solve nonlinear/non-Gaussian signal processing problems.

MC methods involve techniques to estimate the posterior distribution of interest using either numerical integration-based methods (when possible) or sample-based simulation methods which attempt to produce *independent-identically-distributed* (*i.i.d.*) samples from a targeted posterior distribution and use them to make statistical inferences. Following Smith [18], we develop this idea further starting out with Bayes' rule for the variable or parameter, X, and the corresponding data, Y, that is, the posterior distribution is given by

$$\Pr(X|Y) = \frac{\Pr(Y|X) \times \Pr(X)}{\Pr(Y)} = \frac{\Pr(Y|X) \times \Pr(X)}{\int \Pr(Y|X) \times \Pr(X)\, dX} \tag{3.1}$$

with the usual definitions of likelihood, prior and evidence (normalization) distributions. Once the posterior is estimated, then the inferences follow immediately. For instance, the conditional mean is given by

$$E\{X|Y\} = \int X \Pr(X|Y)\, dX \tag{3.2}$$

In the continuous (random variable) case the explicit evaluation of these integrals is required, yet rarely possible, leading to sophisticated numerical or analytical approximation techniques to arrive at a solution. These methods rapidly become intractable or computationally intensive to be of any real value in practice. Thus an alternative method is necessary.

Just as we can use the distribution to generate a set of random samples, the samples can be used to generate (approximately) the underlying distribution (e.g., histogram). The idea of using samples generated from the prior through the likelihood to obtain the posterior parallels that of using analytic distributions. For example, ignoring the evidence we use the *method of composition* [16] to sample:

$$\Pr(X|Y) \propto \Pr(Y|X) \times \Pr(X) \tag{3.3}$$

so that we can

- Generate samples from the prior: $X_i \rightarrow \Pr(X);\ i = 1, \ldots, N$;
- Calculate the likelihood: $\Pr(Y|X_i)$; and
- Estimate the posterior: $\hat{\Pr}(X|Y) \approx \Pr(Y|X_i) \times \Pr(X_i)$.

Unfortunately, this methodology is not realistic unless we can answer the following questions: (1) How do we generate samples from distributions for simulations? and (2) How do we guarantee that the samples generated are *i.i.d.* from the targeted posterior distribution?

However, before we develop the idea of sampling and simulation-based methods, we must have some mechanism to evaluate the performance of these samplers. In the next section, we briefly describe techniques to estimate the underlying probability distributions from data samples.

3.2 PROBABILITY DENSITY FUNCTION ESTIMATION

One of the requirements of Bayesian signal processing is to generate samples from an estimated posterior distribution. If the posterior is of a known closed form (e.g., Gaussian), then it is uniquely characterized by its particular parameters (e.g mean and variance) that can be "fit" to the samples directly using parameter estimation techniques. But, if the probability density[1] is too complex or cannot readily be represented in closed form, then *nonparametric* techniques must be employed to perform the estimation.

Since most parametric forms rarely fit the underlying posterior distribution encountered in the real world, especially if they are multimodal (multiple peaks) distributions, we investigate the so-called *kernel* (smoothing) method of *PDF* estimation. The basic idea is to fit a simple model individually at each target sample (random variate) location, say $\hat{x}$, using the observations close to the target, $\mathrm{p}(\hat{x})$. This estimate can be accomplished by using a *weighting function* or equivalently, *kernel* function, $\mathcal{K}(\hat{x}; x_i)$ that assigns a weight to x_i based on its proximity to $\hat{x}$ [4]. Different kernels lead to different estimators. For instance, the classical *histogram* can be considered a rectangular or box kernel *PDF* estimator [5], while the fundamental Parzen window multidimensional estimator is based on a hypercube kernel [3]. Some of the popular kernel smoothing functions include the triangle, Gaussian, Epanechnikov and others [4]. In any case, kernel density estimation techniques [6] provide the basic methodology required to estimate the underlying *PDF* from random data samples and subsequently evaluate sampler performance—one of the objectives of this chapter.

Theoretically, the underlying principle of distribution estimation is constrained by the properties of the *PDF* itself, that is,

- $\mathrm{p}_X(x) \geq 0\ \forall x \in \mathcal{R}_X$; and
- $\int_{\mathcal{R}_x} \mathrm{p}_X(x)\, dx = 1$.

[1] In this section we introduce more conventional notation for both the cumulative and density or mass functions, since it is required. The *CDF* of the random variable X with realization x is defined by $\mathrm{P}_X(x)$, while the *PDF* or *PMF* is $\mathrm{p}_X(x)$. In instances where it is obvious, we do not use the subscript X to identify the random variable.

Formally, the probability that the random variate, x, will fall within the interval, $x_i \leq X \leq x_j$ for $i < j$ is given by [3]

$$\Pr(x_i < X < x_j) = \int_{x_i}^{x_j} \mathrm{p}_X(x)\, dx \approx \mathrm{p}_X(x) \times \Delta_x \quad \text{for } \Delta_x \ll 1 \text{ (small)} \tag{3.4}$$

which is simply the *area* under the *PDF* in the interval (x_i, x_j). Thus the probability that X is in a small interval of length Δ_x is proportional to its *PDF*, $\mathrm{p}_X(x)$.

Therefore, it follows that the probability at a point, say $\hat{x}$, can be approximated by allowing $x_i = x$ and $x_j = x + \Delta_x$, then

$$\Pr(x < \hat{x} < x + \Delta_x) = \int_{x}^{x+\Delta_x} \mathrm{p}_X(x) dx \quad \text{for } x \in \mathcal{R}_x \tag{3.5}$$

demonstrating that

$$\mathrm{p}_X(x) = \lim_{\Delta_x \to 0} \frac{\Pr(x < \hat{x} < x + \Delta_x)}{\Delta_x} \tag{3.6}$$

Consider a random draw of *N-i.i.d.* (total) samples according to $\mathrm{p}_X(x)$ above. The probability that n_N of these reside within $\mathcal{R}_x$ is binomial and can be approximated by the *frequency ratio*, n_N/N, leading to a relative frequency interpretation of the probability. That is, it follows that

$$\mathrm{p}_X(x) \times \Delta_x = \frac{n_N}{N} \tag{3.7}$$

But this is just the well-known [1] *histogram* estimator given by

$$\hat{\mathrm{p}}_X(x) := \hat{\mathrm{p}}_X(\hat{x}) = \frac{1}{\Delta_x} \times \frac{n_N}{N} \quad |x - \hat{x}| \leq \frac{\Delta_x}{2} \tag{3.8}$$

Here $\hat{x}$ is located at the midpoint of the interval (bin), $\hat{x} - \frac{\Delta_x}{2} \leq x \leq \hat{x} + \frac{\Delta_x}{2}$ and the corresponding probability is assumed *constant* throughout the bin as shown in Fig. 3.1. If

$$\mathrm{p}_N(x) \approx \frac{n_N/N}{\Delta_x} \tag{3.9}$$

then as $N \to \infty$, histogram estimator converges to the true *PDF* that is, $\mathrm{p}_N(x) \to \mathrm{p}_X(x)$ and the following three conditions must hold:

1. $\lim_{N\to\infty} \Delta_x \to 0$;
2. $\lim_{N\to\infty} n_N \to \infty$; and
3. $\lim_{N\to\infty} \frac{n_N}{N} \to 0$.

The first condition must hold to achieve the limit in Eq. 3.6, while the second condition indicates how n_N is to grow to guarantee convergence. Thus, at all nonzero points of

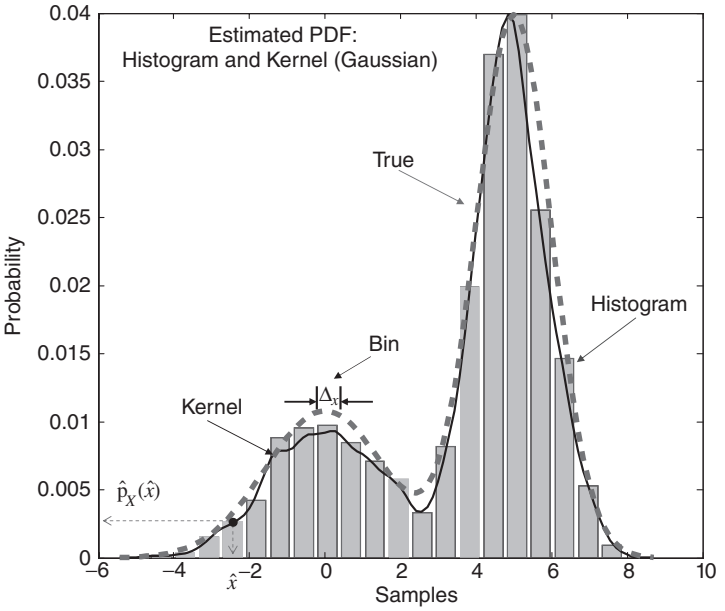


FIGURE 3.1 Estimated *PDF* using both histogram and kernel (Gaussian window, $\mathcal{N}(0,10)$) methods for random samples generated from a Gaussian mixture: $\mathcal{N}_1(0, 2.5)$ and $\mathcal{N}_2(5,1)$ and mixing coefficients, $P_1 = 0.3$ and $P_2 = 0.7$.

the *PDF* by fixing the size of the interval, the probability of samples falling within this interval is finite and therefore, $n_N \approx \Pr(x) \times N$ and $n_N \to \infty$ as $N \to \infty$. The final condition arises arises because as the interval size shrinks, $\Delta_x \to 0$, the corresponding probability $\Pr(x) \to 0$. These conditions indicate the manner in which the parameters should be chosen: *N large*, Δ_x *small* with n_N *large*. Satisfaction of all of these conditions imply convergence to $\mathrm{p}_X(x)$ (see [3, 5] for more details).

There are two common ways of ensuring convergence: (1) "shrink" the region by selecting the interval as a function of the number of samples such as $\Delta_x = \frac{1}{\sqrt{N}}$ (Parzen-window method)[2]; and (2) "increase" the number of samples within the interval which can be done as $n_N = \sqrt{N}$. Here the local region grows until it encloses n_N-neighbors of x (nearest-neighbor method). Both methods converge in probability. This is the principle theoretical result in *PDF* estimation [3, 5, 6]. Next consider the kernel density estimator as a "generalized" histogram.

Let $x_1, \ldots, x_N$ be a set of *i.i.d.* samples of the random variate, x, then the *kernel density* estimator approximation of the underlying probability density *at* x is given by

$$\hat{\mathrm{p}}_X(x) = \frac{1}{\Delta_x}\left(\frac{1}{N}\sum_{i=1}^{N}\mathcal{K}\left(\frac{x - x_i}{\Delta_x}\right)\right) = \frac{1}{N\,\Delta_x}\sum_{i=1}^{N}\mathcal{K}\left(\frac{x - x_i}{\Delta_x}\right) \tag{3.10}$$

[2] It has been shown that the "optimal" bin (interval) size can be obtained by searching for the bin that minimizes the cost function, $C(\Delta_x) = \frac{(2\bar{x} - \sigma_x^2)}{\Delta_x^2}$ [7].

where $\mathcal{K}$ is defined as the kernel density (non-negative and integrates to unity) with Δ_x the corresponding smoothing parameter or *bandwidth* used to "tune" the estimate. The classic bias-variance tradeoff arises here. That is, for smaller Δ_x (high resolution) yields smaller bias, but higher variance. However, a larger Δ_x (low resolution), gives a larger bias with smaller variance [5]. We demonstrate the application of kernel techniques in the following example of estimating a multimodal distribution.

Example 3.1

Suppose we would like to estimate the *PDF* of a set of 1000-samples generated from a mixture of two Gaussian densities such that $\mathcal{N}(\mu_x, \sigma_x^2) \rightarrow \mathcal{N}_1(0, 2.5)$ and $\mathcal{N}_2(5, 1)$ with mixing coefficients, $P_1 = 0.3$ and $P_2 = 0.7$. We use both the histogram and kernel density estimators. For the histogram we choose $N = 20$ bins to illustrate (graphically) the bin estimates, while for the kernel density estimator we choose a Gaussian window specified by $K(x) = \frac{1}{\sqrt{20\pi}}\, e^{\frac{-x^2}{20}}$ with an optimal bandwidth of $\Delta_x = \sigma_x \times \left(\frac{4}{3N}\right)^{\frac{1}{5}}$ [3].

The results are shown in Fig. 3.1 for 100-bins or points where we see the estimated *PDF* using both methods along with the "true" Gaussian density. Clearly the Gaussian kernel estimator provides a much "smoother" estimate of the true density. A smoother histogram estimate can be obtained by decreasing the bin-width. Both techniques capture the general shape of the bimodal distribution. △△△

This completes the section on *PDF* estimation techniques. Note such details as selecting the appropriate bin-size for the histogram and the bandwidth for the kernel density estimator as well as its structure (Gaussian, Box, etc.) is discussed in detail in [3, 5–7]. These relations enable "tuning" of the estimators for improved performance.

Next, we consider the simulation-based methods in and effort to answer the questions posed previously: (1) How do we generate samples from distributions for simulations? and (2) How do we guarantee that the samples generated are *i.i.d.* from the targeted posterior distribution?

3.3 SAMPLING THEORY

The generation of random samples from a known distribution is essential for simulation. If the distribution is standard and has a closed analytic form (e.g., Gaussian, exponential, Rayleigh, etc.), then it is usually possible to perform this simulation easily. This method is called the *direct method* because it evolves "directly" from the analytic form. An alternative is the "inversion" method which is based on uniform variates and is usually preferred. Thus, the uniform distribution becomes an extremely important distribution in sampling because of its inherent simplicity and the ease in which samples can be generated [15].

Many numerical sample generators evolve by first generating a set of random numbers that are uniformly distributed on the interval $[a, b]$ such that all of the subintervals are of equal length and have "equal probability". Thus, X is said to be a *uniform random variable*

$$X \sim \mathcal{U}(a, b) \tag{3.11}$$

that has a *cumulative distribution function* (*CDF*) defined by

$$\mathrm{P}_X(x) = \Pr(X \le x) = \begin{cases} 0 & x < a \\ \frac{x-a}{b-a} & a \le x < b \\ 1 & x \ge b \end{cases} \tag{3.12}$$

along with the corresponding probability density

$$\mathrm{p}_X(x) = \frac{1}{b-a} \quad a \le x \le b \tag{3.13}$$

with mean and variance: $\mu_x = \frac{b+a}{2}$ and $\sigma_x^2 = \frac{(b-a)^2}{12}$.

Because of its simplicity and ease of numerical generation, the uniform variate is the starting point of many simulation schemes. For instance, to generate a binomial random number (e.g., coin tossing with Pr(heads) = p), we first generate uniform random variates and then count the number of times they are greater than p. The result is a set of binomial variates with trial parameter, N, and success (heads) probability, p.

Since statistical sampling techniques are based on the generation of random samples from a given distribution, we must understand the relationship between an *input PDF*, $\mathrm{p}_X(x)$ and an *output PDF*, $\mathrm{p}_Y(y)$ when there exists a relationship between the random variables x and y defined by: $y = y(x)$. The problem is:

GIVEN, x and $\mathrm{p}_X(x)$, **FIND** the output *PDF*, $\mathrm{p}_Y(y)$ defining the probability of y.

When we have functions of random variables with known analytic distributions, then the usual *transformation* method applies. That is, given the known distribution, $\mathrm{P}_X(x)$ or density $\mathrm{p}_X(x)$ and the monotonic, one-to-one, invertible transformation, $y = T(x)$, then the distribution or density of y is found by the using the transformation relation [1],

$$\mathrm{p}_Y(y) = \mathrm{p}_X(x) \times \left|\frac{\partial y}{\partial x}\right|^{-1} = \mathrm{p}_X(x = T^{-1}(y)) \times \left|\frac{\partial x}{\partial y}\right| \tag{3.14}$$

where $\left|\frac{\partial y}{\partial x}\right|$ is the Jacobian of the transformation and T^{-1} is its inverse. The derivation of this relationship follows directly from the definitions of *CDF* [2]. Although simple, this relation establishes one of the basic concepts in random sampling.

Perhaps the most fundamental relation is obtained by applying the transformation directly to the cumulative distribution function (*CDF*) defined by

$$y(x) = \mathrm{P}_X(x) = \int_{-\infty}^{\infty} \mathrm{p}_X(\alpha)\, d\alpha \tag{3.15}$$

That is, applying the transformation of 3.14 and taking the required derivatives gives

$$\frac{dy}{dx} = \frac{d\mathrm{P}_X(x)}{dx} = \mathrm{p}_X(x) \tag{3.16}$$

which follows from the fundamental theorem of calculus. Therefore we have

$$\mathrm{p}_Y(y) = y(x) = \frac{\mathrm{p}_X(x)}{\left|\mathrm{p}_X(x)\right|} = 1 \tag{3.17}$$

yielding a uniformly distributed random variable, $y \sim \mathcal{U}(0,1)$, that is,

$$\mathrm{P}_Y(y) = y(x) = 1 \quad \text{for } 0 \leq y \leq 1 \tag{3.18}$$

Thus, the *CDF* of any random variable is *always* uniformly distributed on the interval, [0, 1], independently of $\mathrm{p}_X(x)$! This important result enables us to "sample" from any arbitrary $\mathrm{p}_X(x)$, since the random variable, x can be sampled by first generating samples from $y \sim \mathcal{U}(0,1)$ and then performing the inversion $x = \mathrm{P}_X^{-1}(y)$. This is called the *inversion method* in sampling theory that simply entails generating random samples from $\mathcal{U}(0,1)$ and determining the samples x by inversion of its (known) *CDF*. Note that the *CDF* can be known analytically or just in tabular form to perform the inversion. We discuss this theorem more formally in the next section. First, consider the following example to illustrate this idea.

Example 3.2

Suppose we have a uniform random variable, $x \sim \mathcal{U}(0,1)$ and a transformation, $y = T(x) = -\frac{1}{\lambda} \ln x$. We would like to know the analytic form of the density, $\mathrm{p}_Y(y)$. Since x is uniform, the density is $\mathrm{p}_X(x) = \frac{1}{b-a} = 1$, the inverse transformation is, $x = T^{-1}(y) = e^{-\lambda y}$; therefore, taking the derivative and substituting its absolute value into Eq. 3.14 gives

$$\mathrm{p}_Y(y) = (1) \times \left|-\lambda\, e^{-\lambda y}\right| = \lambda\, e^{-\lambda y}$$

an exponential distribution with parameter, λ. △△△

Transformations of *discrete* random variables are much simpler where we use the probability mass and discrete cumulative distribution functions in place of their continuous counterparts. In this case the transformation is still continuous with the identical conditions (invertible, etc.), that is,

$$y_i = T(x_i) \quad i = 1, \ldots, N \tag{3.19}$$

and therefore, we have the discrete *CDF* as

$$\mathrm{P}_Y(y) = \Pr(Y \leq y_i) = \Pr(X \leq x_i) = \Pr(X \leq x_i = T^{-1}(y_i)); \quad i = 1, \ldots, N \tag{3.20}$$

3.3.1 Uniform Sampling Method

As noted, the uniform distribution plays a vital role in simulating random variates and is applied heavily in sample-based simulation schemes. We formally present two

fundamental theorems and a corollary [1] to justify the theoretical foundation of this approach and motivate their use through some simple examples.

Uniform Transformation Theorem Given a random variable, X, with distribution, $\mathrm{P}_X(x)$, there *exists* a unique, monotonic, transformation, $T(X)$ such that the random variable, $U = T(X)$ is distributed as $\mathcal{U}(0, 1)$, that is, if $T(X)$ is selected as

$$T(X) = U$$

then

$$U \sim \mathcal{U}(0, 1)$$

The proof of this theorem follows directly [1] from the properties of the *CDF*. Suppose x is arbitrary and $u = \mathrm{P}_X(x)$. Then with a monotonic $\mathrm{P}_X(x)$, we have $U \leq u$ iff $X \leq x$ and therefore it follows that

$$\mathrm{P}_U(u) = \Pr(U \leq u) = \Pr(X \leq x) = \mathrm{P}_X(x) = u$$

giving the desired result. This is a strong result stating that *any CDF* can be represented in terms of a set of uniform random variates as discussed before.

The second theorem [1] provides the basis of most sample-based simulations and is reliant on the existence of the "inverse" of the *CDF*.

Inverse Transformation Theorem Given a uniform random variable, $U \sim \mathcal{U}(0, 1)$ and a distribution (target), $\mathrm{P}_X(x)$, then there *exists* a unique, monotonic, transformation, $T(U)$, such that the random variable, $X = T(U)$ is distributed as $\mathrm{P}_X(x)$, that is, if $T(U)$ is selected as

$$T(U) = X = \mathrm{P}_X^{-1}(U)$$

then

$$X \sim \mathrm{P}_X(x)$$

Again the proof of this theorem follows directly from the properties of the *CDF* [1]. From the Uniform Transformation Theorem above, we know that U is uniform and $\mathrm{P}_X(X)$ is arbitrary; therefore, $U = \mathrm{P}_X(X) \longrightarrow X = \mathrm{P}_X^{-1}(U)$ and it follows that

$$\mathrm{P}_X(X \leq x) = \mathrm{P}_X(X \leq \mathrm{P}_X^{-1}(U)) = \mathrm{P}_X(\mathrm{P}_X^{-1}(U)) = u = \mathrm{P}_X(X)$$

providing the proof. We illustrate this technique with the following example.

Example 3.3

Suppose we would like to generate a random variable, X, exponentially distributed with parameter λ, $X \sim \mathrm{Exp}(\lambda)$. From the Uniform Transformation Theorem, we have

$$u = \mathrm{P}_X(x) = 1 - e^{-\lambda x}$$

and

$$x = \mathrm{P}_X^{-1}(U) = -\frac{1}{\lambda}\ln(1-u)$$

To generate exponential random variables:

- Generate uniform samples: $u_i \longrightarrow \mathcal{U}(0,1)$
- Transform to exponential: $x_i = -\frac{1}{\lambda}\ln(1-u_i)$

Also since $\ln(1-u_i) \sim \mathcal{U}(0,1)$, then the exponential variates are generated more efficiently from $x_i = -\frac{1}{\lambda}\ln(u_i)$. △△△

To further illustrate the *inverse CDF method*, we apply it to a discrete problem that occurs quite frequently in nonlinear processing.

Example 3.4

Suppose we have estimated a discrete empirical distribution representing a posterior density given by

$$\hat{\mathrm{p}}_X(x) = \sum_{i=1}^{N} W_i \delta(x - x_i)$$

with corresponding *CDF*

$$\hat{\mathrm{P}}_X(x) = \sum_{i=1}^{N} W_i \mu(x - x_i)$$

where $\mu(x)$ is a unit-step function, $W_i := \Pr(X = x_i)$ and of course, $\sum_{i=1}^{N} W_i = 1$. The *CDF* is shown in Fig. 3.2a. Using the inverse *CDF* method, we can generate realizations of $X = x_i$ by:

- Generate uniform samples: $u_k \sim \mathcal{U}(0,1)$
- Simulate samples: $x_i = \mathrm{P}_X^{-1}(u)$ or

$$x_i = x_k \quad \text{for } \mathrm{P}_X(x_{k-1}) \le u_k < \mathrm{P}_X(x_k)$$

or using the empirical distribution

$$x_i = x_k \quad \text{for } \sum_{j=1}^{k-1} W_j \le u_k < \sum_{j=1}^{k} W_j$$

This transformation is illustrated in Fig. 3.2b. △△△

Summarizing, we first generate a uniform random variate, u_i and then "bracket" its probability to determine the desired random variable, x_i. So we see from these examples that using the inverse *CDF* method enables us to generate continuous and discrete

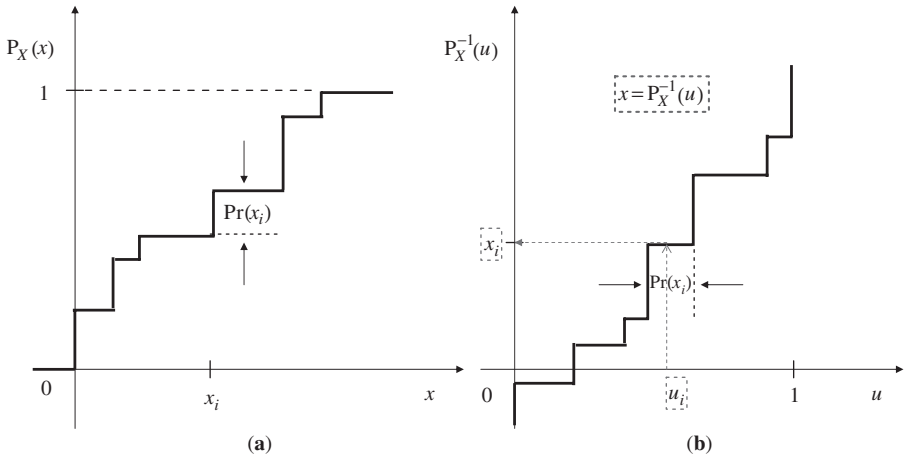


FIGURE 3.2 Empirical Distribution: (a) Estimated CDF. (b) Estimated ICDF.

random variates from both the analytic function as well as discrete samples from an empirical distribution. We will apply this technique frequently in the simulation-based processor. This approach is also used heavily in the resampling algorithms to follow (e.g., uniform, systematic, etc.). Numerical methods can also be used to perform the inversion by solving $P_X(X) - U = 0$ for X [5, 15].

Before we close this section, let us note that the above theorems can be generalized to simulate any distribution from any other distribution using the uniform variates as an intermediate step leading to the following corollary.

Corollary Given a distribution, $P_X(X)$ and a corresponding target distribution, $P_Y(Y)$, then there *exists* a unique, monotonic, transformation, $T(X)$, such that the random variable, $Y = T(X)$, is distributed as $P_Y(y)$, that is, if $T(X)$ is selected as

$$T(X) = Y = P_Y^{-1}(P_X(x))$$

then

$$P_Y(Y \leq y) = P_Y(y)$$

The proof of this corollary follows by applying the results of the previous theorems with $U = P_X(x)$ and $Y = P_Y^{-1}(U)$ for $U \sim \mathcal{U}(0, 1)$. Consider the following example to demonstrate this approach to simulation.

Example 3.5

Consider a random variable, X, distributed as $X \sim P_X(x) = 1 - e^{-x} - x\,e^{-x}$. We would like to generate a random variable, Y, exponentially distributed with parameter λ, $Y \sim \text{Exp}(\lambda)$. Then from the Corollary, we have

$$u = P_Y(y) = 1 - e^{-\lambda y} \quad \text{and} \quad y = P_Y^{-1}(U) = -\frac{1}{\lambda}\ln(1-u)$$

Thus it follows that

$$\mathrm{P}_Y^{-1}(\mathrm{P}_X(x)) = -\frac{1}{\lambda}\ln(1-\mathrm{P}_X(x)) = -\frac{1}{\lambda}\ln(e^{-x}+x\,e^{-x})$$

Therefore we have

$$y_i = -\frac{1}{\lambda}\ln(e^{-x_i}+x_i\,e^{-x_i})$$

Sampling from the uniform (as before), we first generate $\{x_i\}$ and then the desired set of random variates, $\{y_i\}$. △△△

We complete this subsection by stating the *Golden Rule of Sampling* [22] given by:

- Generate uniform samples: $u_i \to \mathcal{U}(0,1);\ i = 1,\ldots,N$;
- Define the transformation: $u_i = \mathrm{P}_X(x_i)$; and
- Apply the inverse transformation: $x_i = \mathrm{P}_X^{-1}(u_i)$.

Next we investigate a more general approach.

3.3.2 Rejection Sampling Method

In order to use the uniform sampling simulation methods of the previous section, we require the inverse *CDF*, which is not a simple task, even when the distribution is known in closed form. The rejection sampling method offers an alternative that not only eliminates the inversion problem, but also becomes an integral part of many of the sophisticated sampling algorithms to follow because of its simplicity and generality. In principle, the rejection method can be applied to *any* distribution with a density given up to a normalization constant [32].

The basic sampling problem in this context is that we are trying to generate samples from a density (or distribution) that is capable of being evaluated and we have a function

$$\mathrm{p}_X(x) = c \times \Pr(X)$$

where $\Pr(X)$ is the target distribution and $\mathrm{p}_X(x)$ is related and computable up to the normalization constant c which is *not* known. Suppose we select a sampling distribution, say $\mathrm{q}(x)$, such that there exists a "covering" constant M with

$$\mathrm{p}_X(x) \le Mq(x) \quad \forall x$$

The *rejection sampling method* is illustrated in Fig. 3.3 ($c = 1$) and summarized as:

- Generate a sample: $x_k \longrightarrow \mathrm{q}(x)$
- Generate a uniform sample: $u_k \longrightarrow \mathcal{U}(0,1)$
- ACCEPT the sample: $x_i = x_k$ if $u_k \le \frac{\mathrm{p}_X(x)}{M \times \mathrm{q}(x)}$
- Otherwise, REJECT the sample and generate the next trial sample: x_i

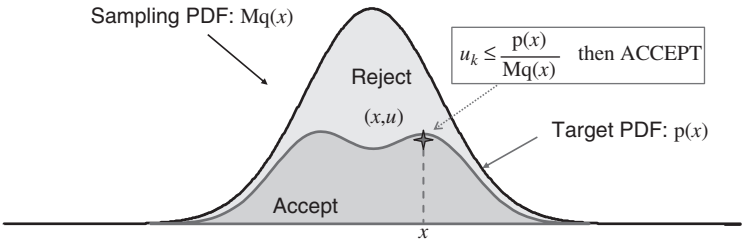


FIGURE 3.3 Rejection Sampling Method.

The proof of this method is given in Liu [32] using an *indicator* function, $\mathcal{I}(x)$, defined by

$$\mathcal{I}(x) = \begin{cases} 1 & \text{if } x \text{ Accepted} \\ 0 & \text{if } x \text{ Rejected} \end{cases} \tag{3.21}$$

Proceeding,

$$\Pr(\mathcal{I}(x) = 1) = \int \Pr(\mathcal{I}(x) = 1 | X = x)\, \mathrm{q}(x)\, dx = \int \left(\frac{c\Pr(x)}{M\mathrm{q}(x)} \right) \mathrm{q}(x)\, dx = \frac{c}{M}$$

and

$$\Pr(x | \mathcal{I}(x) = 1) = \frac{\left(\frac{c\Pr(x)}{M\mathrm{q}(x)} \right) \mathrm{q}(x)}{\Pr(\mathcal{I}(x) = 1)} = \Pr(x)$$

which shows that the acceptance of the sample corresponds to sampling directly from the target distribution, $\Pr(x)$.

The expected number of samples to accept an $x \sim \Pr(x)$ is M and therefore, the key to using this approach is to select a good proposal distribution $\mathrm{q}(x)$ with a low M—a nontrivial problem. Note that Example 1.1 in which the area of a circle was estimated using sampling methods is a simple geometric illustration of this methodology. The following example from Papoulis [1] demonstrates the method in an analytic form.

Example 3.6

Suppose we are given a random variable, $x \sim \mathrm{Exp}(1)$ and we would like to simulate the random variable with truncated Gaussian, $y \sim \mathcal{N}(0, 1)$, that is, we have

$$\mathrm{p}(x) = \frac{2}{\sqrt{2\pi}} e^{\frac{-x^2}{2}} \times \mu(x) \quad \text{and} \quad \mathrm{q}(x) = e^{-x}$$

where $\mu(x)$ is a unit step function as before, then for $x > 0$, we have

$$\frac{\mathrm{p}(x)}{\mathrm{q}(x)} = \sqrt{\frac{2e}{\pi}}\, e^{\frac{-(x-1)^2}{2}} \times e^1 \le \sqrt{\frac{2e}{\pi}}$$

Setting $M = \sqrt{\frac{\pi}{2e}}$, we have the acceptance/rejection sampling rule:

$$x_i = x_k \quad \text{if } u_k < e^{-\frac{(x_k-1)^2}{2}} \quad \text{(Accept)} \qquad \triangle\triangle\triangle$$

This completes the background on sampling theory, next we consider the Monte Carlo approach.

3.4 MONTE CARLO APPROACH

The Monte Carlo approach to solving Bayesian estimation problems is to replace complex analytic or unknown probability distributions with sample-based representations to solve a variety of "unsolvable" problems in inference (integration, normalization, marginalization, expectation, etc.), optimization (parameter, *MAP*, *ML* estimation), statistical mechanics (Boltzmann equation), nuclear physics (fission, diffusion, etc.) [32, 35]. The key to the *MC* approach is to generate independent random samples from a probability distribution, $\Pr(X)$, usually known only up to a normalizing constant (evidence) [32]. Typically, generating independent samples from this distribution is not feasible implying sample dependencies or using a proposal distribution, $q(X)$, that is similar to but not the exact target distribution, $\Pr(X)$. *Independent* (*i.i.d.*) samples are important because both strong and weak Laws of Large Numbers (mean converges in distribution to population mean) ensures that the inferences (e.g., mean, variance) can be made as accurate as desired by increasing the number of samples. However, the samples can be *dependent* and still properly reflect the probability of the target distribution opening the possibility of Markov chain methods (see Sec. 3.4.1).

The rejection method, just discussed, [22], importance sampling [16] and sampling-importance-resampling [26] are methodologies that do employ a proposal distribution. The Metropolis technique and its variants provide the foundation for this approach using Markov chain concepts generating dependent samples from a chain with $\Pr(X)$ as its invariant or stationary distribution. In this section we develop the idea of the *MC* simulation-based approach to Bayesian estimation using *iterative* rather than *sequential* Monte Carlo techniques.

Theoretically, the Monte Carlo approach to sampling is based on the following principles [10, 27, 29, 30, 35]. Here N *i.i.d.* samples, $\{X(i)\}_{i=1}^{N}$, are drawn from the *target density*, $p(X)$, to produce an estimate of the *empirical distribution* (density)

$$\hat{\Pr}(X) = \hat{p}_N(X) = \frac{1}{N}\sum_{i=1}^{N}\delta(X - X(i)) \tag{3.22}$$

which can be used to approximate integrals (pdfs, areas, etc.) with sums, that is,

$$I_N(f) = \frac{1}{N}\sum_{i=1}^{N} f(X)\,\delta(X - X(i)) = \frac{1}{N}\sum_{i=1}^{N} f(X(i)) \xrightarrow[N\to\infty]{\text{a.s.}} I(f) = \int f(X)\,p(X)\,dX \tag{3.23}$$

Here $I_N(f)$ is *unbiased* and *converges* (almost surely) to $I(f)$ according to the *strong Law of Large Numbers*. Its corresponding *variance* is bounded, ($\sigma_f^2 < \infty$) and

$$\text{var}(I_N(f)) = \frac{\sigma_f^2}{N} \tag{3.24}$$

A *central limit theorem* argument shows that the error converges in-distribution as

$$\sqrt{N}(I_N(f) - I(f)) \overset{N\to\infty}{\longrightarrow} \mathcal{N}(0, \sigma_f^2) \tag{3.25}$$

The main advantage of *MC* over deterministic integration is that it positions its samples over regions of *high probability*.

In signal processing, we are usually interested in some statistical measure of a random signal or parameter expressed in terms of its moments. Let us take a slightly more detailed look at just how *MC* concepts can be applied with this perspective in mind. Suppose we have some signal function, say $f(X)$, with respect to some underlying probabilistic distribution, $\Pr(X)$. Then a typical measure to seek is its performance "on the average" which is characterized by the expectation

$$E_X\{f(X)\} = \int f(X)\Pr(X)\,dX \tag{3.26}$$

From the Bayesian perspective, the embedded distribution can be thought of as the "posterior" of the signals/parameters. The Bayesian approach must integrate over high-dimensional probability distributions to make inferences about parameters or predictions about signals. Unless the integral is analytically tractable, the usual method of evaluation is through numerical integration (deterministic) techniques. Unfortunately the number of points to evaluate both $f(\cdot)$ and $\Pr(\cdot)$ increases exponentially with the dimensionality of the signal/parameter space. Also it is not possible to evaluate this integral over the entire space in practice; therefore, we concentrate on specific regions where the integrand is dense (not null). Instead of attempting to use numerical integration techniques, stochastic sampling techniques known as *Monte Carlo* (*MC*) *integration* have evolved as an alternative (see Fig. 1.2 for concept). As mentioned, the *key idea* embedded in the *MC* approach is to represent the required distribution as a set of random *samples* rather than a specific analytic function (e.g., Gaussian). As the number of samples becomes large, they provide an equivalent representation of the distribution enabling moments to be estimated directly. A functional estimate of the distribution could also be fit to the samples, if desired.

Integration has been used throughout statistics to evaluate probabilities and expectations. However, with Monte Carlo techniques the process is reversed and expectations are used to calculate integrals [1, 16, 28, 32]. Suppose we are asked to evaluate a multidimensional integral,

$$I = \int_X g(x)\,dx \tag{3.27}$$

Then an *MC* approach would be to factorize the integrand as

$$g(x) \longrightarrow f(x)\,p(x) \ni p(x) \geq 0 \quad \text{and} \quad \int p(x)\,dx = 1$$

where $p(x)$ is interpreted as a probability distribution in which samples can be drawn. This is the foundation of sampling techniques based on *MC* integration. Monte Carlo approaches draw samples from the required distribution and then form sample averages to approximate the sought after distributions, that is, they map integrals to discrete sums. Thus, *MC* integration evaluates Eq. 3.26 by drawing samples, $\{X(i)\}$ from $\Pr(X)$. Assuming perfect sampling, this produces the estimated or *empirical distribution* given by

$$\hat{\Pr}(X) \approx \frac{1}{N}\sum_{i=1}^{N} \delta(X - X(i)) \overset{N\to\infty}{\longrightarrow} \mathcal{N}(0, \sigma_f^2)$$

which is a probability distribution of mass or weights, $\frac{1}{N}$ and random variable or location $X(i)$. Substituting the empirical distribution into the integral gives

$$E_X\{f(X)\} = \int f(X)\hat{\Pr}(X)\,dX \approx \frac{1}{N}\sum_{i=1}^{N} f(X(i)) \equiv \bar{f} \tag{3.28}$$

which follows directly from the sifting property of the delta function. Here $\bar{f}$ is said to be a *Monte Carlo estimate* of $E_X\{f(X)\}$. Clearly, it is *unbiased*, since

$$E\{\bar{f}\} = \frac{1}{N}\sum_{i=1}^{N} E\{f(X(i))\} = E_X\{f(X)\}$$

with *variance* given by

$$\text{var}(\bar{f}) = \frac{1}{N}\int [f(X) - E_X\{f(X)\}]^2 \Pr(X)\,dX$$

Additionally, if the variance is *finite*, then the *central limit theorem* holds and the error in estimation converges to a zero-mean, Gaussian with $\mathcal{N}(0, \text{var}(\bar{f}))$. Consider the following examples to illustrate these concepts. First we discuss an analytic case, then a numerical case to solidify both *MC* approaches.

Example 3.7

Suppose we would like to solve for the integral

$$I = \int_0^1 f(x)\,dx$$

using the *MC* approach. Let u be defined as a random variate drawn from a uniform distribution, $u \sim \mathcal{U}(0, 1)$, then we can express the integral in terms of an expectation as

$$I = E\{f(u)\} = \int_0^1 f(u) \Pr(u)\, du$$

Generating a set of independent, identically distributed (*i.i.d.*) uniform samples, $u_1, \ldots, u_N$, then the corresponding set of functions, $\{f(u_i)\}$ are also *i.i.d.* with mean I as defined above. Therefore, the uniform sampling distribution is given by (equally likely)

$$\Pr(u = u_i) = \frac{1}{N} \sum_{i=1}^{N} \delta(u - u_i) \quad \text{for } 0 \leq u_i \leq 1$$

Substituting this expression into the expectation relation, we obtain

$$I = E\{f(u)\} = \int_0^1 f(u) \left[\frac{1}{N} \sum_{i=1}^{N} \delta(u - u_i) \right] du = \frac{1}{N} \sum_{i=1}^{N} f(u_i)$$

From the *Law of Large Numbers*[3] as $N \to \infty$, it follows that

$$\sum_{i=1}^{N} f(u_i) \longrightarrow E\{f(u)\} = I$$

Therefore, we can approximate the integral by generating a large number of random variables drawn from a uniform sampling distribution, transform them according to some functional relationship and calculate the sample mean to approximate the desired integral. △△△

This simple example illustrates the basic *MC* concept that will be applied throughout this text. Consider another example that illustrates this approach further by developing a simulation-based solution to a familiar statistical estimation problem.

Example 3.8

Suppose we have a Gaussian random variable and we would like to estimate its mean and variance. Knowledge that it is Gaussian enables us to write the closed form expression for the distribution

$$x \sim \mathcal{N}(m_x, \sigma_x^2) \quad \text{or} \quad \Pr(x) = \frac{1}{\sqrt{2\pi\sigma_x^2}} \exp\left\{ \frac{-(x - m_x)^2}{2\sigma_x^2} \right\}$$

[3] The Law of Large Numbers states simply that the mean converges with probability one to the population mean as the number of samples become large.

and the mean and the variance can be calculated analytically as

$$m_x = \int x \Pr(x)\, dx$$
$$\sigma_x^2 = \int (x - m_x)^2 \Pr(x)\, dx$$

In contrast, the Monte Carlo approach is to generate N samples from a Gaussian. Assuming perfect sampling, $x_i \longrightarrow \mathcal{N}(m_x, \sigma_x^2)$, and we have that

$$\hat{\Pr}(x) \approx \frac{1}{N} \sum_{i=1}^{N} \delta(x - x_i)$$

Now the mean and variance can be estimated from the samples directly using

$$\hat{m}_x = \int x \hat{\Pr}(x)\, dx = \int x \left(\frac{1}{N} \sum_{i=1}^{N} \delta(x - x_i) \right) dx = \frac{1}{N} \sum_{i=1}^{N} x_i$$

with variance

$$\hat{\sigma}_x^2 = \int (x - m_x)^2 \hat{\Pr}(x)\, dx = \int (x - m_x)^2 \left(\frac{1}{N} \sum_{i=1}^{N} \delta(x - x_i) \right) dx = \frac{1}{N} \sum_{i=1}^{N} (x_i - \hat{m}_x)^2$$

Summarizing the *MC* approach for this example, we must:

- Generate N samples from a Gaussian: $x_i \longrightarrow \mathcal{N}(m_x, \sigma_x^2)$;
- Estimate the desired statistics of the distribution from its samples as: $\hat{m}_x$ and $\hat{\sigma}_x^2$

We performed a simulation in *MATLAB* to generate Gaussian variates with $m_x = 2$ and $\sigma_x^2 = 4$. The results are shown in Fig. 3.4. In Fig. 3.4a we see a simulation for $N = 1000$ samples (+) with corresponding estimated mean (solid line) and upper and lower 95%-confidence limits about the mean ($m_x \pm 1.96\sigma_x$). The sample mean and variance for this *MC* realization were at: $\hat{m}_x = 1.97$ and $\hat{\sigma}_x^2 = 4.01$. The distribution was estimated using a histogram with 100-bins and is shown in Fig. 3.4b with a near perfect *MC* solution using $N = 10^6$ samples shown in the inset. △△△

So we see from this example how *MC* methods can be used to used to approximate (estimate) distributions and their associated statistics from simulated samples.

These methods are acceptable as long as high accuracy in not required. Monte Carlo techniques tend to perform a "divide and conquer" approach to integration by breaking the integral up into distinct regions around the integrand consisting of strong local peaks at known locations. They are typically very time consuming methods, on the order of tens and hundreds of thousands of points, and have only recently gained

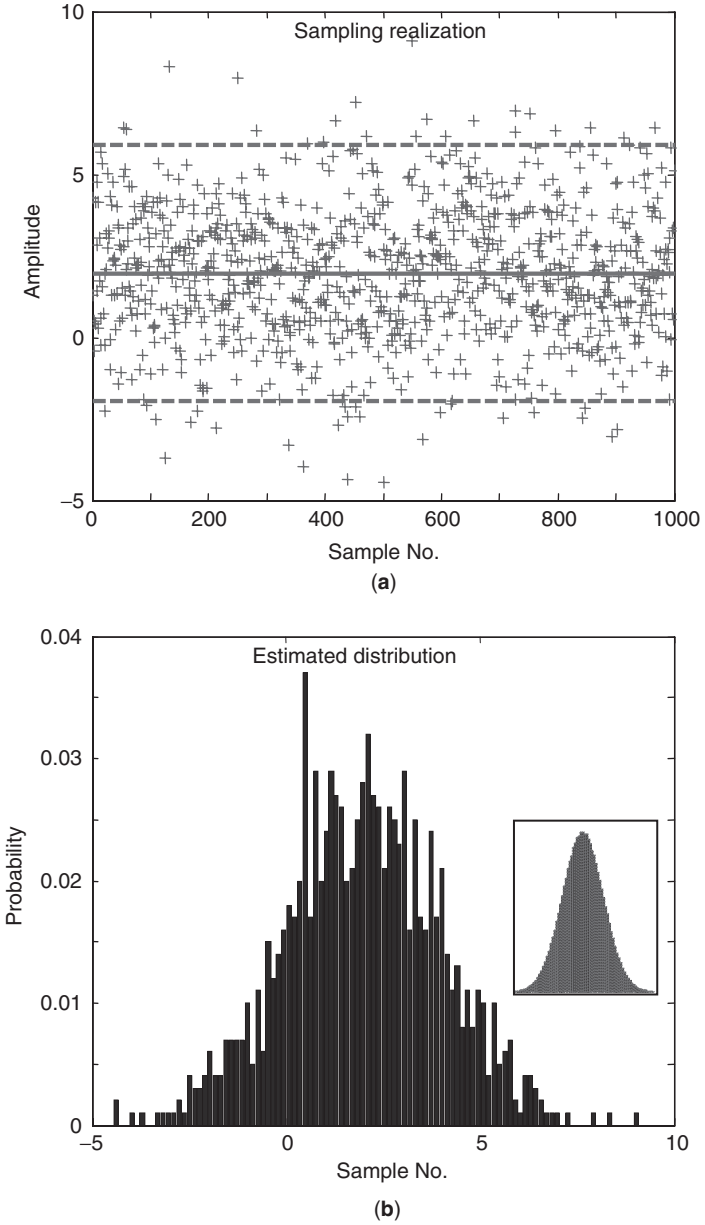


FIGURE 3.4 Monte Carlo approach for estimation of the statistics of a Gaussian random process: (a) Simulation ($m_x = 2, \sigma_x^2 = 4, N = 1000$) with 95% confidence limits (dashed line) about $\hat{m}_x$ (solid line). (b) Estimated distribution (histogram) from samples. Inset is estimate for $N = 1 \times 10^6$ samples.

notable prominence in the signal processing literature due to the major advances in fast computers [28]. Next let us consider the extension of Monte Carlo techniques using Markov chains.

3.4.1 Markov Chains

In *MC* integration, the population mean of $f(X)$ is estimated by the sample mean. When the samples $\{X_i(t) = \mathcal{X}_i\}$ are independent, the *Law of Large Numbers* ensures that the approximation can be made as accurately as required by increasing the number of samples, N. Generally, however, drawing samples from $\Pr(X)$ is not feasible especially when it is a non-standard distribution. However, dependent samples can be generated by any process that draws samples throughout the support (range) of the distribution. One efficient technique to accomplish this sampling is through a Markov chain having $\Pr(X)$ as its unique stationary or *invariant distribution*—this methodology is termed *Markov chain Monte Carlo* (*MCMC*). This technique is basically *MC* integration where the random samples are produced using a Markov chain. Recall that a *Markov chain* is a discrete random process possessing the property that the conditional distribution at the present sample (given all of the past samples) depends *only* on the previous sample (first-order), that is,

$$\Pr(X_i(t)|X_j(t-1), \ldots, X_k(0)) = \Pr(X_i(t)|X_j(t-1))$$

Summarizing, the Markov chain dynamics are represented by the *transition probability*, $\mathcal{P}_{ij}(t-1,t) := \Pr(X(t) = \mathcal{X}_i | X(t-1) = \mathcal{X}_j)$ denoting the probability that the state at time t will be $\mathcal{X}_i$ given that it is currently in state $\mathcal{X}_j$ at time $t-1$. Further, if the chain is also *homogeneous* in *time*, then $\mathcal{P}_{ij}(t-1,t)$ depends only on the time difference (in general) and therefore the transition probability is *stationary* such that $\mathcal{P}_{ij}(t-1,t) \to \mathcal{P}_{ij}$ with $\mathcal{P}_{ij} \geq 0$ and $\sum_{i=1}^{N_x} \mathcal{P}_{ij} = 1$ [1].

Thus, the basic requirement in Monte Carlo techniques is to generate random samples from a probability distribution or target distribution only known up to a normalizing constant. Typically it is not possible to generate the samples from the target but generating from a known trial distribution that is similar to the target distribution, just as in rejection sampling, is applied. In order to understand the *MCMC* approach we must first define critical properties of the Markov chain.

Markov chains possess certain crucial properties that must exist for them to be useful in *MC* simulations [32, 33]. A Markov chain begins with an initial distribution, $\Pr(X_i(0))$ and evolves to another indexed variable $X_i(t)$ determined by a transition kernel, that is, at index t we have

$$\Pr(X_i(t)) = \sum_j \overbrace{\Pr(X_i(t)|X_j(t-1))}^{\text{Transition Kernel}} \times \Pr(X_j(t-1)) \tag{3.29}$$

An *invariant distribution* is a fixed point solution to Eq. 3.29. For distribution estimation the constraint is even more stringent, since we require *time reversible*

chains which must satisfy a *detailed balance* as (ignoring the index)

$$\Pr(X_i(t)) \times \Pr(X_i(t)|X_j(t)) = \Pr(X_j(t)) \times \Pr(X_j(t)|X_i(t)) \quad \forall\, X_i(t),\, X_j(t) \tag{3.30}$$

which means that the transition probability from $X_i(t)$ to $X_j(t)$ is identical to the probability from $X_j(t)$ to $X_i(t)$ implying invariance [28, 32]. The chain is also required to be *ergodic* which means that regardless of the initial distribution, the probability at t *converges* to the invariant distribution as $t \to \infty$, that is,

$$\lim_{t\to\infty} \Pr(X_i(t)) \longrightarrow \Pr(X_i(t)) \tag{3.31}$$

Thus, *MCMC* methods for the simulation of a distribution is any technique producing an ergodic or reversible chain with invariant distribution being the desired target distribution. Armed with this information we can now discuss the most powerful and efficient *MCMC* methods: the Metropolis-Hastings and Gibbs sampler and their variants.

3.4.2 Metropolis-Hastings Sampling

Markov chain simulation is essentially a general technique based on generating samples from proposal distributions and then correcting (acceptance or rejection) those samples to approximate a target posterior distribution. Here we must *know* both the target, $\mathrm{p}_X(x)$ (up to a normalizing constant), and the proposal, $\mathrm{q}(x)$, *a priori*. The samples are sequentially generated forming a Markov chain with properties defined in the previous section. Typically, in Markov chain simulation, samples are generated from the transition kernel or distribution. The *key*, however, is not really the chain itself, but the fact that the approximate distribution improves sequentially as it converges to the target posterior.

In this subsection we discuss the basic Metropolis-Hastings sampling method. We start with the original Metropolis algorithm [24] and then introduce the Hastings generalization [23]. The fundamental idea is similar to the rejection method discussed previously. The Metropolis-Hastings (*M-H*) technique defines a Markov chain such that a new sample, x_i is generated from the previous samples, x_{i-1}, by first drawing a "candidate" sample, $\hat{x}_i$ from a proposal distribution, $\mathrm{q}(x)$, and then making a decision whether this candidate should be accepted and retained or rejected and discarded using the previous sample as the new. If accepted, $\hat{x}_i$ replaces x_i ($\hat{x}_i \to x_i$) otherwise the old sample x_{i-1} is saved ($x_{i-1} \to x_i$). This is the heart of the *M-H* approach in its simplest form.

We start with the basic *Metropolis* technique to describe the method:

- Initialize: $x_o \to \mathrm{p}_X(x_o)$
- Generate a candidate sample from proposal: $\hat{x}_i \to \mathrm{q}(x)$
- Calculate the acceptance probability: $\mathcal{A}(x_{i-1}, \hat{x}_i) = \min\left\{\frac{\mathrm{p}_X(\hat{x}_i)}{\mathrm{p}_X(x_{i-1})}, 1\right\}$

- ACCEPT candidate sample with probability, $\mathcal{A}(x_{i-1}, \hat{x}_i)$ according to:

$$x_i = \begin{cases} \hat{x}_i & \text{if } \mathrm{p}_X(\hat{x}_i) > \mathrm{p}_X(x_{i-1}) \\ x_{i-1} & \text{otherwise} \end{cases} \tag{3.32}$$

We see from this technique that when the candidate sample probability is greater than the previous sample's probability it is accepted with probability

$$\mathrm{p}_X(x_i) = \begin{cases} \mathcal{A}(x_i, \hat{x}_i) & \text{if Accepted} \\ 1 - \mathcal{A}(x_i, \hat{x}_i) & \text{if Rejected} \end{cases} \tag{3.33}$$

The idea is that we can detect when the chain has converged to its invariant distribution (posterior), when $\mathrm{p}_X(x_i) = \mathrm{p}_X(x_{i-1})$. It is clear from this discussion that in order for the chain to converge to the posterior it must be *reversible* and therefore $\mathrm{q}(x)$ must be a symmetric distribution. This was the original Metropolis assumption. Hastings [23] generalized this technique by removing the symmetry constraint enabling asymmetric proposals. The basic algorithm remains the same except that the acceptance probability becomes ($\mathrm{p}_X \to \mathrm{p}$)

$$\mathcal{A}(x_i, \hat{x}_i) = \min \left\{ \frac{\mathrm{p}(\hat{x}_i) \times \mathrm{q}(x_i|\hat{x}_i)}{\mathrm{p}(x_i) \times \mathrm{q}(\hat{x}_i|x_i)}, 1 \right\} \tag{3.34}$$

This process continues until the desired N-samples of the Markov chain have been generated. The critical step required to show that the *M-H* converges to the invariant distribution or equivalently the posterior, $\mathrm{p}_X(x)$, evolves directly from the detailed balance of Eq. 3.30 given by

$$\mathrm{p}(x_{i+1}|x_i) \times \mathrm{p}(x_i) = \mathrm{p}(x_i|x_{i+1}) \times \mathrm{p}(x_{i+1}) \tag{3.35}$$

where $\mathrm{p}(x_{i+1}|x_i)$ is the Markov chain transition probability. If we assume that the i^{th}-sample was generated from the posterior, $x_i \sim \mathrm{p}_X(x)$, then it is also assumed that the chain has converged and all subsequent samples have the same posterior. Thus, we must show that the next sample, x_{i+1} is also distributed $\mathrm{p}(x_i) = \mathrm{p}_X(x)$. Starting with the detailed balance definition above and integrating (summing) both sides with respect to x_i, it follows that

$$\begin{aligned} \int \mathrm{p}(x_{i+1}|x_i)\, \mathrm{p}(x_i)\, dx_i &= \int \mathrm{p}(x_i|x_{i+1})\, \mathrm{p}(x_{i+1})\, dx_i \\ &= \mathrm{p}(x_{i+1}) \int \mathrm{p}(x_i|x_{i+1})\, dx_i = \mathrm{p}(x_{i+1}) \end{aligned} \tag{3.36}$$

which shows that the relation on the left side of Eq. 3.36 gives the marginal distribution of x_{i+1} assuming x_i is from $\mathrm{p}(x_i)$. This implies that if the assumption that x_i is from $\mathrm{p}(x_i)$ is true, then x_{i+1} must also be from $\mathrm{p}(x_i)$ and therefore $\mathrm{p}(x_i) \to \mathrm{p}_X(x)$ is the

invariant distribution of the chain. Thus, once a sample is obtained from the invariant distribution, all subsequent samples will be from that distribution as well, proving that the invariant distribution is $p(x_i)$. A full proof of the *M-H* technique requires a proof that $p(x_i|x_0)$ will converge to the invariant distribution (see [19] for details).

For the *M-H* technique, the transition from $x_{i+1} \neq x_i$ to x_i occurs with probability

$$p(x_{i+1}|x_i) = q(x_{i+1}|x_i) \times \min\left\{\frac{p(\hat{x}_i) \times q(x_i|\hat{x}_i)}{p(x_i) \times q(\hat{x}_{i+1}|x_i)}, 1\right\} = \mathcal{A}(x_i, x_{i+1}) \times q(x_{i+1}|x_i) \tag{3.37}$$

This completes the basic *M-H* theory.

3.4.3 Random Walk Metropolis-Hastings Sampling

Next we consider another version of the *M-H* technique based on a random walk search through the parameter space. The idea is to perturb the current sample, x_i with an addition of a random error, that is,

$$x_{i+1} = x_i + \epsilon_i \quad \text{for } \epsilon_i \sim p_E(\epsilon) \tag{3.38}$$

where ϵ is *i.i.d.* A reasonable choice for this distribution is a symmetric Gaussian, that is, $p_E(\epsilon) \sim \mathcal{N}(0, \sigma_\epsilon^2)$. Thus, the *random walk M-H* method is:

Given the current sample, x_i,

- Draw a random sample: $\epsilon \rightarrow p_E(\epsilon)$
- Generate the candidate sample: $\hat{x}_i = x_i + \epsilon_i$
- Draw a uniform random sample: $u_i \rightarrow \mathcal{U}(0, 1)$
- Calculate the acceptance probability from the *known* densities: $\mathcal{A}(x_i, \hat{x}_i)$
- Update the sample:

$$x_{i+1} = \begin{cases} \hat{x}_i & \text{if } u_i < \mathcal{A}(x_i, \hat{x}_i) \\ x_i & \text{Otherwise} \end{cases} \tag{3.39}$$

- Select the next sample

With this algorithm, we must use both the (known) proposal and target distributions to calculate the acceptance probability and then generate samples (random walk) from the proposal. It is important to realize that a "good" proposal distribution can assure generating samples from the desired target distribution, but the samples must still be generated to "cover" its range. This is illustrated in Fig. 3.5 where our target distribution is a Gaussian mixture with mixing coefficients as: $(0.3, \mathcal{N}(0, 2.5);\ 0.5, \mathcal{N}(5, 1);\ 0.2, \mathcal{N}(10, 2.5))$. In the figure we see the results from choosing a reasonable proposal ($\mathcal{N}(0, 100)$) in the dark color generating enough samples to cover the range of the target distribution and a proposal ($\mathcal{N}(0, 10)$) that does not sample the entire space adequately leading to an erroneous target distribution characterizing the sampler output. Consider the following example of applying a variety of proposals and *M-H* techniques.

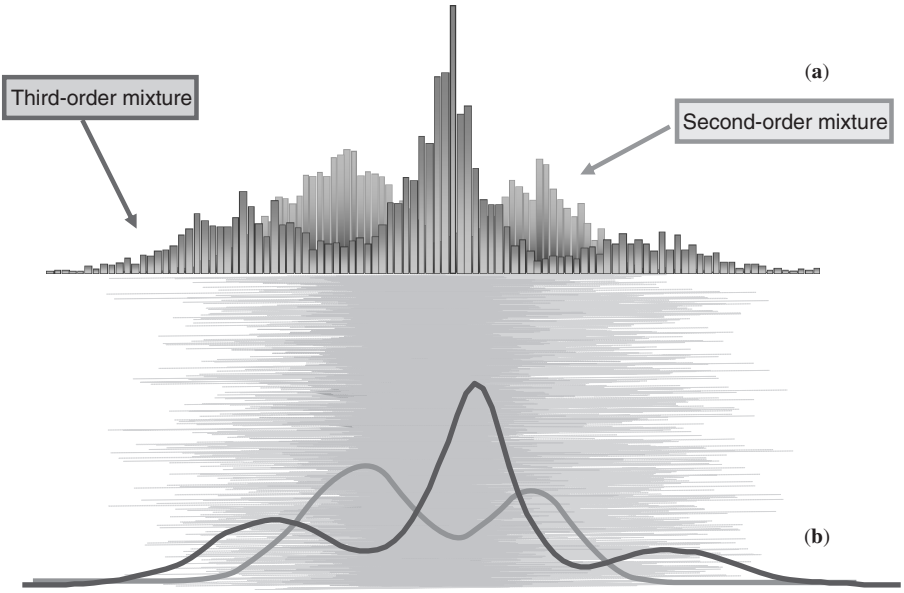


FIGURE 3.5 Metropolis-Hastings for Gaussian mixture distribution *PDF* estimates: (a) Inadequate proposal: $\mathcal{N}(0,10)$ at a 59.2% acceptance rate (light gray). (b) Adequate proposal: $\mathcal{N}(0,100)$ at a 34.0% acceptance rate (dark gray).

Example 3.9

Suppose we would like to generate samples from a unit Gaussian distribution, $\mathcal{N}(0,1)$, using the various *M-H* simulation techniques: (i) symmetric proposal using $\mathcal{U}(-10,10)$; (ii) symmetric proposal using a Student T distribution, $\mathcal{T}(1)$; (iii) random walk using uniform noise, $\mathcal{U}(-5,5)$; and (iv) uniform proposal using $\mathcal{U}(-4,9)$. To start we specify the "target" distribution as $\mathrm{p}_X(x) \sim \mathcal{N}(0,1)$ and we choose the following proposals:

- Case (i): $\mathrm{q}_1(x) \sim \mathcal{U}(-10,10) \Rightarrow \frac{1}{20}$ for $-10 < x < 10$
- Case (ii): $\mathrm{q}_2(x) \sim \mathcal{T}(1)$
- Case (iii): $x_i = x_{i-1} + u_i$ where $u_i \sim \mathcal{U}(-5,5) \Rightarrow \frac{1}{10}$ for $-5 < u_i < 5$
- Case (iv): $\mathrm{q}_4(x) \sim \mathcal{U}(-4,9) \Rightarrow \frac{1}{13}$ for $-4 < x < 9$

To implement the Metropolis, Metropolis-Hastings, Random Walk Metropolis-Hastings techniques we must:

1. Draw a random sample from the proposal: $x_i \rightarrow \mathrm{q}_i(x)$
2. Calculate the Acceptance Ratio: $\mathcal{A}(x_i, x_{i-1})$
3. Draw a uniform sample: $u_i \rightarrow \mathcal{U}(0,1)$
4. Accept or reject sample: $u_i \leq \mathcal{A}(x_i, x_{i-1})$

5. Update sample: x_i
6. Generate next sample: x_{i+1}

The results of these 10^5-sample simulations are shown in Fig. 3.6a–d using the *M-H*-sampler in *MATLAB*. We see the results of using the various proposals. All of the estimated densities give reasonably close estimates of the target posterior, $\mathcal{N}(0, 1)$. We estimated the corresponding posterior distribution from the samples using both histogram and kernel density estimators of Sec. 3.2 with all of the results reasonable. The standard deviation estimates were very close to unity in all cases; however, the means differed slightly from zero. It is interesting to note the corresponding acceptance rates (see the figure caption) with the most Gaussian-like, $\mathcal{T}$ proposal distribution had the highest acceptance rate of 57.8%. The true target distribution is superimposed for comparison purposes. *Acceptance rates* provide an indication of how "probable" a sample from the proposal is accepted as a sample in the target (posterior). When the proposal provides good coverage of the target distribution, then many samples are accepted at a high rate, if not the rate is low. Clearly, the *M-H* sampling technique and its variants provide a very robust method for generating samples from a target distribution especially when the proposal covers the entire range of the target sample space. △△△

There are a variety of *M-H* sampling techniques such as the independence sampler, the hybrid or dynamic (Hamiltonian) sampler, the multipoint samplers, etc. [32]. Also note that many of these methods are available in freeware (e.g. language-based *BUGS* [38] and the *MATLAB*-based, *NETLAB* [39], *PRTools* [40]). We summarize the *M-H* sampling technique is Table. 3.1. Next we discuss a popular special case of this approach—the Gibbs sampler.

3.4.4 Gibbs Sampling

The Gibbs simulation-based sampler (*G-S*), one of the most flexible of the sampling techniques available. It is a special case of the Metropolis-Hastings approach in which the acceptance probability, $\mathcal{A}(x_i, \hat{x}_i)$, is unity, that is, *all* samples are accepted [27]. Theoretically, the *G-S* is based on the fact that the targeted joint posterior distribution can be determined completely by a set of underlying conditional distributions evolving directly from Bayes' rule (joint $=$ conditional $\times$ marginal) [29]. It falls into the class of sampling algorithms termed, *block-at-a-time* or *variable-at-a-time* methods [23]. Proof of these methods have a significant practical implication, since it can be shown that the product of the transition distribution of the Markov chain is a product of conditional transitions which converge to joint posterior as its invariant distribution [30]. It is for this reason that the *G-S* is called "sampling-by-conditioning" [28], which will become obvious after we investigate its underlying structure. As before, it should be realized that both target and proposal distributions must be known (approximately) and samples must be easily generated from the proposal to be effective.

Gibbs sampling can be considered an implementation of the *M-H* technique on a component-by-component basis of a random vector. It is more restrictive than the *M-H* method, but can be more efficient leading to a simpler implementation. The

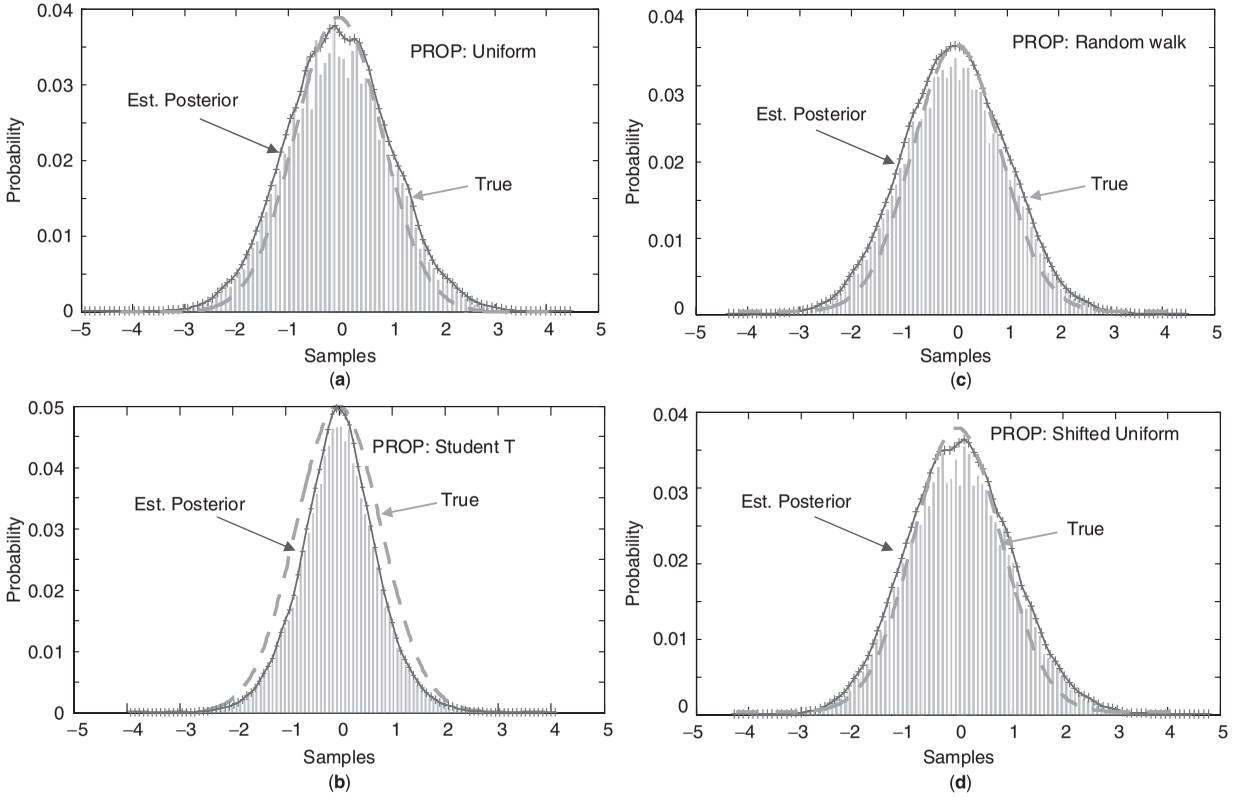


FIGURE 3.6 Metropolis-Hastings sampler posterior distribution estimates (10^5 samples): (a) *M-H* symmetric proposal: $\mathcal{U}(-10, 10)$ with mean and standard deviation estimates (−0.02, 0.998) at a 16.2% acceptance rate. (b) *M-H* symmetric proposal: $\mathcal{T}(1)$ with mean and standard deviation estimates (−0.004, 0.72) at a 57.8% acceptance rate. (c) *M-H* random walk: $\mathcal{U}(-5, 5)$ with mean and standard deviation estimates (−0.005,1.006) at a 31.7% acceptance rate. (d) Uniform proposal: $\mathcal{U}(-4, 9)$ with mean and standard deviation estimates (−0.03, 1.004) at a 24.2% acceptance rate.

TABLE 3.1 Metropolis-Hastings Sampling Algorithm

Initialize

$x_o \rightarrow \mathrm{p}_X(x_o)$ [draw sample]

Proposal

$\hat{x}_i \rightarrow \mathrm{q}(x)$ [draw sample]

Acceptance probability

$$\mathcal{A}(x_i, \hat{x}_i) = \min\left\{\frac{\mathrm{p}(\hat{x}_i) \times \mathrm{q}(x_i|\hat{x}_i)}{\mathrm{p}(x_i) \times \mathrm{q}(\hat{x}_i|x_i)}, 1\right\}$$

Uniform sample

$u_k \longrightarrow \mathcal{U}(0,1)$ [draw sample]

Decision

$$x_i = \begin{cases} \hat{x}_i & \text{if } u_k < \mathcal{A}(x_i, \hat{x}_i) \\ x_{i-1} & \text{otherwise} \end{cases}$$

Next sample

$x_{i+1} \longrightarrow x_i$ [draw sample]

G-S is especially important in Bayesian problems, since it is uniquely designed for multivariate problems, that is, it replaces sampling from a high-dimensional vector with sampling from low order component blocks [67]. It can be considered a concatination of *M-H* samplers, one for each component variable of the random vector. This decomposition has individual target distributions representing a conditional density or mass for *each* component given values for all of the other variables. Thus, the proposal for the component variable of the vector is the conditional density of that variable given the most recent values for all of the others.

More formally, suppose the random vector, $\mathbf{X} \in \mathcal{R}^{N_x \times 1}$ is decomposed into its components, X_k for $k = 1, \ldots, N_x$. Therefore, the idea is to generate, say $X_1(i)$, based on the conditional distribution, $\Pr(X_1|X_2(i-1), \ldots, X_{N_x}(i-1), Y)$ and the next sample drawn, $X_2(i)$, uses it and the samples available from the previous iteration to sample from, $\Pr(X_2|X_1(i) \cup X_3(i-1), \ldots, X_{N_x}(i-1), Y)$ and so forth so that at the i^{th}-iteration, we have the k^{th} component sample generated from

$$X_k(i) \longrightarrow \Pr(X_k|\{X_n(i)\} \cup \{X_m(i-1)\} \ni m > k; n < k, Y) \tag{3.40}$$

If we expand this relation, then we observe the underlying structure of the Gibbs sampler,

Given the sample set:, $\{\mathbf{X}(i-1)\}$, then

- Generate the first sample: $X_1 \rightarrow \Pr(X_1|X_2(i-1), \ldots, X_{N_x}(i-1), Y)$ and then
- Generate the next sample: $X_2 \rightarrow \Pr(X_2|X_1(i) \cup X_3(i-1), \ldots, X_{N_x}(i-1), Y)$
- Generate the k^{th}-sample: $X_k \rightarrow \Pr(X_k|\{X_{n-k}, \ldots, X_n(i)\}, \{X_{N_{m+k}}(i-1), \ldots, X_{m+k-N_x}\} \ni m > k; n < k, Y)$

So we see that the vector is decomposed component-wise and the corresponding conditional distributions evolve creating the vector sequence of iterates which are the realization of a Markov chain with transition distribution. We assume that we would like to go from $\mathbf{X}' \rightarrow \mathbf{X}$ giving the transition probability:

$$\begin{aligned}\Pr(\mathbf{X}', \mathbf{X}) = \Pr(X_1|X_2', \ldots, X_{N_x}', Y) \times \Pr(X_2|X_1, X_3', \ldots, X_{N_x}', Y) \times \cdots \\ \times \Pr(X_{N_x}|X_1, \ldots, |X_{N_x-1}, \ldots, X_{N_x}', Y)\end{aligned} \tag{3.41}$$

The *G-S* can be shown to satisfy the detailed balance. As a result it converges to the invariant distribution of the Markov chain which in this case is the joint posterior distribution [28, 32]. Consider the following example from [20].

Example 3.10

Suppose we have a measurement vector, $\mathbf{y}$ from a bivariate Gaussian with unknown mean and known covariance, that is,

$$\Pr(\mathbf{X}|\mathbf{Y}) = \mathcal{N}(\mathbf{Y}, \mathbf{R}_{xx}) \quad \text{for } \mathbf{R}_{xx} = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} \tag{3.42}$$

To apply the *G-S* to $\mathbf{X}$, we require the conditional posterior from the well-known properties of the bivariate Gaussian [1] given by

$$\begin{aligned}\Pr(X_1|X_2, \mathbf{Y}) &\sim \mathcal{N}(y_1 + \rho(x_2 - y_2), 1 - \rho^2) \\ \Pr(X_2|X_1, \mathbf{Y}) &\sim \mathcal{N}(y_2 + \rho(x_1 - y_1), 1 - \rho^2)\end{aligned}$$

Thus, the *G-S* proceeds by alternating samples from these Gaussian distributions, given $(x_1(0), x_2(0))$

Let $i = 1, 2, \ldots$

- Draw $x_1(i) \rightarrow \Pr(x_1|x_2(i-1), \mathbf{Y})$
- Draw $x_2(i) \rightarrow \Pr(x_2|x_1(i), \mathbf{Y})$

So we generate the pairs: $(x_1(1), x_2(1)), (x_1(2), x_2(2)), \ldots, (x_1(i), x_2(i))$ from the sampling distribution converging to the joint bivariate Gaussian, $\Pr(\mathbf{X})$ as the invariant distribution of the Markov chain. △△△

Next we consider a generalized variant of the Gibbs's sampler—the slice sampler.

3.4.5 Slice Sampling

Slice sampling (*S-S*) is a *MCMC* sampling technique based on the premise of sampling uniformly from the region under the target distribution, $\Pr(X)$ [19, 31, 32, 36, 37]. Therefore, like all of the previous sampling methods, it can be applied to any problem

for which the target distribution can be evaluated at a point, say x. It has an advantage over Metropolis methods being more robust to step-size variations especially since it performs adaptive adjustments. Slice sampling (*S-S*) is a generalized case of the Gibbs sampler based on iterative simulations of the *uniform* distribution consisting of one dimensional transitions. It is also similar to rejection sampling in that it draws samples from under the target distribution. However, in all of these cases the *S-S* is not bound by their restriction [37]. It is based on the following proposition proven in Cappe [36].

Proposition: Let $x \sim \Pr(X)$ and $u \sim \mathcal{U}(0, M)$, then the pair of random variables, (x, u) are distributed uniformly as, $(x, u) \sim \mathcal{U}(0, M \times \Pr(X))$. Conversely, if (x, u) is uniformly distributed, then x admits $\Pr(X)$ as its marginal distribution.

In its simplest form, the slice sampler technique generates uniform intervals that capture the samples:

- Initialize: x_{i-1}
- Draw uniform samples: $u_i \longrightarrow \mathcal{U}(0, \Pr(x_{i-1}))$
- Draw uniform samples: $x_i \longrightarrow \mathcal{U}(0, S(u_i))$ where $S(u_i) = \{x : \Pr(x_i) \geq u_i)$

The actual implementation of the algorithm is much more complex and we refer the interested reader to Neal [31] or MacKay [37] for more details. We merely state important features of the slice sampler technique.

The *S-S* involves establishing intervals to ensure the sample points of the target distribution are included by using an adaptive step-size (interval size) applying two techniques: (1) step-out technique; and (2) shrinking technique. The step-out technique is used to increase the size of the interval until the new sample, x_i is included, while the shrinking technique does the opposite, it decreases the interval size to assure the original sample x_{i-1} is included. Consider the following example from [19] to illustrate the *S-S*.

Example 3.11

Suppose we would like to generate samples from a unit Gaussian distribution, $x \sim \mathcal{N}(0, 1)$ using the slice sampling technique:

- Initialize: $x_{i-1} = 0$ (random draw)
- Draw uniform samples: $u_i \longrightarrow \mathcal{U}(0, \Pr(x_{i-1})) = \mathcal{U}\left(0, \frac{1}{\sqrt{2\pi}} e^{-x^2/2}\right)$
- Draw uniform samples: $x_i \longrightarrow \mathcal{U}(-\alpha, \alpha)$ where $\alpha = \sqrt{-\ln \sqrt{2\pi} u_i}$

Simulations can now be performed and analyzed. △△△

We conclude this section with the a signal processing example. We are given an autoregressive (all-pole) model and we would like to generate samples from the corresponding posterior distribution.

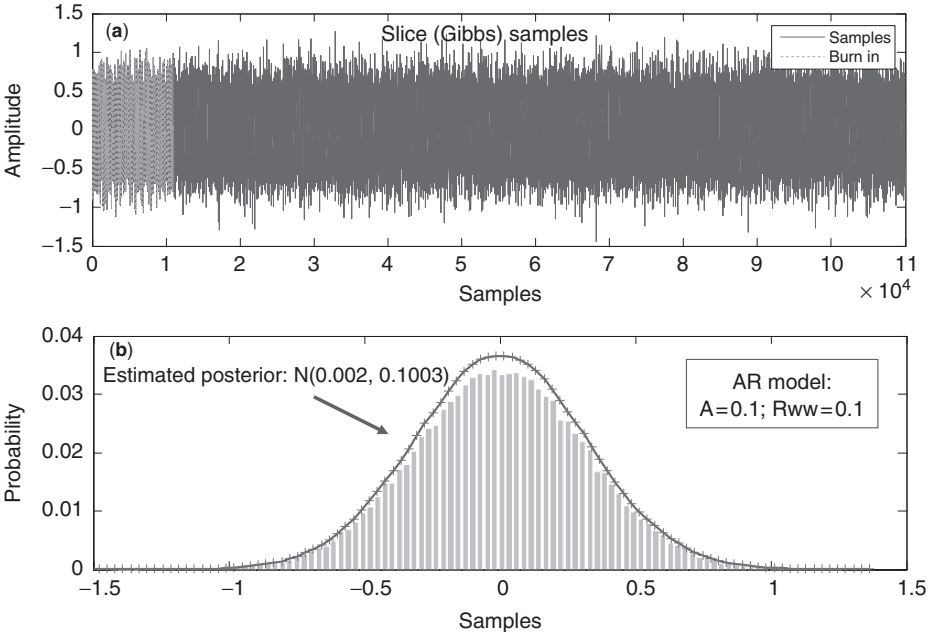


FIGURE 3.7 Gibbs (slice) sampler for autoregressive ($AR(1)$) model: (a) Simulated samples ($N = 10^5$). (b) Estimated posterior distribution: $\mathcal{N}(0.002, 0.1003)$ with 10% burn-in ($N = 10^4$)-samples.

Example 3.12

We have the following all-pole ($AR(1)$) model:

$$x(t) = -ax(t-1) + w(t-1) \quad \text{for } w \sim \mathcal{N}(0, R_{ww})$$

Suppose we choose the following parameters for the model: $N = 10^5$ samples, $a = 0.1$, $R_{ww} = 0.1$ and we would like to develop samples from the posterior. Analytically, we know that the posterior distribution is given by: $x \sim \mathcal{N}\left(0, \frac{R_{ww}}{1-a^2}\right) = \mathcal{N}(0, 0.101)$. We generate the samples using the slice (Gibbs) sampler with the proposal: $q(x) \sim \mathcal{N}(0, 0.1)$. The results are shown in Fig. 3.7. Using *MATLAB* we synthesized the set of samples and estimated their underlying distribution using both histogram and kernel density with a Gaussian window estimators. The samples including a 10% burn-in period are shown in Fig. 3.7a along with the estimated distribution in b. The sampler has a 100%-acceptance rate and the estimates are quite good with the posterior estimated at $\mathcal{N}(0.002, 0.1003)$. △△△

This concludes the section on sampling theory and iterative sampling techniques. Next we investigate the importance sampler that will lead to the sequential approaches required to construct Bayesian model-based processors.

3.5 IMPORTANCE SAMPLING

One way to mitigate difficulties with the inability to directly sample from a posterior distribution is based on the concept of importance sampling. *Importance sampling* is a method to compute expectations with respect to one distribution using random samples drawn from another. That is, it is a method of simulating samples from a proposal distribution to be used to approximate a targeted (posterior) distribution by appropriate weighting. Importance sampling is a generalization of the *MC* approach which evolves by rewriting Eq. 3.27 as:

$$I = \int_X f(x)\,dx = \int_X \left(\frac{f(x)}{q(x)}\right) \times q(x)\,dx \quad \text{for } \int q(x)\,dx = 1 \tag{3.43}$$

Here $q(x)$ is referred to as the sampling distribution or more appropriately the *importance sampling distribution*, since it samples the *target distribution*, $f(x)$ non-uniformly giving "more importance" to some values of $f(x)$ than others. We say that the *support* of $q(x)$ covers that of $f(x)$, or the samples drawn from $q(\cdot)$ overlap the same region (or more) corresponding to the samples of $f(\cdot)$ as illustrated previously in Fig. 3.3. That is, we say that $f(x)$ and $q(x)$ have the same support if

$$f(x) > 0 \;\Rightarrow\; q(x) > 0 \quad \forall x \in R^{N_x \times 1}$$

a *necessary* condition for importance sampling to hold. If we interpret the prior of Fig. 1.1 as the proposal, $q(x)$ and the posterior as the target, $f(x)$, then this figure provides a visual example of coverage.

The integral in Eq. 3.43 can be estimated by:

- drawing N-samples from $q(x)$: $X(i) \sim q(x)$ and $\hat{q}(x) \approx \frac{1}{N}\sum_{i=1}^{N} \delta(x - X(i))$; and
- computing the sample mean [28],

$$I = E_q\left\{\frac{f(x)}{q(x)}\right\} \approx \int \left(\frac{f(x)}{q(x)}\right) \times \frac{1}{N}\sum_{i=1}^{N} \delta(x - X(i))\,dx = \frac{1}{N}\sum_{i=1}^{N} \frac{f(X(i))}{q(X(i))}$$

with corresponding error variance

$$\text{Var}\left[E_q\left\{\frac{f(x)}{q(x)}\right\}\right] = \int \left(\frac{f(x)}{q(x)} - I\right)^2 \times q(x)\,dx$$

It is interesting to note that the *MC* approach provides an unbiased estimator with the corresponding error variance easily calculated from the above relation.

Consider the case where we would like to estimate the expectation of the function of X given by $f(X)$. Then choosing an importance distribution, $q(x)$, that is similar to $f(x)$ with covering support gives the expectation estimator

$$E_p\{f(x)\} = \int_X f(x) \times p(x)\,dx = \int_X f(x)\left(\frac{p(x)}{q(x)}\right) \times q(x)\,dx \tag{3.44}$$

If we draw samples, $\{X(i)\},\ i = 0, 1, \ldots, N$ from the importance distribution, $q(x)$ and compute the sample mean, then we obtain the importance sampling estimator. That is, assume the perfect sampler, $\hat{q}(x) \approx \frac{1}{N}\sum_{i=1}^{N} \delta(x - X(i))$, and substitute

$$E_p\{f(x)\} = \int_X f(x)\left(\frac{p(x)}{q(x)}\right) \times \hat{q}(x)\, dx \approx \frac{1}{N}\sum_{i=1}^{N} f(X(i)) \times \left(\frac{p(X(i))}{q(X(i))}\right) \tag{3.45}$$

demonstrating the concept.

The "art" in importance sampling is in choosing the importance distribution, $q(\cdot)$ that approximates the target distribution, $p(\cdot)$, as closely as possible. This is the *principal factor* effecting performance of this approach, since variates must be drawn from $q(x)$ that cover the target distribution. Using the concepts of importance sampling, we can approximate the posterior distribution with a function on a finite discrete support. Since it is usually not possible to sample directly from the posterior, we use *importance sampling* coupled with an easy to sample proposal distribution, $q(X_t|Y_t)$—this is the crucial choice and design step required in Bayesian importance sampling methodology. Here $X_t = \{x(0), \ldots, x(t)\}$ represents the set of dynamic variables and $Y_t = \{y(0), \ldots, y(t)\}$, the set of measured data. Therefore, starting with a function of the set of variables, say $f(X_t)$, we would like to estimate its mean using the importance concept. That is, using the *MC* approach, we would like to sample from this posterior directly and then use sample statistics to perform the estimation. Therefore we insert the proposal importance distribution, $q(X_t|Y_t)$ as before

$$\hat{f}(t) := E\{f(X_t)\} = \int f(X_t)\left[\frac{\Pr(X_t|Y_t)}{q(X_t|Y_t)}\right] \times q(X_t|Y_t)\, dX_t \tag{3.46}$$

Now applying Bayes' rule to the posterior target distribution, and defining a weighting function as

$$\tilde{W}(t) := \frac{\Pr(X_t|Y_t)}{q(X_t|Y_t)} = \frac{\Pr(Y_t|X_t) \times \Pr(X_t)}{\Pr(Y_t) \times q(X_t|Y_t)} \tag{3.47}$$

Unfortunately, $\tilde{W}(t)$ is not useful because it requires knowledge of the evidence or normalizing constant $\Pr(Y_t)$ given by

$$\Pr(Y_t) = \int \Pr(Y_t|X_t) \times \Pr(X_t)\, dX_t \tag{3.48}$$

which is usually not available. But by substituting Eq. 3.47 into Eq. 3.46 and defining a *new* weight, $W(t)$, as

$$W(t) \propto \frac{\Pr(X_t|Y_t)}{q(X_t|Y_t)} = \frac{\Pr(Y_t|X_t) \times \Pr(X_t)}{q(X_t|Y_t)} \tag{3.49}$$

we obtain

$$\hat{f}(t) = \frac{1}{\Pr(Y_t)} \int f(X_t) \left[\frac{\Pr(Y_t|X_t) \times \Pr(X_t)}{q(X_t|Y_t)} \right] q(X_t|Y_t)\, dX_t$$
$$= \frac{1}{\Pr(Y_t)} \int W(t) f(X_t)\, q(X_t|Y_t)\, dX_t \tag{3.50}$$

which is simply the expectation of the weighted function, $E_q\{W(t)f(X_t)\}$ scaled by the normalizing constant. From this definition of the new weighting function in Eq. 3.49, we have

$$W(t) \times q(X_t|Y_t) = \Pr(Y_t|X_t) \times \Pr(X_t) \tag{3.51}$$

Thus, we can now replace the troublesome normalizing constant of Eq. 3.48 using Eq. 3.51, that is,

$$\hat{f}(t) = \frac{E_q\{W(t)f(X_t)\}}{\Pr(Y_t)} = \frac{E_q\{W(t)f(X_t)\}}{\int W(t) \times q(X_t|Y_t)\, dX_t} = \frac{E_q\{W(t)f(X_t)\}}{E_q\{W(t)\}} \tag{3.52}$$

Now drawing samples from the proposal $X_t(i) \sim q(X_t|Y_t)$ and using the *MC* approach leads to the desired result. That is, from the "perfect" sampling distribution, we have that

$$\hat{q}(X_t|Y_t) \approx \frac{1}{N} \sum_{i=1}^{N} \delta(X_t - X_t(i)) \tag{3.53}$$

and therefore substituting, applying the sifting property of the Dirac delta function and defining the "normalized" weights

$$\mathcal{W}_i(t) := \frac{W_i(t)}{\sum_{i=1}^{N} W_i(t)} \quad \text{for } W_i(t) = \frac{\Pr(Y_t|X_t(i)) \times \Pr(X_t(i))}{q(X_t(i)|Y_t)} \tag{3.54}$$

we obtain the final estimate

$$\hat{f}(t) \approx \sum_{i=1}^{N} \mathcal{W}_i(t) \times f(X_t(i)) \tag{3.55}$$

The importance estimator is biased being the ratio of two sample estimators (as in Eq. 3.52), but it can be shown that it asymptotically converges to the true statistic and the central limit theorem holds [32, 49]. Thus, as the number of samples increase ($N \to \infty$), an asymptotically optimal estimate of the posterior is

$$\hat{\Pr}(X_t|Y_t) \approx \sum_{i=1}^{N} \mathcal{W}_i(t) \times \delta(X_t - X_t(i)) \tag{3.56}$$

which is the goal of Bayesian estimation. Note that the new weight is, $W(t) \propto \tilde{W}(t)$ where $\propto$ is defined as "proportional to" up to a normalizing constant.

3.6 SEQUENTIAL IMPORTANCE SAMPLING

Suppose we would like to develop a sequential version [41–60] of the batch Bayesian importance sampling estimator of the previous section. The importance distribution can be modified to enable a sequential estimation of the desired posterior distribution, that is, we estimate the posterior, $\hat{\Pr}(X_{t-1}|Y_{t-1})$ using importance weights, $\mathcal{W}(t-1)$. As a new sample becomes available, we estimate the new weight, $\mathcal{W}(t)$ leading to an updated estimate of the posterior, $\hat{\Pr}(X_t|Y_t)$. This means that in order to obtain the new set of samples, $X_t(i) \sim q(X_t|Y_t)$ sequentially, we must use the previous set of samples, $X_{t-1}(i) \sim q(X_{t-1}|Y_{t-1})$. Thus, with this in mind, the importance distribution, $q(X_t|Y_t)$ must admit a marginal distribution $q(X_{t-1}|Y_{t-1})$ implying the following Bayesian factorization

$$q(X_t|Y_t) = q(X_{t-1}|Y_{t-1}) \times q(x(t)|X_{t-1}, Y_t) \tag{3.57}$$

which satisfies the probability chain rule decomposition

$$q(X_t|Y_t) = \prod_{k=0}^{t} q(x(k)|X_{t-k}, Y_k) \tag{3.58}$$

Now let us see how this type of importance distribution choice can lead to the desired sequential solution [58]. We start with the definition of the unnormalized weight of Eq. 3.49 and substitute into Eq. 3.57 while applying Bayes' rule to the numerator. The resulting weighting function is

$$W(t) = \frac{\Pr(Y_t|X_t) \times \Pr(X_t)}{q(X_{t-1}|Y_{t-1}) \times q(x(t)|X_{t-1}, Y_t)} \tag{3.59}$$

Motivated by the definition of $W(t-1)$, we multiply and divide by the Bayesian factor $\Pr(Y_{t-1}|X_{t-1}) \times \Pr(X_{t-1})$ and group to obtain

$$W(t) = \underbrace{\left[\frac{\Pr(Y_{t-1}|X_{t-1}) \times \Pr(X_{t-1})}{q(X_{t-1}|Y_{t-1})}\right]}_{\text{Previous Weight}} \times \frac{\Pr(Y_t|X_t) \times \Pr(X_t)}{[\Pr(Y_{t-1}|X_{t-1}) \times \Pr(X_{t-1})] \times q(x(t)|X_{t-1}, Y_t)}$$

Therefore we can write

$$W(t) = W(t-1) \times \frac{\Pr(Y_t|X_t) \times \Pr(X_t)}{\Pr(Y_{t-1}|X_{t-1}) \times \Pr(X_{t-1}) \times q(x(t)|X_{t-1}, Y_t)} \tag{3.60}$$

Using the probabilistic chain rule of Eq. 2.63 for each of the conditionals and imposing the Markov property of the dynamic variable along with the conditional

independence conditions of the measurements, we obtain

$$\begin{aligned}\Pr(Y_t|X_t) &= \prod_{k=0}^{t} \Pr(y(k)|x(k)) = \Pr(y(t)|x(t)) \prod_{k=0}^{t-1} \Pr(y(k)|x(k)) \\ \Pr(X_t) &= \prod_{k=0}^{t} \Pr(x(k)|x(k-1)) = \Pr(x(t)|x(t-1)) \prod_{k=0}^{t-1} \Pr(x(k)|x(k-1))\end{aligned} \tag{3.61}$$

Therefore, recognizing the relationship between these expansions as well as those at $t-1$ and factoring the t-th term (as shown above), we can cancel both numerator and denominator chains to extract the final recursions, that is,

$$\begin{aligned}W(t) = W(t-1) \times & \frac{\Pr(y(t)|x(t)) \left[\prod_{k=0}^{t-1} \Pr(y(k)|x(k))\right]}{\left[\prod_{k=0}^{t-1} \Pr(y(k)|x(k))\right]} \\ & \times \frac{\Pr(x(t)|x(t-1)) \left[\prod_{k=0}^{t-1} \Pr(x(k)|x(k-1))\right]}{\left[\prod_{k=0}^{t-1} \Pr(x(k)|x(k-1))\right]} \times \frac{1}{q(x(t)|X_{t-1}, Y_t)}\end{aligned} \tag{3.62}$$

which gives the final recursion

$$W(t) = W(t-1) \times \frac{\Pr(y(t)|x(t)) \times \Pr(x(t)|x(t-1))}{q(x(t)|X_{t-1}, Y_t)} \tag{3.63}$$

Another way of developing this relationship is to recall the Bayesian solution to the batch posterior estimation problem in Eq. 2.79. We have

$$\Pr(X_t|Y_t) = \left[\frac{\Pr(y(t)|x(t)) \times \Pr(x(t)|x(t-1))}{\Pr(y(t)|Y_{t-1})}\right] \times \Pr(X_{t-1}|Y_{t-1})$$

or recognizing the denominator as just the evidence or normalizing distribution and not a function of X_t, we have

$$\Pr(X_t|Y_t) = C \times \Pr(y(t)|x(t)) \times \Pr(x(t)|x(t-1)) \times \Pr(X_{t-1}|Y_{t-1}) \tag{3.64}$$

or simply

$$\Pr(X_t|Y_t) \propto \Pr(y(t)|x(t)) \times \Pr(x(t)|x(t-1)) \times \Pr(X_{t-1}|Y_{t-1}) \tag{3.65}$$

Substituting this expression for the posterior in the weight relation as before, we have

$$W(t) \propto \frac{\Pr(Y_t|X_t)}{q(X_t|Y_t)} = \frac{\Pr(y(t)|x(t)) \times \Pr(x(t)|x(t-1))}{q(x(t)|X_{t-1}, Y_t)} \times \underbrace{\frac{\Pr(X_{t-1}|Y_{t-1})}{q(X_{t-1}|Y_{t-1})}}_{\text{Previous Weight}} \tag{3.66}$$

giving the desired expression of Eq. 3.63

These results enable us to formulate a generic Bayesian *sequential importance sampling* algorithm:

1. Draw samples from the proposed importance distribution: $x_i(t) \rightarrow q(x(t)|X_{t-1}, Y_t)$;
2. Determine the required conditional distributions: $\Pr(x_i(t)|x(t-1)), \Pr(y(t)|x_i(t))$;
3. Calculate the unnormalized weights: $W_i(t)$ using Eq. 3.63 with $x(t) \rightarrow x_i(t)$;
4. Normalize the weights: $\mathcal{W}_i(t)$ in Eq. 3.54; and
5. Estimate the posterior distribution: $\hat{\Pr}(X_t|Y_t) = \sum_{i=1}^{N} \mathcal{W}_i(t)\delta(x(t) - x_i(t))$

Once the posterior is estimated, then the desired statistics evolve directly. We summarize the generic sequential importance sampling in Table 3.2.

TABLE 3.2 Bayesian Sequential Importance Sampling Algorithm

Initialize

$x_i(0) \sim q(x(0)|y(0)); \quad i = 1, \ldots, N_p$ [sample prior]

$$W_i(0) = \frac{\Pr(y(0)|x_i(0)) \times \Pr(x_i(0))}{\Pr(x_i(0)|y(0))}$$ [weights]

$$\mathcal{W}_i(0) = \frac{W_i(0)}{\sum_{i=1}^{N_p} W_i(0)}$$ [normalize]

Importance sampling

Sample

$x_i(t) \sim q(x(t)|X_{t-1}, Y_t); \quad i = 1, \ldots, N_p$ [sample]

Weight Update

$$W_i(t) = W_i(t-1) \times \frac{\Pr(y(t)|x_i(t)) \times \Pr(x(t)|x_i(t))}{q(x_i(t)|X_i(t-1), Y_t)}$$ [weights]

Weight Normalization

$$\mathcal{W}_i(t) = \frac{W_i(t)}{\sum_{i=1}^{N_p} W_i(t)}$$

Distribution

$$\hat{\Pr}(x(t)|Y_t) \approx \sum_{i=1}^{N_p} \mathcal{W}_i(t)\delta(x(t) - x_i(t))$$ [posterior distribution]

Introducing these ideas of Bayesian importance sampling, we are now ready to consider applying this approach to variety of models which we discuss in the next chapter.

3.7 SUMMARY

In this chapter we discussed the importance of simulation-based sampling methods for nonlinear signal processing. Starting with the basics of *PDF* estimation and statistical sampling theory, we motivated statistical approaches to sampling both when we have analytical expressions for the distributions and we do not have them and must resort to pure sampling methodologies. We discussed the uniform sampling and rejection sampling methods examining their inherent advantages and disadvantages. We showed how these approaches led to more sophisticated techniques evolving from Markov chain theory and leading to the Metropolis-Hastings sampler. In certain cases the Gibbs sampler, a variant of the Metropolis-Hastings approach, was developed and discussed along with its variant—the slice sampler. All of these methods fall into the general class of iterative methods. Next we concentrated on the importance sampling approach leading to its recursive version—the sequential importance sampler which is the workhorse of this text.

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. *MATLAB* has a *Statistics Toolbox* that incorporates a large suite of *PDF*s and *CDF*s as well as "inverse" *CDF* functions ideal for simulation-based algorithms. The **mhsample** command incorporate the Metropolis, Metropolis-Hastings and Metropolis independence samplers in a single command while the Gibbs sampling approach is adequately represented by the more efficient slice sampler (**slicesample**). There are even specific "tools" for sampling as well as the inverse *CDF* method captured in the **randsample** command. *PDF* estimators include the usual histogram (**hist**) as well as the sophisticated kernel density estimator (**ksdensity**) offering a variety of kernel (window) functions (Gaussian, etc.) and ICDF techniques. As yet no sequential algorithms are available. Type *help stats* in *MATLAB* to get more details or go to the MathWorks website.

REFERENCES

1. A. Papoulis and S. Pillai, *Probability, Random Variables and Stochastic Processes* (New York: McGraw-Hill, 2002).
2. P. Peebles, *Probability, Random Variables and Random Signal Parameters*, 4^{th} *Ed.* (New York: McGraw-Hill, 2001).

3. R. Duda and P. Hart, *Pattern Classsification* (New York: Wiley, 2000).
4. T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning* (New York: Wiley, 2001).
5. S. Theodoridis and K. Koutroumbas, *Pattern Recognition* (New York: Academic Press, 1999).
6. E. Parzen, "On estimation of a probability density function and mode," *Ann. Math. Stat.*, **33**, 1065–1076, 1962.
7. H. Shimazaki and S. Shinomoto, "A method for selecting the bin size of a time histogram," *Neural Computation*, **19**, 1503–1527, 2007.
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. B. Anderson and J. Moore, *Optimal Filtering* (Englewood Cliff, NJ: Prentice-Hall, 1979).
11. J. Candy, *Model-Based Signal Processing* (Hoboken, NJ: Wiley/IEEE Press, 2006).
12. H. Tanizaki, *Nonlinear Filters* (New York: Springer-Verlag, 1994).
13. I. Rhodes, "A tutorial introduction to estimation and filtering," *IEEE Trans. Autom. Contr.*, **AC-16**, 1971.
14. J. Hammersley and K. Morton, "Poor man's Monte Carlo," *Symposium on Monte Carlo Methods*, Royal Statistical Society, pp. 23–38, 1954.
15. S. Ross, *A Short Course in Simulation* (New York: McMillan, 1990).
16. M. Tanner, *Tools for Statistical Inference*, *2nd Ed.* (New York: Springer-Verlag, 1993).
17. W. Gilks, S. Richardson and D. Spiegelhalter, *Markov Chain Monte Carlo in Practice* (New York: Chapman & Hall/CRC, 1996).
18. A. Smith and A. Gelfand, "Bayesian statistics without tears: a sampling-resampling perspective," *Am. Statistician*, **44**, 4, 84–88, 1992.
19. C. Robert and G. Casella, *Monte Carlo Statistical Methods* (New York: Springer, 1999).
20. A. Gelman, J. Carlin, H. Stern and D. Rubin, *Bayesian Data Analysis*, *2nd Ed.* (New York: Chapman & Hall/CRC, 2004).
21. M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis* (Cambridge: Cambridge University Press, 2005).
22. J. von Neumann, "Various techniques used in connecton with random digits," *Nat. Bureau Standards Applied Math. Series*, **12**, 36–38, 1951.
23. W. Hastings, "Monte Carlo sampling methods using Markov chains and their applications," *Biometrika*, **57**, 97–109, 1970.
24. N. Metropolis, N. Rosenbutt, A. Rosenbutt, M. Teller and A. Teller, "Equations of state calculations by fast computing machines," *J. Chem. Phys.*, **21**, 1087–1092, 1953.
25. S. Geman and D. Geman, "Stochastic relaxation: Gibbs distributions and Bayesian restoration of images," *IEEE Trans. Patten Analy. and Mach. Intell.*, **6**, 721–741, 1984.
26. D. Rubin, "A noniterative sampling/importance resampling alternative to the data augmentation algorithm for creating a few imputations when fractions of missing information are modest: the SIR algorithm," *J. Amer. Stat. Assoc.*, **52**, 543–546, 1987.
27. A. Gelfand and A. Smith, "Sampling-based approaches to calculating marginal densities," *J. Amer. Stat. Assoc.*, **85**, 410, 398–409, 1990.

28. J. Ruanaidh and W. Fitzgerald, *Numerical Bayesian Methods Applied to Signal Processing* (New York: Springer-Verlag, 1996).
29. G. Casella and E. George, "Explaining the Gibbs sampler," *Am. Statistician*, **46**, 3, 167–174, 1992.
30. S. Chib and E. Greenberg, "Understanding the Metropolis-Hastings algorithm," *Am. Statistician*, **49**, 4, 327–335, 1995.
31. R. Neal, "Slice sampling," *Annals of Statistics*, **31**, 3, 705–767, 2003.
32. J. Liu, *Monte Carlo Strategies in Scientific Computing* (New York: Springer-Verlag, 2001).
33. W. Fitzgerald, "Markov chain Monte Carlo methods with applications to signal processing," *Signal Proc.*, **81**, 3–18, 2001.
34. C. Andrieu, N. de Freitas, A. Doucet and M. Jordan, "An introduction to MCMC for machine learning," *Mach. Learn.*, **50**, 5–43, 2003.
35. D. Frenkel, "Introduction to Monte Carlo methods," J. von Neumann Inst. Comput. NIC series, Vol. 23, 29–60, 2004.
36. O. Cappe, E. Moulines and T. Ryden, *Inference in Hidden Markov Models* (New York: Springer-Verlag, 2005).
37. D. MacKay, *Information Theory, Inference and Learning Algorithms* (Cambridge, UK: Cambridge Univ. Press, 2006).
38. D. Spiegelhalter, A. Thomas, N. Best and D. Lunn, *WinBUGS User Manual*, Imperial College: London, UK, 2003 (website: **http://www.mrc-bsu.cam.ac.uk**).
39. I. Nabney, *NETLAB Algorithms for Pattern Recognition* (New York: Springer, 2001).
40. R. Duin, P. Juszczak, P. Paclik, E. Pekalska, D. de Ridder and D. Tax, *PRTools4: A MATLAB Toolbox for Pattern Recognition*, Delft University: Delft, Netherlands, 2004 (website: **http://prtools.org**).
41. A. Doucet and X. Wang, "Monte Carlo methods for signal processing," *IEEE Signal Proc. Mag.* **24**, 5, 152–170, 2005.
42. G. Kitagawa, "Non-Gaussian modeling of nonstationary time series," *J. Am. Statistical Assoc.*, **82**, 400, 1032–1063, 1987.
43. G. Kitagawa, "A nonlinear smoothing method for time series analysis," *Statistica Sinica*, **1**, 2, 371–388, 1991.
44. N. Gordon, D. Salmond and A. Smith, "A novel approach to nonlinear non-Gaussian Bayesian state estimation," *IEE Proc. F.*, **140**, pp. 107–113, 1993.
45. A. Kong, J. Liu and W. Wong, "Sequential imputations and Bayesian missing data problems," *J. Am. Statistical Assoc.*, **89**, 425, 278–288, 1994.
46. J. Liu and R. Chen, "Blind deconvolution via sequential imputations," *J. Am. Statistical Assoc.*, **90**, 430, 567–576, 1995.
47. G. Kitagawa, "Monte Carlo filter and smoother for non-Gaussian nonlinear state-space models," *J. Comput. Graphical Stat.*, **5**, 1, 1–25, 1996.
48. G. Kitagawa and W. Gersch, *Smoothness Priors Analysis of Time Series* (New York: Springer-Verlag, 1996).
49. M. West and J. Harrison, *Bayesian Forcasting and Dynamic Models*, *2nd Ed.* (New York: Springer-Verlag, 1997).
50. J. Liu and R. Chen, "Sequential Monte Carlo methods for dynamic systems," *J. Am. Statistical Assoc.*, **93**, 443, 1032–1044, 1998.

51. G. Kitagawa, "Self-organizing state space model," *J. Am. Statistical Assoc.*, **93**, 443, 1203–1215, 1998.
52. M. Isard and A. Blake, "Condensation–conditional density propagation for visual tracking," *Int. J. Comput. Vis.*, **29**, 1, 5–28, 1998.
53. J. Liu, R. Chen and W. Wong, "Rejection control and sequential importance sampling," *J. Am. Statistical Assoc.*, **93**, 443, 1022–1031, 1998.
54. M. Pitt and N. Shephard, "Filtering via simulation: Auxiliary particle filters," *J. Am. Statistical Assoc.*, **94**, 446, 590–599, 1999.
55. A. Doucet, S. Godsill and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Statist. Comput.*, **10**, 3, 197–208, 2000.
56. A. Doucet, N. de Freitas and N. Gordon, *Sequential Monte Carlo Methods in Practice* (New York: Springer-Verlag, 2001).
57. S. Godsill and P. Djuric, "Special Issue: Monte Carlo methods for statistical signal processing." *IEEE Trans. Signal Proc.*, **50**, 2002.
58. M. Arulampalam, S. Maskell, N. Gordon and T. Clapp "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking." *IEEE Trans. Signal Proc.*, **50**, 2, 174–188, 2002.
59. 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.
60. B. Ristic, S. Arulampalam and N. Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Applications* (Boston: Artech House, 2004).
61. S. Haykin and N. de Freitas, "Special Issue: Sequential state estimation: from Kalman filters to particle filters." *Proc. IEEE*, **92**, 3, 2004.
62. C. Andrieu, A. Doucet, S. Singh, and V. Tadic, "Particle methods for change detection, system identification and control," *Proc. IEEE*, **92**, 3, 423–438, 2004.
63. A. Harvey, S. Koopman and N. Shephard, *State Space and Unobserved Component Models* (Cambridge: Cambridge University Press, 2004).
64. H. Sorenson and D. Alspach, "Recursive Bayesian estimation using Gaussian sums," *Automatica*, **7**, 465–479, 1971.
65. R. van der Merwe, A. Doucet, N. de Freitas and E. Wan, "The unscented particle filter," in *Advances in Neural Information Processing Systems 13*, MIT Press: Cambridge, MA, 2000.
66. S. Haykin, *Kalman Filtering and Neural Networks* (Hoboken, NJ: Wiley, 2001).
67. J. Spall, "Estimation via Markov chain Monte Carlo," *IEEE Control Sys. Magz.*, **4**, 34–45, 2003.

PROBLEMS

3.1 Let x be a discrete random variable with probability mass function (*PMF*)

$$\mathrm{p}_X(x) = \frac{3!}{x!\,(3-x)!}\left(\frac{2}{3}\right)^x\left(\frac{1}{3}\right)^{3-x} \qquad x = 0, 1, 2, 3$$

What is the *PMF* of y, $\mathrm{p}_Y(y)$, when $y = x^2$?

3.2 Let z be the distance from the origin to a random point selected within the unit circle ($x^2 + y^2 < 1$). Let $z = x^2 + y^2$, then

(a) What is the probability of the selected point lying within the unit circle?

(b) What is the *CDF* of z? What is its *PDF*?

(c) Suppose $w = z^2$, what is $\mathrm{p}_W(w)$? $\mathrm{P}_W(w)$?

3.3 Let $x \sim \mathcal{U}(0, 1)$ and $y = -2 \ln x$, what is $\mathrm{p}_X(x)$?

3.4 Suppose we have a bivariate distribution with point (X,Y) from a unit square such that

$$\mathrm{p}_{XY}(x, y) = 1, \quad 0 < x < 1, \quad 0 < y < 1$$

Let $z = x + y$, what is the *CDF* of z? *PDF* of z?

3.5 A source emits particles that decay at a distance x ($x \sim \mathcal{E}xp(\lambda)$) from the source. The measurement instrument can only observe these events in a window of length 20 cm ($x = 1 - 20$ cm). N decays are observed at locations: $\{x_i\} = \{1, \ldots, N\}$. Using Bayes' rule [37]

(a) What is the characteristic length λ?

(b) Plot the distributions for $\{x_i\} = \{1.5, 2, 3, 4, 5, 12\}$.

3.6 For a particular television show, a contestant is given the following instructions:

- There are three doors labeled 1, 2, 3 with a prize hidden behind one of them. Select one of the doors, but it will NOT be opened.
- The host will open one of the other two doors, but will NOT reveal the prize should it be there.
- The contestant must now make a decision to keep his original choice or choose another door.
- All the doors will then be opened and the prize revealed

What should the contestant do?

(a) Stay with the original choice?

(b) Switch to the remaining door?

(c) Does it make any difference?

(*Hint:* Use Bayes' rule to answer these questions)

3.7 Suppose we would like to simulate a random variable X such that $\Pr(X = i) = \{0.2, 0.15, 0.25, 0.40\}$.

(a) Sketch out an algorithm using the inverse transform method to generate realizations of X choosing an *ascending* approach for $X = 1$, $X = 2$, $X = 3$ and $X = 4$.

(b) Sketch out an algorithm using the inverse transform method to generate realizations of X choosing an *descending* approach for $X = 4$, $X = 3$, $X = 1$ and $X = 2$.

(c) Which approach is more efficient? Why?

3.8 We would like to simulate the value of a discrete random variable X with associated probabilities: $\Pr(X=i)=\{0.11, 0.12, 0.09, 0.08, 0.12, 0.10, 0.09, 0.09, 0.10, 0.10\}$. Using the rejection method with $M=\max \frac{\Pr(X=i)}{\Pr(u_i)}$ for $u_i \sim \mathcal{U}(0,10)$

(*a*) Sketch out rejection sampling algorithm for this problem.

(*b*) Using this approach synthesize 1000 samples and estimate the histogram? Estimate the kernel density? (*Hint*: *MATLAB* has the commands **hist** and **ksdensity** to perform these operations).

(*c*) Does the estimated distribution appear to be any classical closed form (e.g. Poisson)? If so which one?

3.9 Suppose the continuous random variable X has cumulative distribution, $P_X(x)=x^k$. Using the inverse transform approach, sketch the methodology to synthesize X. How would you do this using the rejection approach? Generate 1000 samples and estimate the distributions.

3.10 Use the rejection sampling method to generate the continuous random variable x with density $\mathrm{p}_X(x)=20(x(1-x)^3, \quad 0<x<1$. (*Hint:* select to be $q(x)=1$ and find the maximum ratio of the densities to obtain M).

3.11 We would like to compute the solution to the integral

$$\mathcal{I}=\int_a^b f(x)\,dx$$

Using the results of Ex. 3.14, develop the more general *MC* solution.

3.12 Suppose we have a bivariate Gaussian distribution with unknown mean $\mu=[\mu_1\ \mu_2]'$ and known covariance matrix

$$C=\begin{bmatrix}1 & p\\ p & 1\end{bmatrix}$$

and a uniform prior on μ. A single observation (y_1, y_2) then has Gaussian posterior:

$$\Pr(\mu|Y)\sim\mathcal{N}(Y,C) \quad \text{for } Y=[y_1\ y_2]'$$

Sketch out the Gibbs sampler algorithm for this problem.

3.13 Develop the Metropolis algorithm for a bivariate unit Gaussian target distribution, $\mathcal{N}(\Theta:0,I))$ with prior $\Pr(\Theta_o)$ (e.g. $\mathcal{U}(0,1)$). The proposal distribution is bivariate Gaussian also, $\mathcal{N}(\Theta^*:0,(1/5)^2I))$.

3.14 We would like to use *MATLAB* to simulate the Metropolis-Hastings sampler (**mhsample**) for the following case with the target distribution being a standard Gaussian, $\mathcal{N}(5,1)$ and the proposal Rayleigh distributed with parameter, $b=1.5$. Use a 10% sample burn-in. Compare these results to the slice sampler

(**slicesample**). Use (**ksdensity**) to estimate the distribution of the resulting samples for each algorithm for the comparison.

3.15 Develop the Metropolis-Hastings sampler for a 2^{nd}-order autoregressive model ($AR(2)$) with known coefficients, $\{a_1, a_2\} = \{1, -0.5\}$ driven by Gaussian noise with $\epsilon \sim \mathcal{N}(0, 1)$.

(a) Develop the exact likelihood for the parameters, $\Pr(Y|\mathbf{a}, \sigma^2)$.

(b) What is the posterior distribution for the parameters, $\Pr(\mathbf{a}, \sigma^2|Y)$? (*Hint*: Assume the prior is just an indicator function)

(c) Develop the *M-H*-sampler for this problem.

3.16 Suppose we would like to generate samples from a bivariate Gaussian with mean vector *zero* and covariance

$$C = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

(a) Choose a uniform proposal: $\mathcal{U}(-3, 3)$ and develop the *M-H*-sampler algorithm with a 5% burn-in and $N = 10,000$ samples. (*Hint*: Use *MATLAB* **mhsample** command).

(b) Using the *MC*-approach, estimate the expected value: $E\{f(\mathbf{X})\} = [1 \quad 1]\mathbf{X}$

(c) Compare these results to those obtained using the slice-sampler. (*Hint*: Use *MATLAB* **slicesample** command).

3.17 Set up the Gibbs sampler (*G-S*) for a joint (X, Y) exponential distribution on an interval of length $(0, I)$. Estimate the marginal distribution of X, $\Pr(X)$, and compare the results to a simulated data set based on $N = 500$ samples.

4

STATE–SPACE MODELS FOR BAYESIAN PROCESSING

4.1 INTRODUCTION

In this chapter we investigate the development of models for Bayesian estimation [1–15] using primarily the state–space representation—a versatile and robust model especially for random signals. We start with the definition of state and the basic principles underlying these characterizations and then show how they are incorporated as propagation distributions for Bayesian processors in the following chapter. We review the basics of state–space model development with all of their associated properties starting with the continuous-time processes, then sampled-data systems and finally proceeding to the discrete-time state–space. Next we develop the stochastic version leading to Gauss-Markov representations when the models are driven by white noise and then proceed to the nonlinear case [6–15]. Here we again drive the models with white Gaussian noise, but the results are not necessarily Gaussian. We develop linearization techniques based on Taylor-series expansions to arrive at linearized Gauss-Markov models.

State–space models are easily generalized to multichannel, nonstationary, and nonlinear processes. They are very popular for model-based signal processing primarily because most physical phenomena modeled by mathematical relations naturally occur in state–space form (see [13] for details). With this motivation in mind, let us proceed to investigate the 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 sampled-data followed by the discrete-time representation—the primary focus of this text. We begin by formally defining the concept of state [1].

Bayesian Signal Processing. By James V. Candy

4.2 CONTINUOUS-TIME STATE–SPACE MODELS

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 $t \in [t_0, \infty)$. 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 ensure that the smallest number of "independent" states have been defined in order to avoid possible violation of some important system theoretic properties [2, 3].

Let us consider a general *deterministic* formulation of a *nonlinear dynamic system* including the output (measurement) model in state–space form (continuous-time)[1]

$$\begin{aligned} \dot{x}_t &= A(x_t, u_t) = a(x_t) + b(u_t) \\ y_t &= C(x_t, u_t) = c(x_t) + d(u_t) \end{aligned}$$

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, measurement (output) and feedthrough functions. The N_x-dimensional system and input functions are defined by $a(\cdot)$ and $b(\cdot)$, while the N_y-dimensional output and feedthrough 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 dimension of 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

$$\begin{aligned} \dot{x}_t &= A_t x_t + B_t u_t \\ y_t &= C_t x_t + D_t u_t \end{aligned} \tag{4.1}$$

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 feedthrough 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 only their coefficients differ. 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 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

[1] We separate x_t and u_t for future models, but it is not really necessary.

system under investigation. This is one of the major reasons why state–space models are employed in signal processing, especially when the system is *multivariable* 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 (deterministic) *linear time-invariant* (*LTI*) *state–space* model given by

$$\begin{aligned} \dot{x}_t &= A_c x_t + B_c u_t \\ y_t &= C_c x_t + D_c u_t \end{aligned} \tag{4.2}$$

where $A_t \to A_c$, $B_t \to B_c$, $C_t \to C_c$ and $D_t \to D_c$ their time invariant counterparts with the subscript, "c", annotating continuous-time matrices.

This *LTI* model corresponds the constant coefficient differential equation solutions which can be solved using Laplace transforms. Taking the Laplace transform of these equations, we have that

$$sX(s) - x_{t_0} = A_c X(s) + B_c U(s)$$

and solving for $X(s)$

$$X(s) = (sI - A_c)^{-1} x_{t_0} + (sI - A_c)^{-1} B_c U(s) \tag{4.3}$$

where $I \in \mathcal{R}^{N_x \times N_x}$ is the identity matrix. The corresponding output is

$$Y(s) = C_c X(s) + D_c U(s) \tag{4.4}$$

Therefore, combining these relations, we obtain

$$Y(s) = [C_c(sI - A_c)^{-1} B_c + D_c]U(s) + C_c(sI - A_c)^{-1} x_{t_0} \tag{4.5}$$

From the definition of transfer function (zero initial conditions), we have the desired result

$$H(s) = C_c(sI - A_c)^{-1} B_c + D_c \tag{4.6}$$

Taking the inverse Laplace transform of this equation gives us the corresponding *impulse response matrix* of the *LTI*-system as [1]

$$H(t, \tau) = C_c e^{A_c(t-\tau)} B_c + D_c \delta(t - \tau) \quad \text{for } t \geq \tau \tag{4.7}$$

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 that frequently appear in the signal processing literature.

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. First we simplify the notation by defining the Laplace transform of the state transition matrix or the so-called *resolvent matrix* of systems theory [1, 3] as

$$\Phi_c(s) := (sI - A_c)^{-1} \tag{4.8}$$

Therefore we can rewrite the transfer function matrix as

$$H(s) = C_c \Phi_c(s) B_c + D_c \tag{4.9}$$

and the corresponding state–input transfer matrix by

$$X(s) = \Phi_c(s) x_{t_0} + \Phi_c(s) B_c U(s) \tag{4.10}$$

Taking the inverse Laplace transformation gives the time domain solution

$$x_t = \mathcal{L}^{-1}[X(s)] = \Phi_c(t, t_0) x_{t_0} + \Phi_c(t, t_0) B_c * u_t$$

or

$$x_t = \underbrace{\Phi_c(t, t_0) x_{t_0}}_{zero-input} + \underbrace{\int_{t_0}^{t} \Phi_c(t, \alpha) B_c u_\alpha \, d\alpha}_{zero-state} \tag{4.11}$$

with corresponding output solution

$$y_t = C_c \Phi_c(t, t_0) x_{t_0} + \int_{t_0}^{t} C_c \Phi_c(t, \alpha) B_c u_\alpha \, d\alpha \tag{4.12}$$

The *state transition matrix*, $\Phi_c(t, t_0)$, 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 (homogeneous) vector-matrix state equations

$$\dot{x}_t = A_c x_t \tag{4.13}$$

It is well known from linear algebra [1] that this equation has the matrix exponential as its solution

$$x_t = \Phi_c(t, t_0) x_{t_0} = e^{A_c(t - t_0)} x_{t_0} \tag{4.14}$$

The meaning of "transition" is now clear since knowledge of the transition matrix $\Phi_c(t, t_0)$ enables us to calculate the transition of the state vector from time t_0 to any $t > t_0$. Taking the Laplace transform of this equation gives

$$X(s) = \Phi_c(s) x_{t_0} = (sI - A_c)^{-1} x_{t_0}$$

with

$$e^{A_c t} = \mathcal{L}^{-1}[(sI - A_c)^{-1}] \tag{4.15}$$

We have that the *state transition matrix* for a *LTI* system is

$$\Phi_c(t, t_0) = e^{A_c(t-t_0)}, \quad t \geq t_0 \tag{4.16}$$

Revisting the continuous-time system of Eq. 4.11 and substituting the matrix exponential for the state transition matrix gives the *LTI* solution as

$$x_t = e^{A_c(t-t_0)} x_{t_0} + \int_{t_0}^{t} e^{A_c(t-\alpha)} B_c u_\alpha \, d\alpha \tag{4.17}$$

with corresponding measurement system

$$y_t = C_c x_t \tag{4.18}$$

In general, the state transition matrix satisfies the following *properties* [1, 2]:

1. $\Phi_c(t, t_0)$ is uniquely defined for $t, t_0 \in [0, \infty)$ [Unique]
2. $\Phi_c(t, t) = I$ [Identity]
3. $\Phi_c(t)$ satisfies the matrix differential equation:

$$\dot{\Phi}_c(t, t_0) = A_t \Phi_c(t, t_0), \qquad \Phi_c(t_0, t_0) = I, \quad t \geq t_0 \tag{4.19}$$

4. $\Phi_c(t, t_0) = \Phi_c(t, \tau) \times \Phi_c(\tau, \alpha) \times \cdots \times \Phi_c(\beta, t_0)$ [Semi-Group]
5. $\Phi_c(t, \tau)^{-1} = \Phi_c(\tau, t)$ [Inverse]

Thus, the transition matrix plays a pivotal role in *LTI* systems theory for the analysis and prediction of the response of linear time-invariant and time-varying systems [2]. For instance, the poles of a *LTI* govern such important properties as 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 resolvent, that is,

$$|\Phi_c(s)| = |(sI - A_c)|_{s=p_i} = 0 \tag{4.20}$$

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 its response as:

$$y_t = \sum_{i=1}^{N_x} K_i \, e^{-p_i t} \tag{4.21}$$

where A_c is diagonal which in this case is given by $A_c = \text{diag}[p_1, p_2, \ldots, p_{N_x}]$, the eigenvalues of A_c. Next we consider the sampled-data, state–space representation.

4.3 SAMPLED-DATA STATE–SPACE MODELS

Sampling a continuous-time system is commonplace with the advent of high speed analog-to-digital converters (*ADC*) and modern computers. A sampled-data system lies somewhere between the continuous analog domain (physical system) and the purely discrete domain (stock market prices). Since we are strictly sampling a continuous-time process, we must assure that all of its properties are preserved. The well-known *Nyquist sampling theorem* precisely expresses the required conditions to achieve "perfect" reconstruction of the process from its samples [13].

Thus, if we have a physical system governed by continuous-time dynamics and we "sample" it at given time instants, then a sampled-data model can be obtained directly from the solution of the continuous-time state–space model. That is, we know from the previous section that

$$x_t = \Phi_c(t, t_0)x_{t_0} + \int_{t_0}^{t} \Phi_c(t, \alpha)B_c(\alpha)u_\alpha \, d\alpha$$

where $\Phi_c(\cdot, \cdot)$ is the continuous-time state transition matrix that satisfies the matrix differential equation

$$\dot{\Phi}_c(t, t_0) = A_t\Phi_c(t, t_0), \qquad \Phi_c(t_0, t_0) = I, \quad t \geq t_0$$

Sampling this system such that $t \to t_k$ over the interval $(t_k, t_{k-1}]$, then we have the corresponding *sampling interval* defined by $\Delta t_k := t_k - t_{k-1}$. Note this representation need *not* necessarily be equally spaced—another important property of the state–space representation. Thus the sampled solution becomes (with notation change)

$$x(t_k) = \Phi(t_k, t_{k-1})x(t_{k-1}) + \int_{t_{k-1}}^{t_k} \Phi(t_k, \alpha)B_c(\alpha)u_\alpha \, d\alpha \tag{4.22}$$

and therefore from the differential equation above, we have the solution

$$\Phi(t_k, t_{k-1}) = \int_{t_{k-1}}^{t_k} A(\alpha)\Phi(t_k, \alpha) \, d\alpha \quad \text{for } \Phi(t_0, t_0) = I \tag{4.23}$$

where $\Phi(t_k, t_{k-1})$ is the *sampled-data* state transition matrix—the critical component in the solution of the state equations enabling us to calculate the state evolution in time.

If we further assume that the input excitation is *piecewise constant* ($u_\alpha \to u(t_{k-1})$) over the interval $(t_k, t_{k-1}]$, then it can be removed from under the superposition integral in Eq. 4.22 to give

$$x(t_k) = \Phi(t_k, t_{k-1})x(t_{k-1}) + \left(\int_{t_{k-1}}^{t_k} \Phi(t_k, \alpha)B_c(\alpha) \, d\alpha \right) \times u(t_{k-1}) \tag{4.24}$$

Under this assumption, we can define the sampled *input transmission* matrix as

$$B(t_{k-1}) := \int_{t_{k-1}}^{t_k} \Phi(t_k, \alpha) B_c(\alpha)\, d\alpha \tag{4.25}$$

and therefore the *sampled-data state–space system* with equally or unequally sampled data is given by:

$$\begin{aligned} x(t_k) &= \Phi(t_k, t_{k-1}) x(t_{k-1}) + B(t_{k-1}) u(t_{k-1}) \\ y(t_k) &= C(t_k) x(t_k) \end{aligned} \tag{4.26}$$

Computationally, sampled-data systems pose no particular problems when care is taken, especially since reasonable approximation and numerical integration methods exist [17]. For instance, the following three methods of solving for the state transition and input transmission matrices can be quite effective. If we constrain the system to be *LTI*, then we have that the matrix exponential can be represented by the Taylor series expansion

$$e^{A_c \Delta t_k} = \sum_{i=0}^{\infty} \frac{(A_c\, \Delta t_k)^i}{i!} \tag{4.27}$$

Truncating the series is possible at an acceptable error magnitude [17]. This is called the *series approach* to estimating the state transition matrix and can be determined for a finite sum. For example, a simple first-order approximation uses the relations:

$$\begin{aligned} \Phi(t_k, t_{k-1}) &\approx (I + \Delta t_k A_c) \\ B(t_k) &\approx \Delta t_k B_c \end{aligned} \tag{4.28}$$

This direct approach can yield unsatisfactory results; however, one improved solution is based on the *Pade' approximation* incorporating a scaling and squaring technique [17, 18]. This *scaling and squaring* property of the matrix exponential is given by

$$e^{A_c \Delta t_k} = (e^{A_c \Delta t_k / m})^m \tag{4.29}$$

and is based on choosing the integer m to be a power-of-two such that the exponential term can reliably and efficiently be calculated followed by repeated squaring. A typical criterion is to choose $\|A\|/m \ll 1$ yielding a very effective numerical technique for either Taylor or Pade' approximants.

Ordinary differential equation methods using *numerical integration* techniques (e.g., Runge-Kutta, Gear's method, etc.) offer another practical approach to solving for the state transition matrix and the corresponding input matrices as given by Eqs. 4.23 and 4.25, respectively. The advantages of numerical integration techniques are reliability and accuracy as well as applicability to time-varying and nonlinear systems. The major disadvantage is computational time which can be very high for stiff

(large eigenvalue spread) differential equations and variable integration step-sizes [17]. In any case, the sampled-data system has the property that it has evolved from a system with continuous dynamics and must be accurately approximated or numerically integrated to produce reliable solutions.

The final class of methods we discuss briefly are the *matrix decomposition methods* [18]. These methods are based on *similarity transformations*, that is,

$$\tilde{A}_c = TA_cT^{-1} \quad \text{and} \quad e^{\tilde{A}_c\Delta t} = T\, e^{A_c\Delta t}T^{-1} \tag{4.30}$$

where $\Delta t := t - \tau$ in the continuous-time case. If the similarity transformation matrix is chosen to be an *eigenvalue–eigenvector* transformation say, $T = V$, then

$$e^{\tilde{A}_c\Delta t} = V\, e^{\Lambda_c\Delta t}V^{-1} \tag{4.31}$$

with Λ_c diagonal. The matrix exponential operation becomes a simple scalar computation,

$$e^{\Lambda_c\Delta t} = \text{diag}(e^{\lambda_1\Delta t}, \ldots, e^{\lambda_N\Delta t}) \tag{4.32}$$

In fact, using the eigen-decomposition and applying the ordinary differential equation approach, we have that

$$\dot{x}(t) = A_c x(t)$$

and therefore the solution is given in terms of the eigenvectors, v_i

$$x(t) = \sum_{i=0}^{N} \alpha_i\, e^{\lambda_i\Delta t} v_i \tag{4.33}$$

where the coefficients, α_i are the solution of the linear equations, $V \times \alpha = x(0)$. This approach works well when A_c is symmetric leading to an orthogonal set of eigenvectors, but can be plagued with a wealth of numerical issues that need reconciliation and sophisticated numerical techniques [18].

Consider the following example to demonstrate these approaches.

Example 4.1

Suppose we are given the following system:

$$\begin{aligned} \dot{x}_t &= -0.303x_t + u_t \\ y_t &= 2x_t \end{aligned}$$

with sampling interval, $\Delta t = 0.1$, initial state, $x_{t_0} = 2$ and input u_t a sequence of irregularly-spaced step functions. Using a first-order approximation, we obtain

$$x(t_k) = (1 - 0.303\Delta t_k)x(t_{k-1}) + \Delta t_k u(t_{k-1}) = 0.97x(t_{k-1}) + 0.1u(t_{k-1})$$
$$y(t_k) = 2x(t_k)$$

We performed numerical integration on the differential equations and a Taylor series approximation using 25-terms to achieve an error tolerance of $\epsilon = 10^{-12}$. The simulations of the state, output and input are shown in Fig. 4.1a–c. The true

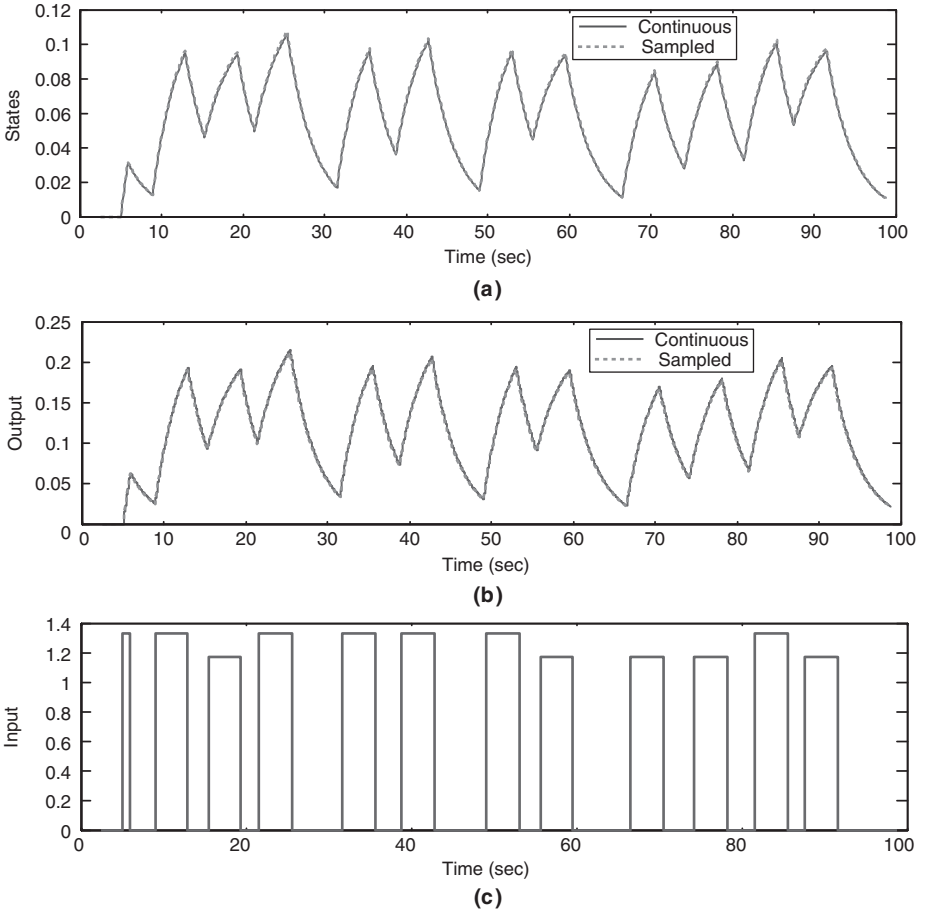


FIGURE 4.1 Sampled-data simulation (unequally spaced) of first order continuous process: (a) States: continuous (numerical integration) and sampled (series approximation). (b) Outputs: continuous (numerical integration) and sampled (series approximation). (c) Input: continuous (numerical integration).

continuous-time solution is shown in the figure as the solid line, while the sampled-data (discrete) solution is shown with the dotted lines. The plots reasonably overlay each another, but it is apparent that there is some truncation error evolving which can be observed at the peak amplitudes of the simulation. Thus we see that the continuous-time system in some cases can be reasonably approximated even by the Taylor-series approach. △△△

This completes the discussion of sampled-data systems and approximations, next we consider the discrete-time systems.

4.4 DISCRETE-TIME STATE–SPACE MODELS

Discrete state–space 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 given the past balance and the amount of the previous check. There is "no information" between time samples and so this model respresents 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 as discussed in the previous section. So we see that discrete-time dynamical systems can evolve from a wide variety of problems both naturally (checkbook) or physically (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.

We can use a first-difference approximation[2] and apply it to the general *LTI* continuous-time state–space model to obtain a discrete-time system, that is,

$$\begin{aligned} \dot{x}_t &\approx \frac{x(t) - x(t-1)}{\Delta T} \approx A_c x(t-1) + B_c u(t-1) \\ y_t &\approx y(t) = C_c x(t) + D_c u(t) \end{aligned}$$

Solving for $x(t)$, we obtain

$$\begin{aligned} x(t) &= (I + A_c\,\Delta T)x(t-1) + B_c\,\Delta T u(t-1) \\ y(t) &= C_c x(t) + D_c u(t) \end{aligned} \tag{4.34}$$

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

[2] This approximation is equivalent to a first-order Taylor series approximation.

feedthrough matrices as

$$\begin{aligned} A &\approx I + A_c\,\Delta T + O(\Delta T^2) \\ B &\approx B_c\,\Delta T \\ C &\approx C_c \\ D &\approx D_c \end{aligned} \tag{4.35}$$

We define the *nonlinear discrete-time state–space representation* by its process or system model

$$x(t) = A(x(t-1), u(t-1)) = a[x(t-1)] + b[u(t-1)] \tag{4.36}$$

and corresponding measurement or output model by

$$y(t) = C(x(t), u(t)) = c[x(t)] + d[u(t)] \tag{4.37}$$

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 feedthrough functions: the N_x-dimensional system and input functions, $a[\cdot]$, $b[\cdot]$ and the N_y-dimensional output and feedthrough functions, $c[\cdot]$, $d[\cdot]$.

The discrete *linear time-varying state–space representation* is given by the *system* or *process model* as

$$x(t) = A(t-1)x(t-1) + B(t-1)u(t-1) \tag{4.38}$$

and the corresponding discrete *output* or *measurement model* as

$$y(t) = C(t)x(t) + D(t)u(t) \tag{4.39}$$

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 $(N_y \times N_u)$-feedthrough matrices.

The state–space representation for linear, time-invariant, discrete systems is characterized by constant system, input, output and feedthrough 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 the *LTI* system

$$\begin{aligned} x(t) &= Ax(t-1) + Bu(t-1) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{4.40}$$

The discrete system representation replaces the Laplace transform with the Z-transform defined by the transform pair:

$$X(z) := \sum_{t=0}^{\infty} x(t)z^{-t}$$

$$x(t) = \int_{-\infty}^{\infty} X(z)z^{-1}\,dz \tag{4.41}$$

Time-invariant state–space discrete systems can also be represented in *input–output* or *transfer function* form using the Z-transform to give

$$H(z) = C(zI - A)^{-1}B + D \tag{4.42}$$

Also taking inverse Z-transforms, we obtain the discrete impulse response matrix as

$$H(t,k) = CA^{t-k}B + D \quad \text{for } t \geq k \tag{4.43}$$

The solution to the state-difference equations can easily be derived by induction [3] or using the transfer function approach of the previous subsection. In any case it is given by the relations

$$x(t) = \Phi(t,k)x(k) + \sum_{i=k+1}^{t} \Phi(t,i)B(i)u(i) \quad \text{for } t > k \tag{4.44}$$

where $\Phi(t,k)$ is the *discrete-time state-transition* matrix. For time-varying systems, it can be shown (by induction) that the state-transition matrix[3] satisfies

$$\Phi(t,k) = A(t-1)\cdot A(t-2)\cdots A(k)$$

while for time-invariant systems the state-transition matrix is given by

$$\Phi(t,k) = A^{t-k} \quad \text{for } t > k$$

The discrete state transition matrix possesses properties analogous 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:

$$\Phi(t,k) = A(t-1)\Phi(t-1,k), \qquad \Phi(k,k) = I, \quad t \geq k+1 \tag{4.45}$$

[3] Recall that for a sampled-data system, the state-transition matrix is: $\Phi(t,t_k) = e^{A(t-t_k)}$ where A is the sampled-data system or process matrix.

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.

4.4.1 Discrete Systems Theory

In this subsection we investigate the discrete state–space model from a 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 [2] 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:

$$\underline{\text{State Model}}: \quad x(t) = A(t-1)x(t-1) + B(t-1)u(t-1)$$

with corresponding measurement system or output defined by

$$\underline{\text{Measurement Model}}: \quad y(t) = C(t)x(t)$$

Using this representation, the simplest example of an observable system is one in which each state is measured directly, therefore and 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 pseudo-inverse must be performed instead [1, 2]. 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, $\Phi(t,0)$, where recall that the state equation is [3]

$$x(t) = \Phi(t,0)x(0) + \sum_{k=0}^{t-1} \Phi(t,k)B(k)u(k)$$

Therefore, pre-multiplying by the measurement matrix, the output relations are

$$y(t) = C(t)\Phi(t,0)x(0) + \sum_{k=0}^{t-1} C(t)\Phi(t,k)B(k)u(k) \tag{4.46}$$

or rearranging we define

$$\tilde{y}(t) := y(t) - \sum_{k=0}^{t-1} C(t)\Phi(t,k)B(k)u(k) = C(t)\Phi(t,0)x(0) \tag{4.47}$$

The problem is to solve this resulting equation for the initial state; therefore, 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 [2, 4] that the following $N_x \times N_x$ *observability Gramian* has the identical null space, that is,

$$\mathcal{O}(0,t) := \sum_{k=0}^{t-1} \Phi'(k,0)C'(k)C(k)\Phi(k,0) \tag{4.48}$$

which is equivalent to determining that $\mathcal{O}(0,t)$ is nonsingular or rank N_x. Further assuming that the system is *LTI* leads to the $NN_y \times N_x$ *observability matrix* [4] given by

$$\mathcal{O}(N) := \begin{bmatrix} C \\ - - - \\ \vdots \\ - - - \\ CA^{N-1} \end{bmatrix} \tag{4.49}$$

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 $x(t), x(0) \in \mathcal{R}^{N_x}$ there exists an input sequence, $\{u(t)\}$, $t = 0, \ldots, N-1$ 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

$$\mathcal{C}(0,t) := \sum_{k=0}^{t-1} \Phi(0,k)B(k)B'(k)\Phi'(0,k) \tag{4.50}$$

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

$$\mathcal{C}(N) := [B|AB|\ldots|A^{N-1}B] \tag{4.51}$$

must satisfy the rank condition, $\rho[\mathcal{C}] = N_x$ to be completely controllable [4].

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,

$$H(z) = C(zI - A)^{-1}B = \sum_{k=1}^{\infty} H(k)z^{-k} \quad \text{for } H(k) = CA^{k-1}B \tag{4.52}$$

where $H(k)$ is the $N_y \times N_u$ unit impulse response matrix which may also be viewed as a Markov sequence with (A, B, C) defined as the Markov parameters.

The problem of determining the *internal description* (A, B, C) from the *external description* ($H(z)$ or $\{H(k)\}$) of Eq. 4.43 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 $(N \times N_yN_u) \times (N \times N_yN_u)$ *Hankel matrix* by

$$\mathcal{H}(N) := \begin{bmatrix} H(1) & H(2) & \cdots & H(N) \\ H(2) & H(3) & \cdots & H(N+1) \\ \vdots & \vdots & \cdots & \vdots \\ H(N) & H(N+1) & \cdots & H(2N) \end{bmatrix} \tag{4.53}$$

Suppose the dimension of the system is N_x. Then the Hankel matrix could 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) $N \to \infty$ 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:

$$\mathcal{H}(N) = \begin{bmatrix} CB & \cdots & CA^{N-1}B \\ \vdots & & \vdots \\ CA^{N-1}B & \cdots & CA^{2N-2}B \end{bmatrix} = \begin{bmatrix} C \\ \vdots \\ CA^{N-1} \end{bmatrix} [B| \cdots |A^{N-1}B] \tag{4.54}$$

or more simply

$$\mathcal{H}(N) = \mathcal{O}(N)\mathcal{C}(N) \tag{4.55}$$

From this factorization it follows that the $\rho[\mathcal{H}(N)] = \min[\rho(\mathcal{O}(N)), \rho(\mathcal{C}(N))] = N_x$. Therefore, we see that the properties of controllability and observability are carefully woven into that of minimality, and 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, $\Sigma = \{A, B, C\}$, from noisy measurement data. For instance, many of the classical *realization* techniques [19, 20] rely on the factorization of Eq. 4.55 to extract the system or process matrix, that is,

$$\mathcal{O}(N) \times A = \begin{bmatrix} C \\ \vdots \\ CA^{N-2} \\ - - - \\ CA^{N-1} \end{bmatrix} A = \begin{bmatrix} CA \\ \vdots \\ CA^{N-1} \\ - - - \\ CA^{N} \end{bmatrix} =: \mathcal{O}^{\uparrow} \tag{4.56}$$

Solving for A using the pseudo-inverse [18] gives

$$\hat{A} = \mathcal{O}^{\#}(N)\mathcal{O}^{\uparrow} \quad \text{where } \mathcal{O}^{\#}(N) := (\mathcal{O}'(N)\mathcal{O}(N))^{-1}\mathcal{O}'(N) \tag{4.57}$$

An efficient realization technique for a *scalar* (single input/single output) system can be obtained by performing a *singular value decomposition* (*SVD*) of the Hankel matrix constructed from the impulse response sequence, $H(k); k = 1, \ldots, N$, that is,

$$\mathcal{H}(N) = U \times \Lambda \times V \ \text{ with } \rho(\mathcal{H}(N)) = N_x \tag{4.58}$$

where $\Lambda = \text{diag}[\lambda_i]; i = 1, \ldots, N^*; N^* := N \times N_y N_u$, $\{\lambda_i\}$ is the set of singular values, and U and V the corresponding left and right singular vector matrices [18]. A variety of techniques can be used to estimate the *rank* of a matrix, perhaps the simplest (given the *SVD*) being the *best rank approximation* α (percentage) based on the ratio of singular values,

$$\alpha(\hat{N}_x) := \left(\frac{\sum_{i=1}^{\hat{N}_x} \lambda_i}{\sum_{i=1}^{N^*} \lambda_i} \right) \times 100 \tag{4.59}$$

Here $\alpha\,(\%)$ is the percentage of the original matrix (Hankel) approximated by choosing $\hat{N}_x$. A threshold, τ_x (%) can be selected to search over i for that particular $\hat{N}_x$ that "best" approximates $\mathcal{H}$ up to the threshold, that is,

$$\alpha_n(\hat{N}_x) \geq \tau_x(\%) \quad \text{for } n = 1, \ldots, \hat{N}_x \tag{4.60}$$

Once the rank is estimated, then

$$\hat{\mathcal{H}}(N) = U\Lambda V = U \begin{bmatrix} \overline{\Lambda} & | & 0 \\ - - - & | & - - - \\ 0 & | & 0 \end{bmatrix} V = \overline{U}\,\overline{\Lambda}\,\overline{V} \tag{4.61}$$

for $\overline{U} \in \mathcal{R}^{N_Y N_u \times N_x}$, $\overline{\Lambda} \in \mathcal{R}^{N_x \times N_x}$, $\overline{V} \in \mathcal{R}^{N_x \times N_Y N_u}$.

When the decomposition and rank ($\hat{N}_x$) are determined, the system triple (scalar), $\Sigma = (A, b, c)$ are identified by [21]

$$A = \overline{\Lambda}^{-\frac{1}{2}}\,\overline{U}'\,\overline{U}^{\uparrow}\,\overline{\Lambda}^{\frac{1}{2}}; \quad b = \overline{\Lambda}^{\frac{1}{2}}\,\overline{V}; \quad \text{and} \quad c = \overline{U}\,\overline{\Lambda}^{\frac{1}{2}} \tag{4.62}$$

where (as before) $\overline{U}^{\uparrow}$ is the eigenvector matrix, $\overline{U}$ shifted up one row with a row of zeros $(0')$ appended [21]. Here $A^{1/2}(A^{1/2})'$ are the square-root matrices of A. It has been shown that this method results in a stable realization such that the eigenvalues of A, $\lambda(A \leq 1)$.

Example 4.2

Consider the following scalar example with impulse response, $H(k) = \{1,\ 1/2,\ 1/4,\ 1/8\ \ 1/16\}$ with $N = 5$. Using the *SVD* approach, we would like to extract the realization $\Sigma = (A, b, c)$. Creating the Hankel matrix and performing the *SVD*, we obtain

$$H(5) = \begin{bmatrix} 1 & 1/2 & 1/4 \\ 1/2 & 1/4 & 1/8 \\ 1/4 & 1/8 & 1/16 \end{bmatrix} = U\Lambda V$$

$\Lambda = \text{diag}[1.31,\ 0,\ 0]$ yielding a rank, $\hat{N}_x = 1$; the singular vectors are

$$U = \begin{bmatrix} -0.8729 & -0.4364 & -0.2182 \\ -0.4364 & 0.8983 & -0.0509 \\ -0.2182 & -0.0509 & 0.9746 \end{bmatrix}; \quad V = \begin{bmatrix} -0.8729 & -0.4364 & -0.2182 \\ 0.4880 & -0.7807 & -0.3904 \\ 0 & -0.4472 & 0.8944 \end{bmatrix}$$

Thus the best rank approximants are: $\overline{\Lambda} = 1.3125$ and

$$\overline{U} = \begin{bmatrix} -0.8729 \\ -0.4364 \\ -0.2182 \end{bmatrix}; \quad \overline{U}^{\uparrow} = \begin{bmatrix} -0.4364 \\ -0.2182 \\ 0 \end{bmatrix}; \quad \overline{V}' = \begin{bmatrix} -0.8729 \\ -0.4364 \\ -0.2182 \end{bmatrix}$$

Therefore, we obtain the realizations:

$$
\begin{aligned}
A &= \overline{\Lambda}^{-\frac{1}{2}}\, \overline{U}'\, \overline{U}^{\uparrow}\, \overline{\Lambda}^{\frac{1}{2}} \\
&= (0.873)[-0.8729 \;\; -0.4364 \;\; -0.2182] \begin{bmatrix} -0.4364 \\ -0.2182 \\ 0 \end{bmatrix} (1.1456) \\
&= 0.48 \approx 1/2;\; b' = \overline{\Lambda}^{\frac{1}{2}}\, \overline{V} = [1 \;\; 0 \;\; 0];\;\; c = \overline{U}\, \overline{\Lambda}^{\frac{1}{2}} = [-1 \;\; 0 \;\; 0] \qquad \triangle\triangle\triangle
\end{aligned}
$$

This completes the subsection on discrete systems theory. It should be noted that all of the properties discussed in this section exist for continuous-time systems (see [2] for details).

4.5 GAUSS-MARKOV STATE–SPACE MODELS

In this section we extend the state–space representation to incorporate random inputs or noise sources along with random initial conditions. We briefly discuss the continuous-time representation evolving to the sampled-data model and then provide a detailed discussion of the discrete-time Gauss-Markov model which will be used extensively throughout this text.

4.5.1 Continuous-Time/Sampled-Data Gauss-Markov Models

We start by defining the *continuous-time Gauss-Markov model*. If we constrain the state–space representation to be linear in the states, then we obtain the generic continuous-time, *linear time-varying Gauss-Markov state–space* model given by

$$
\begin{aligned}
\dot{x}_t &= A_t x_t + B_t u_t + W_t w_t \\
y_t &= C_t x_t + v_t
\end{aligned} \tag{4.63}
$$

where $x_t \in \mathcal{R}^{N_x \times 1}$, $u_t \in \mathcal{R}^{N_u \times 1}$, $w_t \in \mathcal{R}^{N_w \times 1}$, $y_t \in \mathcal{R}^{N_y \times 1}$, $v_t \in \mathcal{R}^{N_y \times 1}$ are the continuous-time state, input (deterministic), process noise, measurement and measurement noise[4] vectors with corresponding system (process), input, output and feedthrough matrices: $A \in \mathcal{R}^{N_x \times N_x}$, $B \in \mathcal{R}^{N_x \times N_u}$, $W \in \mathcal{R}^{N_x \times N_w}$ and $C \in \mathcal{R}^{N_y \times N_x}$.

The continuous-time Gaussian stochastic processes are $w_t \sim \mathcal{N}(0, R_{w_c w_c}(t))$ and $v_t \sim \mathcal{N}(0, R_{v_c v_c}(t))$ with initial state defined by $x_0 \sim \mathcal{N}(\overline{x}_0, \overline{P}_0)$. The corresponding

[4] Note that process and measurement noise sources are different. The process noise term is used primarily to model uncertainties in the input, state or even possibly unknown model parameters and it is "filtered" or colored (correlated) by the system dynamics, while measurement noise is uncorrelated and used to model instrumentation or extraneous environmental noise.

statistics of this model follow with the dynamic *mean* derived directly from Eq. 4.63 (see [6–8] for details) as

$$\dot{m}_{x_t} = A_t m_{x_t} + B_t u_t \tag{4.64}$$

and the *variance* (continuous-time Lyapunov equation)

$$\dot{P}_t = A_t P_t + P_t A_t' + W_t R_{w_c w_c}(t) W_t' \tag{4.65}$$

with corresponding *covariance*

$$P_{t,\tau} = \begin{cases} \Phi_c(t,\tau) P_{\tau,\tau} & \text{for } t \geq \tau \\ P_{t,t} \Phi_c'(t,\tau) & \text{for } t \leq \tau \end{cases} \tag{4.66}$$

where $\Phi_c(t,\tau)$ is the continuous state transition matrix.

As before, for the deterministic case, the stochastic sampled-data system follows from the continuous-time state solution

$$x_t = \Phi_c(t,t_0)x_{t_0} + \int_{t_0}^{t} \Phi_c(t,\tau) B_\tau u_\tau \, d\tau + \int_{t_0}^{t} \Phi_c(t,\tau) W_\tau w_\tau \, d\tau \tag{4.67}$$

Sampling this system with $t \to t_k$ over the interval $(t_k, t_{k-1}]$, then the sampled solution becomes (with notational change)[5]

$$x(t_k) = \Phi(t_k,t_{k-1})x(t_{k-1}) + \int_{t_{k-1}}^{t_k} \Phi(t_k,\tau) B_\tau u_\tau \, d\tau + \int_{t_{k-1}}^{t_k} \Phi(t_k,\tau) W_\tau w_\tau \, d\tau \tag{4.68}$$

If we further assume that the input excitation is *piecewise constant* ($u_\tau \to u(t_{k-1})$) over the interval $(t_k, t_{k-1}]$, then it can be removed from under the superposition integral in Eq. 4.68 to give

$$x(t_k) = \Phi(t_k,t_{k-1})x(t_{k-1}) + \left(\int_{t_{k-1}}^{t_k} \Phi(t_k,\tau) B_\tau \, d\tau \right) \times u(t_{k-1}) + \int_{t_{k-1}}^{t_k} \Phi(t_k,\tau) W_\tau w_\tau \, d\tau \tag{4.69}$$

We define the *sampled-data input transmission matrix* as

$$B(t_{k-1}) := \int_{t_{k-1}}^{t_k} \Phi(t_k,\tau) B_\tau \, d\tau \tag{4.70}$$

with the *sampled-data process noise covariance* matrix is given by

$$R_{ww}(t_{k-1}) := \int_{t_{k-1}}^{t_k} \Phi(t_k,\tau) W_\tau R_{w_c w_c}(t) W_\tau' \Phi'(t_k,\tau) \, d\tau \tag{4.71}$$

[5] We note in passing that this solution is conceptual and must actually follow a much more rigorous framework embedded in stochastic integrals and beyond the scope of this text (see [6–8] for details).

and therefore the *sampled-data state–space system* with equally or unequally sampled-data is given by:

$$\begin{aligned} x(t_k) &= \Phi(t_k, t_{k-1})x(t_{k-1}) + B(t_{k-1})u(t_{k-1}) + W(t_{k-1})w(t_{k-1}) \\ y(t_k) &= C(t_k)x(t_k) + v(t_k) \end{aligned} \tag{4.72}$$

for $w_{t_k} \sim \mathcal{N}(0, R_{ww}(t_k))$ and $v_{t_k} \sim \mathcal{N}(0, R_{vv}(t_k))$ with initial state defined by $x(t_0) \sim \mathcal{N}(\overline{x}(t_0), \overline{P}(t_0))$. Recall from the deterministic solution that if we use a first order Taylor series approximation we have that:

$$\begin{aligned} \Phi(t_k, t_{k-1}) &\approx I + \Delta t_k \times A(t_{k-1}) \\ B(t_{k-1}) &\approx \Delta t_k \times B_{t_{k-1}} \\ R_{ww}(t_{k-1}) &\approx \Delta t_k \times W_{t_{k-1}} R_{w_c w_c}(t_{k-1}) W'_{t_{k-1}} \\ R_{vv}(t_k) &\approx R_{v_c v_c}(t_k)/\Delta t_k \end{aligned} \tag{4.73}$$

The corresponding mean and covariance of the sampled-data process are:

$$m_x(t_k) = A(t_{k-1})m_x(t_{k-1}) + B(t_{k-1})u(t_{k-1}) \tag{4.74}$$

and the *measurement mean vector* m_y as

$$m_y(t_k) = C(t_k)m_x(t_k) \tag{4.75}$$

The *state variance* is given by

$$P(t_k) = A(t_{k-1})P(t_{k-1})A'(t_{k-1}) + W(t_{k-1})R_{ww}(t_{k-1})W'(t_{k-1}) \tag{4.76}$$

and the measurement variance is

$$R_{yy}(t_k) = C(t_k)P(t_k)C'(t_k) + R_{vv}(t_k) \tag{4.77}$$

This completes the development of the sampled-data Gauss-Markov representation evolving from a continuous-time stochastic process, next we consider the more pragmatic discrete-time model.

4.5.2 Discrete-Time Gauss-Markov 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 Gaussian $w(t-1)$, the *Gauss-Markov model* evolves as

$$x(t) = A(t-1)x(t-1) + B(t-1)u(t-1) + W(t-1)w(t-1) \tag{4.78}$$

where $w \sim \mathcal{N}(0, R_{ww}(t-1))$ and $x(0) \sim \mathcal{N}(\bar{x}(0), P(0))$.

The solution to the Gauss-Markov equations can easily be obtained by induction to give

$$x(t) = \Phi(t,k)x(k) + \sum_{i=k}^{t-1} \Phi(t, i+1)B(i)u(i) + \sum_{i=k}^{t-1} \Phi(t, i+1)W(i)w(i) \tag{4.79}$$

which is first-order Markov depending only on the previous state. Since $x(t)$ is just 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

$$y(t) = C(t)x(t) + v(t) \tag{4.80}$$

where $v \sim \mathcal{N}(0, R_{vv}(t))$. The model is shown diagrammatically in Fig. 4.2.

Since the Gauss-Markov model of Eq. 4.78 is characterized by a Gaussian distribution, it is completely specified statistically by its mean and variance. Therefore, if we take the expectation of Eqs. 4.78 and 4.80, respectively, we obtain the *state mean vector* m_x as

$$m_x(t) = A(t-1)m_x(t-1) + B(t-1)u(t-1) \tag{4.81}$$

and the *measurement mean vector* m_y as

$$m_y(t) = C(t)m_x(t) \tag{4.82}$$

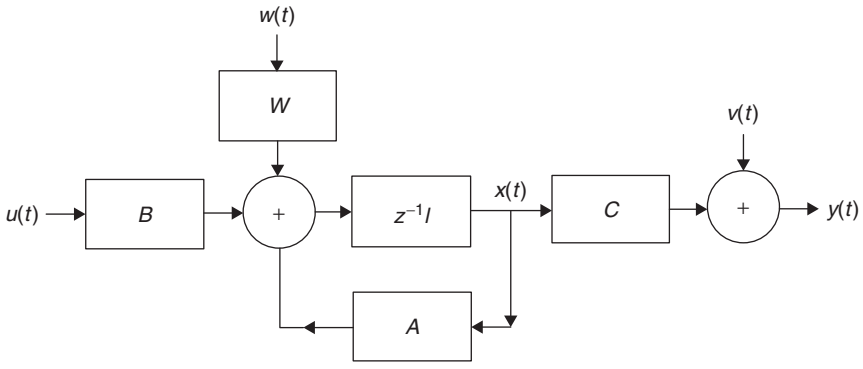


FIGURE 4.2 Gauss-Markov model of a discrete process.

The *state variance*[6] $P(t) := \text{var}\{x(t)\}$ is given by the discrete Lyapunov equation:

$$P(t) = A(t-1)P(t-1)A'(t-1) + W(t-1)R_{ww}(t-1)W'(t-1) \tag{4.83}$$

and the measurement variance, $R_{yy}(t) := \text{var}\{y(t)\}$ is

$$R_{yy}(t) = C(t)P(t)C'(t) + R_{vv}(t) \tag{4.84}$$

Similarly, it can be shown that the *state covariance* propagates according to the following equations:

$$P(t,k) = \begin{cases} \Phi(t,k)P(k) & \text{for } t \geq k \\ P(t)\Phi'(t,k) & \text{for } t \leq k \end{cases} \tag{4.85}$$

We summarize the Gauss-Markov and corresponding statistical models in Table 4.1.

TABLE 4.1 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

$$R_{yy}(t) = C(t)P(t)C'(t) + R_{vv}(t)$$

$$R_{yy}(t,k) = C(t)P(t)C'(t) + R_{vv}(t,k)$$

[6] We use the shorthand notation, $P(k) := P_{xx}(k,k) = \text{cov}\{x(k), x(k)\} = \text{var}\{x(k)\}$, throughout this text.

If we restrict the Gauss-Markov model to the stationary case, then

$$A(t) = A, B(t) = B, \quad C(t) = C, \quad W(t) = W, \quad R_{ww}(t) = R_{ww}, \quad \text{and} \quad R_{vv}(t) = R_{vv}$$

and the variance equations become

$$P(t) = AP(t-1)A' + WR_{ww}W'$$

and

$$R_{yy}(t) = CP(t)C' + R_{vv} \tag{4.86}$$

At steady-state ($t \to \infty$), we have

$$P(t) = P(t-1) = \cdots = P_{ss} := P$$

and therefore, the measurement covariance relations become

$$R_{yy}(0) = CPC' + R_{vv} \quad \text{for lag } k = 0 \tag{4.87}$$

By induction, it can be shown that

$$R_{yy}(k) = CA^{|k|}PC' \quad \text{for } k \neq 0 \tag{4.88}$$

The measurement power spectrum is easily obtained by taking the Z-transform of this equation to obtain

$$S_{yy}(z) = CS_{xx}(z)C' + S_{vv}(z) \tag{4.89}$$

where

$$S_{xx}(z) = T(z)S_{ww}(z)T'(z^{-1}) \quad \text{for } T(z) = (zI - A)^{-1}W$$

with

$$S_{ww}(z) = R_{ww} \quad \text{and} \quad S_{vv}(z) = R_{vv}$$

Thus, using $H(z) = CT(z)$, the spectrum is given by

$$S_{yy}(z) = H(z)R_{ww}H'(z^{-1}) + R_{vv} \tag{4.90}$$

So we see that the Gauss-Markov state–space model enables us to have a more general representation of a multichannel stochastic signal. In fact, we are able to easily handle the multichannel and nonstationary statistical cases within this framework. Generalizations are also possible with the vector models, but the forms become quite complicated and require some knowledge of multivariable systems theory and canonical forms (see [1] for details). Before we leave this subject, let us consider a simple input-output example with Gauss-Markov models.

Example 4.3

Consider the following difference equation driven by random (white) noise:

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

The corresponding state–space representation is obtained as

$$x(t) = -ax(t-1) + w(t-1) \quad \text{and} \quad y(t) = x(t)$$

Taking Z-transforms (ignoring the randomness), we obtain the transfer function

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

Using Eq. 4.83, the variance equation for the above model is

$$P(t) = a^2 P(t-1) + R_{ww}$$

Assume the process is stationary, then $P(t) = P$ for all t and solving for P it follows that

$$P = \frac{R_{ww}}{1 - a^2}$$

Therefore,

$$R_{yy}(k) = CA^{|k|}PC' = \frac{a^{|k|}R_{ww}}{1 - a^2} \quad \text{and} \quad R_{yy}(0) = CPC' + R_{vv} = \frac{R_{ww}}{1 - a^2}$$

Choosing $R_{ww} = 1 - a^2$ gives $R_{yy}(k) = a^{|k|}$. Taking Z-transforms the discrete power spectrum is given by

$$S_{yy}(z) = H(z)R_{ee}H'(z^{-1}) + R_{vv} = \frac{1}{1 - az^{-1}} R_{ww} \frac{1}{1 - az}$$

Therefore, we conclude that for stationary processes these models are equivalent.

Now if we assume a nonstationary process and 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

$$x(t) = 0.75x(t-1) + w(t-1) \quad \text{and} \quad y(t) = x(t) + v(t)$$

The corresponding statistics are given by the mean relations

$$\begin{aligned} m_x(t) &= 0.75m_x(t-1) & m_x(0) &= 1 \\ m_y(t) &= m_x(t) & m_y(0) &= m_x(0) \end{aligned}$$

and the variance equations

$$P(t) = 0.5625P(t-1) + 1 \quad \text{and} \quad R_{yy}(t) = P(t) + 4$$

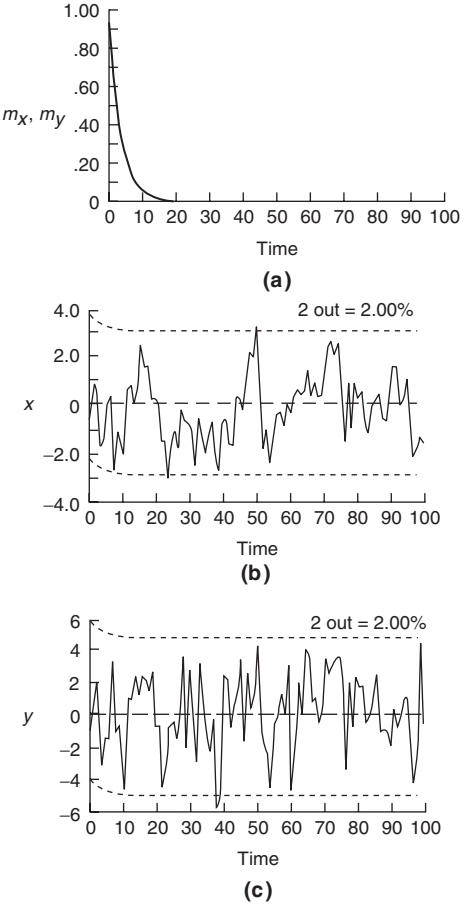


FIGURE 4.3 Gauss-Markov simulation of first-order process.

We apply the simulator available in *SSPACK_PC* [13] to obtain a 100-sample realization of the process. The results are shown in Fig. 4.3a through c. In a and 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

$$P = \frac{R_{ww}}{1 - a^2} = 2.286$$

Using the above confidence interval, we expect 95% of the samples to lie within $(m_x \to 0) \pm 3.03(1.96 \times \sqrt{2.286})$. From the figure we see that only 2 of the 100 samples exceed this 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 $(m_y \to 0)$ $\pm 5.01(1.96 \times \sqrt{6.286})$ at steady-state. This completes the example. △△△

4.6 INNOVATIONS MODEL

In this subsection we briefly develop the innovations model which 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 showing 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

$$\begin{aligned} x(t) &= Ax(t-1) + Bu(t-1) + W(t-1)w^*(t-1) \\ y(t) &= Cx(t) + v^*(t) \end{aligned} \tag{4.91}$$

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

$$R^* = \begin{bmatrix} R_1^* \\ R_2^* \end{bmatrix} [R_1^{*\prime} \quad R_2^{*\prime}] \tag{4.92}$$

where R_i^* are matrix square roots [6, 7]. Once the factorization is performed, then the correlated noise is synthesized "coloring" the uncorrelated noise sources, $w(t)$ and $v(t)$ as

$$\begin{bmatrix} w^*(t) \\ v^*(t) \end{bmatrix} = \begin{bmatrix} R_1^{*\prime} w(t) \\ R_2^{*\prime} v(t) \end{bmatrix} \tag{4.93}$$

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 the *innovations model* [9–13] evolves as:

$$\begin{aligned} x(t) &= A(t-1)x(t-1) + B(t-1)u(t-1) + K(t-1)e(t-1) \\ y(t) &= C(t)x(t) + D(t)u(t) + e(t) \end{aligned} \tag{4.94}$$

where $e(t)$ is the N_y-dimensional innovations vector and $K(t-1)$ is the $(N_x \times N_y)$ weighting matrix.[7]

$$R_{ee}^{*} := \text{cov}\left(\begin{bmatrix} Ke(t) \\ e(t) \end{bmatrix}\right) = \left[\begin{array}{c|c} KR_{ee}K' & KR_{ee} \\ \hline R_{ee}K' & R_{ee} \end{array}\right] \delta(t-k)$$

It is important to note that the innovations model has implications in Wiener-Kalman filtering (spectral factorization) because R_{ee} 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,

$$R_{ee}^{*} := \begin{bmatrix} K\sqrt{R_{ee}} \\ \sqrt{R_{ee}} \end{bmatrix} \begin{bmatrix} \sqrt{R_{ee}}K' & \sqrt{R_{ee}} \end{bmatrix} \delta(t-k) \tag{4.95}$$

Comparing the innovations model to the Gauss-Markov model, we see that they are both equivalent to the case when 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.

4.7 STATE–SPACE MODEL STRUCTURES

In this section we discuss special state–space structures usually called "canonical forms" in the literature, since they respresent unique state constructs that are particularly useful. We will confine the models to single input/single output forms because the multivariable structures are too complicated for this discussion [4]. Here we will first investigate the most popular "time series" models and then their equivalent representation in the state–space. We start with the *ARMAX* model and then progress to the special cases of this generic structure.

4.7.1 Time Series Models

Time series models are particularly useful representations used frequently by statisticians and signal processors to represent time sequences when no physics is available

[7] 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.

to use directly. They form the class of black box or gray box models [13] which are used in predicting data. These models have an input–output structure, but they can be transformed to an equivalent state–space representation. Each model set has its own advantages: the input-output models are easy to use, while the state–space models are easily generalized and usually evolve when physical phenomenology can be described in a model-based sense [13].

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 expressed as

$$H(z) = \frac{B(z^{-1})}{A(z^{-1})} \tag{4.96}$$

where A and B are polynomials in z or z^{-1},

$$A(z^{-1}) = 1 + a_1 z^{-1} + \cdots + a_{N_a} z^{-N_a} \tag{4.97}$$

$$B(z^{-1}) = b_0 + b_1 z^{-1} + \cdots + b_{N_b} z^{-N_b} \tag{4.98}$$

If we consider the equivalent time domain representation, then we have a *difference equation* relating the output sequence $\{y(t)\}$ to the input sequence $\{u(t)\}$.[8] We use the *backward shift operator* q with the property that $q^{-k}y(t) = y(t-k)$.

$$A(q^{-1})y(t) = B(q^{-1})u(t) \tag{4.99}$$

or,

$$y(t) + a_1 y(t-1) + \cdots + a_{N_a} y(t-N_a) = b_0 u(t) + \cdots b_{N_b} u(t-N_b) \tag{4.100}$$

When the system is excited by random inputs, the models are given by the *autoregressive-moving average model with exogenous inputs* (*ARMAX*)[9]

$$\underbrace{A(q^{-1})y(t)}_{AR} = \underbrace{B(q^{-1})u(t)}_{X} + \underbrace{C(q^{-1})e(t)}_{MA} \tag{4.101}$$

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

$$C(q^{-1}) = c_0 + c_1 q^{-1} + \cdots + c_{N_c} q^{-N_c}$$

[8] 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.

[9] 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 ($e(t)$) [10].

The *ARMAX* model usually abbreviated by $ARMAX(N_a, N_b, N_c)$ represents the general form for many popular time series and digital filter models. A summary of these models follows:

- 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 Exogenous 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 Fig. 4.4. *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.

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

$$A(q^{-1})m_y(t) = B(q^{-1})u(t) + C(q^{-1})m_e(t) \tag{4.102}$$

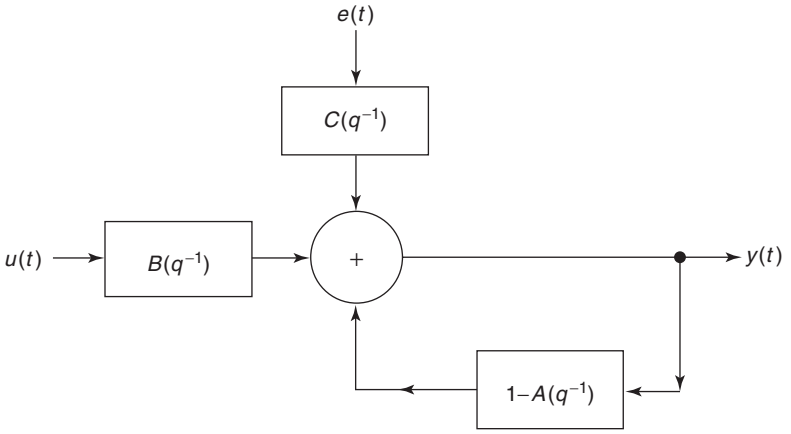


FIGURE 4.4 *ARMAX* input–output model.

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

$$m_y(t) = (1 - A(q^{-1}))m_y(t) + B(q^{-1})u(t) + C(q^{-1})m_e(t) \tag{4.103}$$

$$m_y(t) = -\sum_{i=1}^{N_a} a_i m_y(t-i) + \sum_{i=0}^{N_b} b_i u(t-i) + \sum_{i=0}^{N_c} c_i m_e(t-i) \tag{4.104}$$

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,

$$\begin{aligned} y(t) - m_y(t) = {} & [(1 - A(q^{-1}))y(t) + B(q^{-1})u(t) + C(q^{-1})e(t)] \\ & - [(1 - A(q^{-1}))m_y(t) + B(q^{-1})u(t) + C(q^{-1})m_e(t)] \end{aligned} \tag{4.105}$$

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

$$A(q^{-1})(y(t) - m_y(t)) = C(q^{-1})(e(t) - m_e(t)) \tag{4.106}$$

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 an *IIR* system, that is,

$$\frac{Y(z)}{E(z)} = H(z) = \sum_{t=0}^{\infty} h(t) z^{-t} \tag{4.107}$$

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

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

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

$$R_{yy}(k) = R_{ee} \sum_{i=0}^{\infty} h(t)h(t+k) \quad \text{for } k \geq 0 \tag{4.108}$$

with corresponding *variance*

$$R_{yy}(0) = R_{ee} \sum_{t=0}^{\infty} h^2(t) \tag{4.109}$$

We note that one property of a stationary signal is that its impulse response is bounded which implies from Eq. 4.108 that the variance is bounded [14]. Clearly, since the variance is characterized by an *ARMA* model ($ARMAX(N_a, 0, N_c)$), then we have

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

TABLE 4.2 *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

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

y = output or measurement sequence
u = input sequence
e = process (white) noise sequence with variance R_{ee}
h = impulse response sequence
δ = 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

or taking the inverse Z-transform

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

or

$$h(t) = -\sum_{i=1}^{N_a} a_i h(t-i) + \sum_{i=0}^{N_c} c_i \delta(t), \quad c_0 = 1 \tag{4.110}$$

where $\delta(t)$ is an impulse of weight $\sqrt{R_{ee}}$. So we see that this recursion coupled with Eq. 4.108 provides a method for calculating the variance of an *ARMAX* model. We summarize these results in Table 4.2 and the following example.

Example 4.4

Consider the difference equation of the previous example with $a = 0.5$ which is an AR model with $A(z) = 1 + 0.5z^{-1}$, and $R_{ee} = 1$. We would like to calculate the variance. From Eq. 4.110, we have

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

and

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

△△△

Let us consider a more complex example to illustrate the use of the *ARMA* model.

Example 4.5

Suppose we would like to investigate the structure of an *ARMAX*(2, 1, 1) model with the following difference equation and calculate its underlying mean and variance propagation relations

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

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

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

for $m_e(t) = 1$ for all t. The impulse propagation model is

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

and the variance/covariance propagation model is:

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

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*

$$m_y(t) = E\{C(q^{-1})e(t)\} = C(q^{-1})m_e(t) \tag{4.111}$$

and *covariance* (directly from Eq. 4.108 with $h \rightarrow c$)

$$R_{yy}(k) = R_{ee} \sum_{i=0}^{N_c} c_i c_{i+k} \quad \text{for } k \geq 0 \tag{4.112}$$

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

$$m_y(t) = (1 - A(q^{-1}))m_y(t) + m_e(t) \tag{4.113}$$

and *covariance* which is easily derived by direct substitution

$$R_{yy}(k) = E\{y(t)y(t+k)\} = (1 - A(q^{-1}))R_{yy}(k) = -\sum_{i=1}^{N_a} a_i R_{yy}(k-i) \quad \text{for } k > 0 \tag{4.114}$$

In fact, the *AR* covariance model of Eq. 4.114 is essentially a recursive (all-pole) digital filter which can be propagated by exciting it with the variance $R_{yy}(0)$ as initial condition. In this case the variance is given by

$$R_{yy}(0) = E\{y^2(t)\} = E\left\{\left(-\sum_{i=1}^{N_a} a_i y(t-i) + e(t)\right) y(t)\right\} = -\sum_{i=1}^{N_a} a_i R_{yy}(i) + R_{ee} \tag{4.115}$$

So combining Eqs. (4.114, 4.115), we have that the *covariance propagation* equations for the *AR* model are given by

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

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

Example 4.6

Consider the *AR* model of the previous two examples. We would like to determine the corresponding mean and variance using the recursions of Eqs. (4.113, 4.114) 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

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

and therefore

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

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

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

or

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

as before. △△△

This completes the section on *ARMAX* models.

4.7.2 State–Space and Time Series Equivalence Models

In this section we show the equivalence between the *ARMAX* and state–space models (for scalar processes). That is, we show how to obtain the state–space model given the *ARMAX* 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. 4.101 that the general difference equation form of the *ARMAX* model is given by

$$y(t) = -\sum_{i=1}^{N_a} a_i y(t-i) + \sum_{i=0}^{N_b} b_i u(t-i) + \sum_{i=0}^{N_c} c_i e(t-i) \tag{4.116}$$

or equivalently in the frequency domain as

$$Y(z) = \left(\frac{b_o + b_1 z^{-1} + \cdots + b_{N_b} z^{-N_b}}{1 + a_1 z^{-1} + \cdots + a_{N_a} z^{-N_a}}\right) U(z) + \left(\frac{c_o + c_1 z^{-1} + \cdots + c_{N_c} z^{-N_c}}{1 + a_1 z^{-1} + \cdots + a_{N_a} z^{-N_a}}\right) E(z) \tag{4.117}$$

where $N_a \geq N_b$ and N_c and $\{e(t)\}$ is a zero-mean white sequence with spectrum given by R_{ee}.

It is straightforward but tedious to show (see [5]) that the *ARMAX* model can be represented in *observer canonical form*:

$$\begin{aligned} x(t) &= A_0 x(t-1) + B_0 u(t-1) + W_0 e(t-1) \\ y(t) &= C_0' x(t) + b_0 u(t) + c_0 e(t) \end{aligned} \tag{4.118}$$

where x, u, e, and y are the N_a-state vector, scalar input, noise, and output with

$$A_0 := \left[\begin{array}{c|c} 0 & -a_{N_a} \\ \hline I_{N_a-1} & \begin{array}{c} \vdots \\ -a_1 \end{array} \end{array}\right] \quad B_0 := \begin{bmatrix} -a_{N_a} b_0 \\ \vdots \\ -a_{N_b+1} b_0 \\ \hline b_{N_b} - a_{N_b} b_0 \\ \vdots \\ b_1 - a_1 b_0 \end{bmatrix} \quad W_0 := \begin{bmatrix} -a_{N_a} c_0 \\ \vdots \\ -a_{N_c+1} c_0 \\ \hline c_{N_c} - a_{N_c} c_0 \\ \vdots \\ c_1 - a_1 c_0 \end{bmatrix}$$

$$C_0' := [0 \ \cdots \ 0 \ 1]$$

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

$$\begin{aligned} A_{i,N_a} &= -a_i \quad i = 1, \ldots, N_a \\ B_i &= b_i - a_i b_0 \\ W_i &= c_i - a_i c_0 \\ C_i &= \delta(N_a - i) \end{aligned} \tag{4.119}$$

where

$$\begin{aligned} b_i &= 0 \qquad \text{for } i > N_b \\ c_i &= 0 \qquad \text{for } i > N_c \\ \delta(i-j) &\quad \text{is the Kronecker delta} \end{aligned}$$

Consider the following example to illustrate these relations.

Example 4.7

Let $N_a = 3,\ N_b = 2$, and $N_c = 1$; then the corresponding *ARMAX* model is

$$\begin{aligned} y(t) = &-a_1 y(t-1) - a_2 y(t-2) - a_3 y(t-3) + b_0 u(t) \\ &+ b_1 u(t-1) + b_2 u(t-2) + c_0 e(t) + c_1 e(t-1) \end{aligned} \tag{4.120}$$

Using the observer canonical form of Eq. 4.118 we have

$$\begin{aligned} x(t) = &\left[\begin{array}{cc|c} 0 & 0 & -a_3 \\ \hline 1 & 0 & -a_2 \\ 0 & 1 & -a_1 \end{array}\right] x(t-1) + \begin{bmatrix} -a_3 b_0 \\ \hline b_2 - a_2 b_0 \\ b_1 - a_1 b_0 \end{bmatrix} u(t-1) \\ &+ \begin{bmatrix} -a_3 c_0 \\ -a_2 c_0 \\ \hline c_1 - a_1 c_0 \end{bmatrix} e(t-1) \end{aligned}$$

$$y(t) = [0 \;\; 0 \;\; 1]x(t) + b_0 u(t) + c_0 e(t)$$

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,

$$\begin{aligned} x(t) &= Ax(t-1) + Bu(t-1) + We(t-1) \\ y(t) &= C'x(t) + b_0 u(t) + c_0 e(t) \end{aligned} \tag{4.121}$$

where in this case, $K \to W, D \to b_0$, and $1 \to c_0$. Also, the corresponding covariance matrix becomes:

$$R^*_{ee} := \operatorname{cov}\left(\begin{bmatrix} We(t) \\ c_0 e(t) \end{bmatrix}\right) = \left[\begin{array}{c|c} WR_{ee}W' & WR_{ee}c_0 \\ \hline c_0 R_{ee}W' & c_0 R_{ee}c_0 \end{array}\right] \delta(t-k)$$

This completes the discussion on the equivalence of the general *ARMAX* to state–space.

Next let us develop the state–space equivalent model for some special cases of the *ARMAX* model presented in the previous subsection. We begin with the *moving average MA*

$$y(t) = \sum_{i=1}^{N_c} c_i e(t-i) \quad \text{or} \quad Y(z) = C(z)E(z) = (1 + c_1 z^{-1} + \cdots + c_{N_c} z^{-N_c})E(z)$$

Define the state variable as

$$x_i(t-1) := e(t-i-1), \quad i = 1, \ldots, N_c \tag{4.122}$$

and therefore,

$$x_i(t) = e(t-i) = x_{i-1}(t-1), \quad i = 1, \ldots, N_c \tag{4.123}$$

Expanding this expression, we obtain

$$\begin{aligned} x_1(t) &= e(t-1) \\ x_2(t) &= e(t-2) &&= x_1(t-1) \\ x_3(t) &= e(t-3) &&= x_2(t-1) \\ &\vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ x_{N_c}(t) &= e(t-N_c) &&= x_{N_c-1}(t-1) \end{aligned} \tag{4.124}$$

or in vector-matrix form

$$\begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_{N_c}(t) \end{bmatrix} = \left[\begin{array}{ccc|c} 0 & \cdots & 0 & 0 \\ \hline 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 \end{array}\right] \begin{bmatrix} x_1(t-1) \\ x_2(t-1) \\ \vdots \\ x_{N_c}(t-1) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} e(t-1)$$

$$y(t) = [c_1 \quad c_2 \cdots c_{N_c}] \begin{bmatrix} x_1(t-1) \\ x_2(t-1) \\ \vdots \\ x_{N_c}(t-1) \end{bmatrix} + c_0 e(t) \tag{4.125}$$

Thus the general form for the *moving average* (*MA*) state–space is given by

$$\mathbf{x}(t) = \left[\begin{array}{ccc|c} 0 & \cdots & 0 & 0 \\ \hline & & & \\ & \mathbf{I}_{N_c-1} & & \vdots \\ & & & \\ & & & 0 \end{array}\right] \mathbf{x}(t-1) + \mathbf{b}_1 e(t-1)$$

$$y(t) = \mathbf{c}'\mathbf{x}(t) + c_0 e(t) \tag{4.126}$$

with $N_x = N_c$, $\mathbf{b}_1, \mathbf{c} \in R^{N_x \times 1}$, $\mathbf{b}_1$ a unit vector.

Example 4.8

Suppose 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 $N_x = 2$, and therefore,

$$\begin{aligned} \mathbf{x}(t) &= \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \mathbf{x}(t-1) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} e(t-1) \\ y(t) &= [c_1 \quad c_2]\, \mathbf{x}(t) + c_0 e(t) \end{aligned}$$ △△△

Consider the *AR* model (all-pole) given by

$$\sum_{i=1}^{N_a} a_i y(t-i) = \sigma e(t) \quad \text{or} \quad Y(z) = \frac{\sigma E(z)}{1 + a_1 z^{-1} + \cdots + a_{N_a} z^{-N_a}} \tag{4.127}$$

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

$$\begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_{N_a}(t) \end{bmatrix} = \left[\begin{array}{c|ccccc} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ \hline -a_{N_a} & -a_{N_a-1} & -a_{N_a-2} & \cdots & -a_1 \end{array}\right] \begin{bmatrix} x_1(t-1) \\ x_2(t-1) \\ \vdots \\ x_{N_a}(t-1) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \sigma \end{bmatrix} e(t-1)$$

$$y(t) = [0 \quad 0 \quad \cdots \quad 1] \begin{bmatrix} x_1(t-1) \\ x_2(t-1) \\ \vdots \\ x_{N_a}(t-1) \end{bmatrix} \tag{4.128}$$

In general, we have the *AR* (*all-pole*) state–space model

$$\mathbf{x}(t) = \left[\begin{array}{c|ccccc} 0 & & & & \\ \vdots & & \multicolumn{3}{c}{\mathbf{I}_{N_a-1}} & \\ 0 & & & & \\ \hline -a_{N_a} & -a_{N_a-1} & -a_{N_a-2} & \cdots & -a_1 \end{array}\right] \mathbf{x}(t-1) + \mathbf{b}e(t-1)$$

$$y(t) = \mathbf{c}'\mathbf{x}(t) \tag{4.129}$$

with $N_x = N_a$, $\mathbf{b}, \mathbf{c} \in R^{N_x \times 1}$. Consider the following example to demonstrate this form.

Example 4.9

Given the *AR* model with $N_a = 2$ and $\sigma = \sqrt{2}$, find the equivalent state–space form. We have

$$y(t) = -a_1 y(t-1) - a_2 y(t-2) + \sqrt{(2)}e(t)$$

and therefore,

$$\begin{aligned} x_1(t-1) &= y(t-2) \\ x_2(t-1) &= y(t-1) \end{aligned}$$

which gives

$$\begin{aligned} x_1(t) &= y(t-1) = x_2(t-1) \\ x_2(t) &= y(t) = -a_1 y(t-1) - a_2 y(t-2) = -a_1 x_2(t-1) - a_2 x_1(t-1) \end{aligned}$$

or more succinctly

$$\mathbf{x}(t) = \begin{bmatrix} 0 & 1 \\ -a_2 & -a_1 \end{bmatrix} \mathbf{x}(t-1) + \begin{bmatrix} 0 \\ \sqrt{2} \end{bmatrix} e(t-1)$$

$$y(t) = [0 \quad 1]\mathbf{x}(t)$$

△△△

Another useful state–space representation is the *normal form* which evolves by performing a partial fraction expansion of a rational discrete transfer function model (*ARMA*) to obtain

$$h(t) = \sum_{i=1}^{N_p} R_i (p_i)^{-t} \quad \text{or} \quad H(z^{-1}) = \frac{Y(z^{-1})}{E(z^{-1})} = \sum_{i=1}^{N_p} \frac{R_i}{1 - p_i z^{-1}} \tag{4.130}$$

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

$$y_i(t) - p_i y_i(t-1) = e(t), \quad i = 1, \ldots, N_p$$

Defining the state variable as $x_i(t) := y_i(t)$, then equivalently

$$x_i(t) - p_i x_i(t-1) = e(t), \quad i = 1, \ldots, N_p \tag{4.131}$$

and therefore, the output is given by

$$y(t) = \sum_{i=1}^{N_p} R_i y_i(t) = \sum_{i=1}^{N_p} R_i x_i(t), \quad i = 1, \ldots, N_p \tag{4.132}$$

Expanding these relations over i, we obtain

$$\begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_{N_p}(t) \end{bmatrix} = \begin{bmatrix} p_1 & 0 & \cdots & 0 \\ 0 & p_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & p_{N_p} \end{bmatrix} \begin{bmatrix} x_1(t-1) \\ x_2(t-1) \\ \vdots \\ x_{N_a}(t-1) \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} e(t-1)$$

$$y(t) = [R_1 \quad R_2 \quad \cdots \quad R_{N_p}] \begin{bmatrix} x_1(t-1) \\ x_2(t-1) \\ \vdots \\ x_{N_p}(t-1) \end{bmatrix} \tag{4.133}$$

Thus, the general decoupled form of the *normal state–space model* is given by

$$\mathbf{x}(t) = \begin{bmatrix} p_1 & 0 & \cdots & 0 \\ 0 & p_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & p_{N_p} \end{bmatrix} \mathbf{x}(t-1) + \mathbf{b}e(t-1)$$

$$y(t) = \mathbf{c}'\mathbf{x}(t) \tag{4.134}$$

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 $\mathbf{c}' = [R_1 \quad R_2 \quad \cdots \quad R_{N_p}]$.

Example 4.10

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

$$\begin{aligned} \mathbf{x}(t) &= \begin{bmatrix} p_1 & 0 & 0 \\ 0 & p_2 & 0 \\ 0 & 0 & p_3 \end{bmatrix} \mathbf{x}(t-1) + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} e(t-1) \\ y(t) &= [R_1 \quad R_2 \quad R_3]\, \mathbf{x}(t) \end{aligned}$$

△△△

4.8 NONLINEAR (APPROXIMATE) GAUSS-MARKOV STATE–SPACE MODELS

Many processes in practice are nonlinear rather than linear. Coupling the nonlinearities with noisy data makes the signal processing problem a challenging one. In this section we develop an approximate solution to the nonlinear modeling problem involving the linearization of the nonlinear process about a "known" reference trajectory. We limit our discussion to *discrete* nonlinear systems. Continuous solutions to this problem are developed in [6–13].

Suppose we model a *process* by a set of nonlinear stochastic vector difference equations in state–space form as

$$x(t) = a[x(t-1)] + b[u(t-1)] + w(t-1) \tag{4.135}$$

with the corresponding *measurement* model

$$y(t) = c[x(t)] + v(t) \tag{4.136}$$

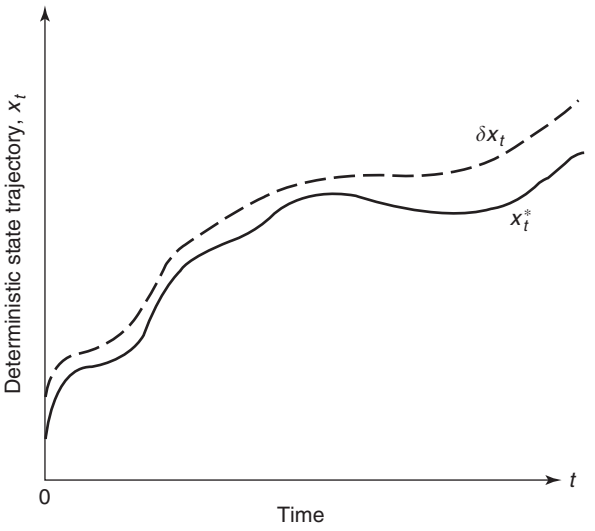


FIGURE 4.5 Linearization of a deterministic system using the reference trajectory defined by $(x^*(t), u^*(t))$.

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 $w \sim \mathcal{N}(0, R_{ww}(t-1))$, $v \sim \mathcal{N}(0, R_{vv}(t))$.

Ignoring the additive noise sources, we "linearize" the process and measurement models about a known deterministic *reference trajectory* defined by $[x^*(t), u^*(t)]$ as illustrated in Fig. 4.5[10], that is,

$$x^*(t) = a[x^*(t-1)] + b[u^*(t-1)] \tag{4.137}$$

Deviations or perturbations from this trajectory are defined by:

$$\begin{aligned} \delta x(t) &:= x(t) - x^*(t) \\ \delta u(t) &:= u(t) - u^*(t) \end{aligned}$$

Substituting the previous equations into these expressions, we obtain the perturbation trajectory as

$$\delta x(t) = a[x(t-1)] - a[x^*(t-1)] + b[u(t-1)] - b[u^*(t-1)] + w(t-1) \tag{4.138}$$

[10] 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.

The nonlinear vector functions $a[\cdot]$ and $b[\cdot]$ can be expanded into a first order Taylor series about the reference trajectory $[x^*(t), u^*(t)]$ as[11]

$$a[x(t-1)] = a[x^*(t-1)] + \frac{da[x^*(t-1)]}{dx^*(t-1)}\delta x(t-1) + \text{H.O.T.}$$

$$b[u(t-1)] = b[u^*(t-1)] + \frac{db[u^*(t-1)]}{du^*(t-1)}\delta u(t-1) + \text{H.O.T.} \quad (4.139)$$

We define the first order Jacobian matrices as

$$A[x^*(t-1)] := \frac{da[x^*(t-1)]}{dx^*(t-1)}, \quad \text{and} \quad B[u^*(t-1)] := \frac{db[u^*(t-1)]}{du^*(t-1)} \quad (4.140)$$

Incorporating the definitions of Eq. 4.140 and neglecting the *higher order terms* (H.O.T.) in Eq. 4.139, the linearized process model in Eq. 4.138 can be expressed as

$$\delta x(t) = A[x^*(t-1)]\delta x(t-1) + B[u^*(t-1)]\delta u(t-1) + w(t-1) \quad (4.141)$$

Similarly, the measurement system can be linearized by using the reference measurement

$$y^*(t) = c[x^*(t)] \quad (4.142)$$

and applying the Taylor series expansion to the nonlinear measurement model

$$c[x(t)] = c[x^*(t)] + \frac{dc[x^*(t)]}{dx^*(t)}\delta x(t) + \text{H.O.T.} \quad (4.143)$$

The corresponding measurement perturbation is defined by

$$\delta y(t) := y(t) - y^*(t) = c[x(t)] - c[x^*(t)] + v(t) \quad (4.144)$$

Substituting the first order approximation for $c[x(t)]$ leads to the linearized measurement perturbation as

$$\delta y(t) = C[x^*(t)]\delta x(t) + v(t) \quad (4.145)$$

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 then developed a *linearized Gauss-Markov perturbation model* valid for small deviations given by

$$\begin{aligned} \delta x(t) &= A[x^*(t-1)]\delta x(t-1) + B[u^*(t-1)]\delta u(t-1) + w(t-1) \\ \delta y(t) &= C[x^*(t)]\delta x(t) + v(t) \end{aligned} \quad (4.146)$$

with A, B and C the corresponding Jacobian matrices and w, v zero-mean, Gaussian.

[11] We use the shorthand notation $\frac{d(\cdot)}{d\theta^*}$ to mean $\left.\frac{d(\cdot)}{d\theta}\right|_{\theta=\theta^*}$.

We can also use linearization techniques to approximate the statistics of the process and measurements. 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

$$m_x(t) = E\{a[x(t-1)]\} + E\{b[u(t-1)]\} + E\{w(t-1)\} \tag{4.147}$$

gives

$$m_x(t) = a[m_x(t-1)] + b[u(t-1)] \tag{4.148}$$

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

$$P(t) = A[m_x(t-1)]P(t-1)A'[m_x(t-1)] + R_{ww}(t-1) \tag{4.149}$$

Using the same approach, we arrive at the accompanying measurement statistics

$$m_y(t) = c[m_x(t)] \quad \text{and} \quad R_{yy}(t) = C[m_x(t)]P(t)C'[m_x(t)] + R_{vv}(t) \tag{4.150}$$

We summarize these results in an "approximate" Gauss-Markov model of Table 4.3. Before we close consider the following example to illustrate the approximation.

Example 4.11

Consider the discrete nonlinear process given by

$$x(t) = (1 - 0.05\Delta T)x(t-1) + 0.04\Delta T x^2(t-1) + w(t-1)$$

with corresponding measurement model

$$y(t) = x^2(t) + x^3(t) + v(t)$$

where $w(t) \sim \mathcal{N}(0, R_{ww})$, $v(t) \sim \mathcal{N}(0, R_{vv})$ and $x(0) \sim \mathcal{N}(\bar{x}(0), \overline{P}(0))$. Performing the differentiations we obtain the following Jacobians:

$$A[x(t-1)] = 1 - 0.05\Delta T + 0.08\Delta T x(t-1) \quad \text{and} \quad C[x(t)] = 2x(t) + 3x^2(t)$$

△△△

TABLE 4.3 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

$$A[x^*(t-1)] \equiv \left.\frac{da[x(t-1)]}{dx(t-1)}\right|_{x=x^*(t-1)} \qquad C[x^*(t)] \equiv \left.\frac{dc[x(t)]}{dx(t)}\right|_{x=x^*(t)} \qquad x^* \to m_x$$

Although the linearization approach discussed here seems somewhat extraneous relative to the previous sections, it becomes a crucial ingredient in the *classical approach* to (approximate) nonlinear estimation of the subsequent chapters. We discuss the linear state–space approach (Kalman filter) to the estimation problem in great detail in the next chapter and then show how these linearization concepts can be used to solve the nonlinear estimation problem in the following chapter. There the popular "extended" Kalman filter processor relies heavily on the linearization techniques developed in this section for its development.

4.9 SUMMARY

In this chapter we have discussed the development of continuous-time, sampled-data and discrete-time state–space models. The stochastic variants of these three types of models were presented leading to the Gauss-Markov representations for both linear and (approximate) nonlinear systems. The discussion of both the deterministic and stochastic state–space models included a brief development of their second order

statistics. We also discussed the underlying discrete systems theory as well as a variety of time series models (*ARMAX*, *AR*, *MA*, etc.) and showed that can easily be represented in state–space form through the use of canonical forms (models). These models form the embedded structure incorporated into the majority of the Bayesian processors that will be discussed in subsequent chapters.

MATLAB NOTES

MATLAB has many commands to convert to/from state–space models to other forms useful in signal processing. Many of them reside in the Signal Processing and Control Systems toolboxes. The matrix exponential invoked by the **expm** command is determined from Taylor/Pade' approximants using the scaling and squaring approach of section 4.2. Also the commands **expmdemo1**, **expmdemo2**, and **expmdemo3** demonstrate the trade-offs of the Pade', Taylor and eigenvector approaches to calculating the matrix exponential. The ordinary differential equation method is available using the wide variety of numerical integrators available (**ode***). Converting to/from transfer functions and state–space is accomplished using the **ss2tf** and **tf2ss** commands, respectively. *ARMAX* simulations are easily accomplished using the **filter** command with a variety of options converting from armax-to/from transfer functions. The Identification Toolbox converts polynomial-based models to state–space and continuous parameters including Gauss-Markov to discrete parameters (**th2ss**, **thc2thd**, **thd2thc**). The the Third Party Toolbox *SSPACK_PC* converts continuous-time models to discrete (**SSCTOD**) performs Gauss-Markov (linear and nonlinear) simulations as well as innovations-based simulations (**SSISIM** and conversions from *GM* to innovations models (**INVTOGM**, **GMTOINV**)). See **http:www.techni-soft.net** for details.

REFERENCES

1. T. Kailath, *Linear Systems* (Englewood Cliffs, NJ: Prentice-Hall, 1980).
2. F. Szidarovszky and A. Bahill, *Linear Systems Theory* (Boca Raton, FL: CRC Press, 1980).
3. R. DeCarlo, *Linear Systems: A State Variable Approach* (Englewood Cliffs, NJ: Prentice-Hall, 1989).
4. C. Chen, *Introduction to Linear System Theory* (New York: Holt, Rhinehart, and Winston, 1984).
5. S. Tretter, *Introduction to Discrete-Time Signal Processing* (New York: Wiley, 1976).
6. A. Jazwinski, *Stochastic Processes and Filtering Theory* (New York: Academic Press, 1970).
7. A. Sage and J. Melsa, *Estimation Theory with Applications to Communications and Control* (New York: McGraw-Hill, 1971).

8. P. Maybeck, *Stochastic Models, Estimation and Control*, Vol. 1 (New York: Academic Press, 1979).
9. G. Goodwin and R. L. Payne, *Dynamic System Idenfication* (New York: Academic Press, 1976).
10. G. Goodwin and K. Sin, *Adaptive Filtering, Prediction and Control* (Englewood Cliffs, NJ: Prentice-Hall, 1984).
11. J. Mendel, *Lessons in Estimation Theory for Signal Processing, Communications, and Control* (Englewood Cliffs, NJ: Prentice-Hall, 1995).
12. R. Brown and P.C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering* (New York: Wiley, 1997).
13. J. Candy, *Model-Based Signal Processing* (Hoboken, NJ: John Wiley/IEEE Press, 2006).
14. E. Robinson and M. Silvia, *Digital Foundations of Time Series Analysis*, Vol. 1 (San Francisco: Holden-Day, 1979).
15. D. Simon, *Optimal State Estimation Kalman, H_∞ and Nonlinear Approaches* (Hoboken, NJ: Wiley, 2006).
16. M. Grewal and A. Andrews, *Kalman Filtering: Theory and Practice* (Englewood Cliffs, NJ: 1993).
17. C. Moler and C. Van Loan, "Nineteen dubious ways to compute the exponential of a matrix, twnety-five years later," *SIAM Review*, **45**, 1, 3–49, 2003.
18. G. Golub and C. Van Loan, *Matrix Computation* (Baltimore, MD: Johns Hopkins University Press, 1989).
19. B. Ho and R. Kalman, "Effective reconstruction of linear state variable models from input/output data," *Regelungstechnik*, **14**, 545–548, 1966.
20. J. Candy, M. Warren and T. Bullock, "Realization of an invariant system description from Markov sequences," *IEEE Trans. Auto. Control*, **AC-23**, 12, 93–96, 1977.
21. S. Kung, K. Arun and D. Bhaskar Rao, "State–space and singular-value decomposition-based approximation methods for the harmonic retrieval problem," *J. Opt. Soc. Am.*, **73**, 12, 1799–1811, 1983.

PROBLEMS

4.1 Suppose the stochastic process $\{y(t)\}$ is generated by

$$y(t) = a\exp(-t) + ct, \quad a, b \text{ random, then}$$

(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 $E\{ab\} = 0$.

4.2 Suppose 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 $y \sim N(am_x + b, a^2R_{xx})$.
(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

$$\begin{aligned} E\{x|y\} &= m_x + R_{xy}R_{yy}^{-1}(y + m_y), \text{ and} \\ R_{x|y} &= R_{xx} + R_{xy}R_{yy}^{-1}R_{yx} \end{aligned}$$

(*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 $e = z - E\{x|y\}$.

4.3 Assume $y(t)$ 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*) $R_{yy}(k) = C\cos(w|k|),\ |k| < \frac{\pi}{2}$.

(*c*) $R_{yy}(k) = C\exp(-a^{|k|})$.

(*Hint*: Recall the *sum decomposition*: $S_{yy}(z) = S_{yy}^{+}(z) + S_{yy}^{-}(z) - R_{yy}(0)$ with S_{yy}^{+} the one-sided Z-transform and $S_{yy}^{-}(z) = S_{yy}^{+}(z^{-1})$)

4.4 Develop a *MATLAB* program to simulate the *ARMA* process

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

where $a = 0.75,\ e \approx N(0, 0.1)$ 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?

4.5 Develop the digital filter to simulate a sequence, $y(t)$ with covariance $R_{yy}(k) = 4e^{-3|k|}$. Perform the simulation using *MATLAB*.

(*Hint*: Recall the spectral factorization (Wiener): $S_{yy}(z) = H(z) \times H(z^{-1})$ where the poles and zeros of $H(z)$ lie inside the unit circle.)

Using the "realized" digital filter perform a simulation as in the previous example and check the validity of the samples lying within bounds.

4.6 Suppose 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 above covariance.

(*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 $\pm 2\sqrt{R_{yy}(0)}$.

4.7 Suppose 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.

4.8 Suppose 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 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 $R_{yy}(0)$.

4.9 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.

4.10 Suppose we have a process characterized by difference equation

$$y(t) = x(t) + 1/2x(t-1) + 1/3x(t-2)$$

(*a*) Determine a recursion for the output covariance, $R_{yy}(k)$.

(*b*) If $x(t)$ is white with variance σ_{xx}^2, determine $R_{yy}(k)$.

(*c*) Determine the output *PSD*, $S_{yy}(z)$.

4.11 We are given a linear system characterized by the difference equation

$$y(t) - 1/5y(t-1) = \frac{1}{\sqrt{3}}x(t)$$

and the system is excited by:

(1) white Gaussian noise, $x \sim N(0, 3)$;

(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, $R_{ye}(k)$.

4.12 We are given the following Gauss-Markov model

$$\begin{aligned} x(t) &= 1/3x(t-1) + 1/2w(t-1) \\ y(t) &= 5x(t) + v(t) \\ w &\sim N(0,3) \quad v \sim N(0,2) \end{aligned}$$

(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}.

4.13 Suppose 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 and $w \sim N(0, 10^{-4})$ and $x(0) \sim N(2.5, 10^{-12})$.

(a) Calculate the covariance of x, i.e., $P(t) = \text{Cov}(x(t))$.

(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?

4.14 Suppose 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 $e \sim N(0, 10)$?

(*b*) Calculate the corresponding covariance matrix R^*_{ee}.

4.15 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

$$\begin{aligned} A(q^{-1}) &= 1 + 1.5q^{-1} + 0.7q^{-2} \\ B(q^{-1}) &= 1 + 0.5q^{-1} \\ C(q^{-1}) &= 1 + 0.7q^{-1} \\ D(q^{-1}) &= 1 + 0.5q^{-1} \end{aligned}$$

(*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 $e(t)$.)

4.16 Given a continuous-discrete Gauss-Markov model

$$\begin{aligned} \dot{x}_t &= \alpha x_t + u_t + w_t \\ y(t_k) &= \beta x(t_k) + v(t_k) \end{aligned}$$

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.

4.17 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: $\ddot{z}_t + 2\zeta\omega_n \dot{z} + \omega_n^2 z_t = \omega_n^2 w_t$; $\dot{z}_0 = z_1$; $z_0 = z_o$

4.18 Develop the continuous-discrete Gauss-Markov model for correlated process noise, that is,

$$\dot{w}_t = A_{cw} w_t + B_{cw} u_t + W_{cw} w_t^* \quad \text{for } w^* \sim \mathcal{N}(0, R_{w^* w^*})$$

4.19 Develop the approximate Gauss-Markov model for the following nonlinear state transition and measurement model are given by

$$x(t) = \frac{1}{2}x(t-1) + \frac{25x(t-1)}{1+x^2(t-1)} + 8\cos(1.2(t-1)) + w(t-1)$$

$$y(t) = \frac{x^2(t)}{20} + v(t)$$

where $w \sim \mathcal{N}(0, R_{ww}(t-1))$ and $v \sim \mathcal{N}(0, R_{vv}(t))$. The initial state is Gaussian distributed with $\bar{x}(0) \sim \mathcal{N}(0, \bar{P}(0))$.

4.20 Consider the discrete nonlinear process given by

$$x(t) = (1 - 0.05\Delta T)x(t-1) + 0.04\Delta T x^2(t-1) + w(t-1)$$

with corresponding measurement model

$$y(t) = x^2(t) + x^3(t) + v(t)$$

where $w \sim \mathcal{N}(0, R_{ww}(t-1))$ and $v \sim \mathcal{N}(0, R_{vv}(t))$. The initial state is Gaussian distributed with $\bar{x}(0) \sim \mathcal{N}(0, \bar{P}(0))$.

Develop the approximate Gauss-Markov process model for this nonlinear system.

5

CLASSICAL BAYESIAN STATE–SPACE PROCESSORS

5.1 INTRODUCTION

In this chapter we introduce the concepts of statistical signal processing from the Bayesian perspective using state–space models. We first develop the Bayesian paradigm using the generic state–space representation of the required conditional distributions and show how they propagate within the Bayesian framework. Next we start with the linear (time-varying) Gauss-Markov model and develop the required conditional distributions leading to the well-known Kalman filter processor [1]. Based on this fundamental theme, we progress to the idea of linearization of the nonlinear state–space system developed in the previous chapter, where we derive the linearized Bayesian processor (*LZ-BP*). It is shown that the resulting processor provides a solution (time-varying) to the nonlinear state estimation. We then develop the extended Bayesian processor (*XBP*) or equivalently the extended Kalman filter (*EKF*), as a special case of the *LZ-BP* linearizing about the most currently available estimate. Next we investigate a further enhancement of the *XBP* by introducing a local iteration of the nonlinear measurement system. Here the processor is called the iterated-extended Bayesian processor (*IX-BP*) and is shown to produce improved estimates at a small computational cost in most cases. We summarize the results with a case study implementing a 2D-tracking filter.

5.2 BAYESIAN APPROACH TO THE STATE–SPACE

In the previous chapter, we briefly developed deterministic and stochastic (Gauss-Markov) state–space models and demonstrated how the states propagate through the state transition mechanism for both continuous and discrete systems and their

Bayesian Signal Processing. By James V. Candy

variants. Here we again take a Bayesian perspective and assume that the state or dynamic variables evolve according to a "probabilistic" transition mechanism.

Bayesian estimation relative to the state–space models is based on extracting the unobserved or hidden dynamic (state) variables from noisy measurement data. The Markovian state vector with initial distribution, $\Pr(x(0))$, propagates temporally throughout the state–space according to the *probabilistic transition distribution*, $\Pr(x(t)|x(t-1))$, while the conditionally independent measurements evolve from the *likelihood distribution*, $\Pr(y(t)|x(t))$. We see that the dynamic state variable at time t is obtained through the transition probability based on the previous state (Markovian property), $x(t-1)$, and the knowledge of the underlying conditional probability. Once propagated to time t, the dynamic state variable is used to update or correct based on the likelihood probability and the new measurement, $y(t)$. Note that it is the knowledge of these conditional distributions that enable the Bayesian processor.

Returning to the usual model-based constructs of the dynamic state variables discussed in the previous chapter, we see that there is an *implied* equivalence between the probabilistic distributions and the underlying state/measurement transition models. Recall from Chapter 4 that the functional discrete state representation is given by

$$\begin{aligned} x(t) &= A(x(t-1), u(t-1), w(t-1)) \\ y(t) &= C(x(t), u(t), v(t)) \end{aligned} \tag{5.1}$$

where w and v are the respective process and measurement noise sources with u a known input. Here $A(\cdot)$ is the nonlinear (or linear) dynamic state transition function and $C(\cdot)$ the corresponding measurement function. Both conditional probabilistic distributions embedded within the Bayesian framework are *completely* specified by these functions and the underlying noise distributions: $\Pr(w(t-1))$ and $\Pr(v(t))$. That is, we have the (implied) equivalence[1]

$$\begin{aligned} A(x(t-1), u(t-1), w(t-1)) &\Rightarrow \Pr(x(t)|x(t-1)) \Leftrightarrow \mathcal{A}(x(t)|x(t-1)) \\ C(x(t), u(t), v(t)) &\Rightarrow \Pr(y(t)|x(t)) \quad \Leftrightarrow \mathcal{C}(y(t)|x(t)) \end{aligned} \tag{5.2}$$

Thus, the state–space model along with the noise statistics and prior distributions define the required Bayesian representation or probabilistic propagation model which defines the evolution of the states and measurements through the transition probabilities. This is sometimes a subtle point that must be emphasized. As illustrated in the diagram of Fig. 5.1, the dynamic state variables propagate throughout state–space specified by the transition probability $\mathcal{A}(x(t)|x(t-1))$ using the embedded process model. That is, the "unobserved" state at time $t-1$ requires the transition probability distribution to propagate to the state at time t. Once evolved, the state combines under the corresponding measurement at time t through the conditional likelihood distribution $\mathcal{C}(y(t)|x(t))$ using the embedded measurement model to obtain

[1] We use this notation to emphasize the influence of both process ($\mathcal{A}$) and measurement ($\mathcal{C}$) representations on the conditional distributions.

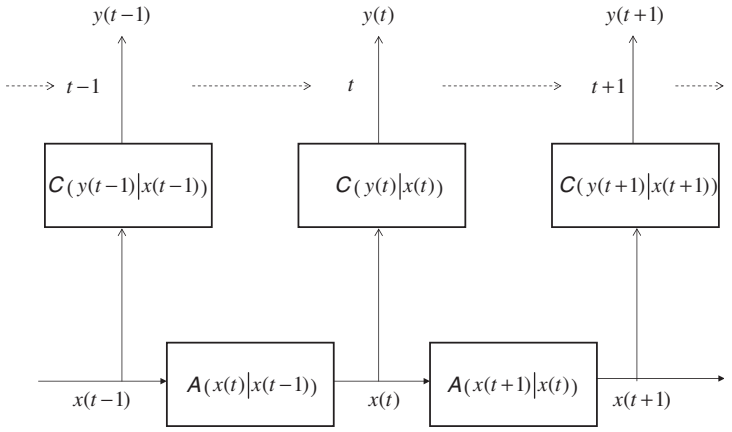


FIGURE 5.1 Bayesian state–space probabilistic evolution.

the required likelihood distribution. These events continue to evolve throughout with the states propagating through the state transition probability using the process model and the measurements generated by the states and likelihood using the measurement model. From the Bayesian perspective, the broad prior is scaled by the evidence and "narrowed" by the likelihood to estimate the posterior.

With this in mind we can now return to the original Bayesian estimation problem, define it and show (at least conceptually) the optimal solution based on the state–space representation.

The basic dynamic state estimation (signal enhancement) problem can now be stated in the Bayesian framework as:

GIVEN a set of noisy uncertain measurements, $\{y(t)\}$, and known inputs, $\{u(t)\}$, $t = 0, \ldots, N$ along with the corresponding prior distributions for the initial state and process and measurement noise sources: $\Pr(x(0))$, $\Pr(w(t-1))$, $\Pr(v(t))$ as well as the conditional transition and likelihood probability distributions: $\Pr(x(t)|x(t-1))$, $\Pr(y(t)|x(t))$ characterized by the state and measurement models: $\mathcal{A}(x(t)|x(t-1))$, $\mathcal{C}(y(t)|x(t))$, **FIND** the "best" estimate of the filtering *posterior*, $\hat{\Pr}(x(t)|Y_t)$, and its associated statistics.

It is interesting to note that the entire *Bayesian system* can be defined by the set Σ as

$$\Sigma := [\{y(t)\}, \{u(t)\}, \Pr(x(0)), \Pr(w(t-1)), \Pr(v(t)), \mathcal{A}(x(t)|x(t-1)), \mathcal{C}(y(t)|x(t))]$$

Compare this to the model-based solutions to follow where we obtain closed form analytic expressions for these distributions.

Analytically, to generate the model-based version of the sequential Bayesian processor, we replace the transition and likelihood distributions with the conditionals

of Eq. 5.2. The solution to the signal enhancement or equivalently state estimation problem is given by the filtering distribution, $\Pr(x(t)|Y_t)$ which was solved previously in Sec. 2.5 (see Table 2.1). We start with the prediction recursion characterized by the *Chapman-Kolmogorov equation* replacing the transition probability with the implied model-based conditional, that is,

$$\Pr(x(t)|Y_{t-1}) = \int \overbrace{\mathcal{A}(x(t)|x(t-1))}^{\text{Embedded Process Model}} \times \overbrace{\Pr(x(t-1)|Y_{t-1})}^{\text{Prior}} dx(t-1) \tag{5.3}$$

Next we incorporate the model-based likelihood into the posterior equation with the understanding that the process model has been incorporated into the prediction

$$\Pr(x(t)|Y_t) = \overbrace{\mathcal{C}(y(t)|x(t))}^{\text{Embedded Measurement Model}} \times \overbrace{\Pr(x(t)|Y_{t-1})}^{\text{Predicition}} \Big/ \Pr(y(t)|Y_{t-1}) \tag{5.4}$$

So we see from the Bayesian perspective that the sequential Bayesian processor employing the state–space representation of Eq. 5.1 is straightforward. Next let us investigate a more detailed development of the processor resulting in a closed-form solution—the linear Kalman filter.

5.3 LINEAR BAYESIAN PROCESSOR (LINEAR KALMAN FILTER)

In this section we constrain the state–space model to be linear (time-varying) and apply the Bayesian approach to obtain the optimal processor assuming additive Gaussian noise.

Suppose we are given the linear discrete Gauss-Markov model of the previous section (ignoring u with $W = I$ for notational simplicity) and we would like to develop the Bayesian processor. Since we know the processes are linear and Gaussian, then we know that the required distributions will also be Gaussian. To develop the processor for this case, we start with the *prediction* equation[2] and use the process model of Eq. 4.78, that is,

$$\Pr(x(t)|Y_{t-1}) = \int \mathcal{A}(x(t)|x(t-1)) \times \Pr(x(t-1)|Y_{t-1})\, dx(t-1)$$

where the *filtered conditional*[3] is:

$$\Pr(x(t-1)|Y_{t-1}) \sim \mathcal{N}(x(t) : \hat{x}(t-1|t-1), \tilde{P}(t-1|t-1))$$

[2] We have changed Gaussian distribution notation to include the random process, that is, $\mathcal{N}(m_z, R_{zz}) \rightarrow \mathcal{N}(z : m_z, R_{zz})$.

[3] This notation is defined in terms of *conditional means* and *covariances* by: $\hat{x}(t|t) := E\{x(t)|Y_t\}$ and $\tilde{P}(t|t) := \text{cov}(\tilde{x}(t|t))$ for the *state estimation error*, $\tilde{x}(t|t) := x(t) - \hat{x}(t|t)$.

Now using the process model, we have that

$$\mathcal{A}(x(t)|x(t-1)) \sim \mathcal{N}(x(t) : A(t-1)\hat{x}(t-1|t-1), A(t-1)\tilde{P}(t-1|t-1) \times A'(t-1) + R_{ww}(t-1))$$

which follows directly from the linearity of the conditional expectation operator, that is,

$$\begin{aligned}\hat{x}(t|t-1) &= E\{x(t)|Y_{t-1}\} = E\{A(t-1)x(t-1) + w(t-1)|Y_{t-1}\} \\ &= A(t-1)\hat{x}(t-1|t-1) \end{aligned} \tag{5.5}$$

Using this result, the predicted *state estimation error* can be obtained as

$$\begin{aligned}\tilde{x}(t|t-1) &= x(t) - \hat{x}(t|t-1) \\ &= [A(t-1)x(t-1) + w(t-1)] - A(t-1)\hat{x}(t-1|t-1) \\ &= A(t-1)\tilde{x}(t-1|t-1) + w(t-1) \end{aligned} \tag{5.6}$$

and the corresponding *state error covariance*, $\tilde{P}(t|t-1) = E\{\tilde{x}(t|t-1)\tilde{x}'(t|t-1)\}$ is easily derived. Summarizing, the conditional means and covariances that completely characterize the current (Gaussian) state evolve according to the following equations:

$$\begin{aligned}\hat{x}(t|t-1) &= A(t-1)\hat{x}(t-1|t-1) && \text{[Prediction]} \\ \tilde{P}(t|t-1) &= A(t-1)\tilde{P}(t-1|t-1)A'(t-1) + R_{ww}(t-1) && \text{[Prediction Covariance]}\end{aligned}$$

Substituting these multivariate Gaussian distributions (transition and filtered) into the prediction equation, we have that

$$\Pr(x(t)|Y_{t-1}) \sim \mathcal{N}(x(t) : \hat{x}(t|t-1), \tilde{P}(t|t-1))$$

With the prediction distribution available, we require the correction or update distribution obtained from the likelihood and the measurement model of Eq. 5.4, that is,

$$\Pr(x(t)|Y_t) = \frac{\mathcal{C}(y(t)|x(t)) \times \Pr(x(t)|Y_{t-1})}{\Pr(y(t)|Y_{t-1})}$$

Under the Gauss-Markov model assumptions, we know that each of the conditional distributions can be expressed in terms of the Gaussian distribution as:

$$\begin{aligned}\mathcal{C}(y(t)|x(t)) &\sim \mathcal{N}(y(t) : C(t)x(t), R_{vv}(t)) \\ \Pr(x(t)|Y_{t-1}) &\sim \mathcal{N}(x(t) : \hat{x}(t|t-1), \tilde{P}(t|t-1)) \\ \Pr(y(t)|Y_{t-1}) &\sim \mathcal{N}(y(t) : \hat{y}(t|t-1), R_{ee}(t))\end{aligned}$$

for $R_{ee}(t)$ the *innovations covariance* with *innovations* defined by $e(t) := y(t) - \hat{y}(t|t-1)$ and *predicted* or *filtered* measurement given by $\hat{y}(t|t-1) = C(t)\hat{x}(t|t-1)$.

Substituting these probabilities into Eq. 5.4 and combining all constants into a single constant κ, we obtain

$$\begin{aligned}\Pr(x(t)|Y_t) = \kappa \times \exp\left[-\frac{1}{2}(y(t) - C(t)x(t))' R_{vv}^{-1}(t)(y(t) - C(t)x(t))\right] \\ \times \exp\left[-\frac{1}{2}(x(t) - \hat{x}(t|t-1))' \tilde{P}^{-1}(t|t-1)(x(t) - \hat{x}(t|t-1))\right] \\ \times \exp\left[+\frac{1}{2}(y(t) - \hat{y}(t|t-1))' R_{ee}^{-1}(t)(y(t) - \hat{y}(t|t-1))\right]\end{aligned}$$

Recognizing the measurement noise, state estimation error and innovation in the above terms, we have that the posterior probability is given in terms of the Gauss-Markov model by

$$\begin{aligned}\Pr(x(t)|Y_t) = \kappa \times \exp\left[-\frac{1}{2}v'(t)R_{vv}^{-1}(t)v(t)\right] \\ \times \exp\left[-\frac{1}{2}\tilde{x}'(t|t-1)\tilde{P}^{-1}(t|t-1)\tilde{x}(t|t-1)\right] \\ \times \exp\left[+\frac{1}{2}e'(t)R_{ee}^{-1}(t)e(t)\right]\end{aligned} \tag{5.7}$$

So we see that the posterior distribution can be estimated under the multivariate Gaussian assumptions and the corresponding linear (time-varying) Gauss-Markov model. This is the *optimal* Bayesian processor under these assumptions. In most cases we are not able to characterize the distributions in closed form and must resort to numerical (simulation-based) solutions.

We realize at this point that we have the optimal Bayesian predictor and posterior, but we still have not extracted the optimal state estimates explicitly and its associated performance metric. Recall from the batch Bayesian solutions of Sec. 2.1 that once we have the posterior, we can estimate a variety of statistics using it as the basis. In this case, the optimal Bayesian processor will be the one that maximizes the posterior; therefore, we continue the development of the linear filter by deriving the Bayesian *MAP* estimator.

Starting with the *MAP* equation of Eq. 2.3 and taking natural logarithms of each side gives

$$\ln \Pr(x(t)|Y_t) = \ln \Pr(y(t)|x(t)) + \ln \Pr(x(t)|Y_{t-1}) - \ln \Pr(y(t)|Y_{t-1})$$

In terms of multivariate Gaussian, the posterior is given by

$$\ln \Pr(x(t)|Y_t) = \ln \kappa - \frac{1}{2} v'(t) R_{vv}^{-1}(t) v(t) - \frac{1}{2} \tilde{x}'(t|t-1) \tilde{P}^{-1}(t|t-1) \tilde{x}(t|t-1) + \frac{1}{2} e'(t) R_{ee}^{-1}(t) e(t) \tag{5.8}$$

with

$$\begin{aligned} \tilde{x}(t|t-1) &= x(t) - \hat{x}(t|t-1) && \text{[State Error]} \\ e(t) &= y(t) - C(t)\hat{x}(t|t-1) = C(t-1)\tilde{x}(t|t-1) + v(t) && \text{[Innovation]} \end{aligned}$$

The *MAP* estimate is then obtained by differentiating Eq. 5.8, setting it to zero and solving, that is,

$$\nabla_x \ln \Pr(x(t)|Y_t)\big|_{x=\hat{X}_{\text{map}}} = 0 \tag{5.9}$$

Using the *chain rule* of the gradient operator (see Eq. 2.11), we obtain the following expression. Note that the last term of Eq. 5.8 is not a function of $x(t)$, but just the data; therefore, its gradient is null.

$$\nabla_x \ln \Pr(x(t)|Y_t) = C'(t) R_{vv}^{-1}(t)[y(t) - C(t)x(t)] - \tilde{P}^{-1}(t|t-1)\tilde{x}(t|t-1) \tag{5.10}$$

Setting Eq. 5.10 to zero and solving for $x(t)$ gives the Bayesian *MAP* estimate

$$\hat{X}_{\text{map}}(t) = [C'(t) R_{vv}^{-1}(t) C(t) + \tilde{P}^{-1}(t|t-1)]^{-1} \times [\tilde{P}^{-1}(t|t-1)\hat{x}(t|t-1) + C'(t) R_{vv}^{-1}(t) y(t)] \tag{5.11}$$

This relation can be simplified by using a form of the *matrix inversion lemma* [1] defined by the following equation

$$(A + BD')^{-1} = A^{-1} - A^{-1}B(I + D'A^{-1}B)^{-1})D'A^{-1} \tag{5.12}$$

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

$$\begin{aligned} &[\tilde{P}^{-1}(t|t-1) + C'(t) R_{vv}^{-1}(t) C(t)]^{-1} \\ &\quad = \tilde{P}(t|t-1) - \tilde{P}(t|t-1) C'(t) R_{vv}^{-1}(t) (I + C(t)\tilde{P}(t|t-1) C'(t) R_{vv}^{-1}(t))^{-1} \\ &\qquad \times C(t)\tilde{P}(t|t-1) \end{aligned} \tag{5.13}$$

Making the observation that the term in parenthesis on the right hand side of Eq. 5.13 can be rewritten by factoring out $R_{vv}^{-1}(t)$ as

$$(I + C(t)\tilde{P}(t|t-1)C'(t)R_{vv}^{-1}(t))^{-1} = R_{vv}(t)(R_{vv}(t) + C(t)\tilde{P}(t|t-1)C'(t))^{-1} \tag{5.14}$$

then Eq. 5.14 can also be expressed as

$$[\tilde{P}^{-1}(t|t-1) + C'(t)R_{vv}^{-1}(t)C(t)]^{-1} = \tilde{P}(t|t-1) - \tilde{P}(t|t-1)C'(t)(R_{vv}(t) + C(t)\tilde{P}(t|t-1)C'(t))^{-1}C(t)\tilde{P}(t|t-1) \tag{5.15}$$

The innovations covariance can be expressed as

$$R_{ee}(t) = \text{Cov}(e(t)) = C(t)\tilde{P}(t|t-1)C'(t) + R_{vv}(t)$$

and substituting Eq. 5.15 becomes

$$[\tilde{P}^{-1}(t|t-1) + C'(t)R_{vv}^{-1}(t)C(t)]^{-1} = \tilde{P}(t|t-1) - \tilde{P}(t|t-1)C'(t)R_{ee}^{-1}(t)C(t)\tilde{P}(t|t-1) = (I - K(t)C(t))\tilde{P}(t|t-1) \tag{5.16}$$

where $K(t) = \tilde{P}(t|t-1)C'(t)R_{ee}^{-1}(t)$ is the gain. We see that Eq. 5.16 is simply the *updated error covariance*, $\tilde{P}(t|t)$, equivalent to

$$\tilde{P}(t|t) \equiv [\tilde{P}^{-1}(t|t-1) + C'(t)R_{vv}^{-1}(t)C(t)]^{-1} \tag{5.17}$$

Thus we can eliminate the first bracketed term in Eq. 5.11 to give

$$\hat{X}_{\text{map}}(t) = \tilde{P}(t|t) \times [\tilde{P}^{-1}(t|t-1)\hat{x}(t|t-1) + C'(t)R_{vv}^{-1}(t)y(t)]$$

Solving Eq. 5.17 for $\tilde{P}(t|t-1)$, we can substitute the result into the above equation to give

$$\hat{X}_{\text{map}}(t) = \tilde{P}(t|t) \times [\tilde{P}^{-1}(t|t) - C'(t)R_{vv}^{-1}(t)C(t)\hat{x}(t|t-1) + C'(t)R_{vv}^{-1}(t)y(t)] \tag{5.18}$$

Multiplying out, regrouping terms and factoring, this relation can be rewritten as

$$\hat{X}_{\text{map}}(t) = \hat{x}(t|t-1) + (\tilde{P}(t|t)C'(t)R_{vv}^{-1}(t))[y(t) - C(t)\hat{x}(t|t-1)] \tag{5.19}$$

or finally

$$\hat{X}_{\text{map}}(t) = \hat{x}(t|t) = \hat{x}(t|t-1) + K(t)e(t) \tag{5.20}$$

Now we only need to show the equivalence of the gain expression using the updated instead of predicted error covariance, that is,

$$\begin{aligned} K(t) &= \tilde{P}(t|t-1)C'(t)R_{ee}^{-1}(t) = \tilde{P}(t|t)\tilde{P}^{-1}(t|t)(\tilde{P}(t|t-1)C'(t)R_{ee}^{-1}(t)) \\ &= \tilde{P}(t|t)[C'(t)R_{vv}^{-1}(t)C(t) + \tilde{P}^{-1}(t|t-1)]\tilde{P}(t|t-1)C(t)R_{ee}^{-1}(t) \\ &= \tilde{P}(t|t)C'(t)R_{vv}^{-1}(t)[C(t)\tilde{P}(t|t-1)C'(t) + R_{vv}(t)]R_{ee}^{-1}(t) \end{aligned} \tag{5.21}$$

TABLE 5.1 Linear *BP* (Kalman Filter) Algorithm

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)
Update	
$\hat{x}(t\|t) = \hat{x}(t\|t-1) + K(t)e(t)$	(state update)
$\tilde{P}(t\|t) = [I - K(t)C(t)]\tilde{P}(t\|t-1)$	(covariance update)
Initial Conditions	
$\hat{x}(0\|0) \quad \tilde{P}(0\|0)$	

which gives the desired result from the definition of innovations covariance. We now have two *equivalent* expressions in terms of the updated or predicted error covariances that can be used to calculate the gain

$$K(t) = \tilde{P}(t|t)C'(t)R_{vv}^{-1}(t) \equiv \tilde{P}(t|t-1)C'(t)R_{ee}^{-1}(t) \tag{5.22}$$

which completes the Bayes' approach to the signal enhancement or equivalently state estimation problem yielding the optimum linear Bayesian processor (Kalman filter). A summary of the linear *BP* algorithm is shown in Table 5.1.

The design of linear Bayesian processors under the Gauss-Markov assumptions is well-understood [1–9]. Based on a variety of properties both theoretically well-founded and pragmatically applied with high success, the *minimum (error) variance* design procedure has evolved [10–14]. We summarize the design steps below and subsequently using the notation of the Bayesian processor algorithm in Table 5.1.

It is important to realize that a *necessary* and *sufficient* condition that the linear *BP* (under the *GM* constraints) is *optimal* is that the innovation sequence is zero-mean and white or uncorrelated! This is the first and most important step in *BP* design. If this condition does not hold then the underlying model and *GM* assumptions are invalid. Therefore, we briefly mention the minimum variance design procedure here and provide more details in Sec. 5.7 where pragmatic statistical tests are developed. We will apply the procedure to the following processors (linear and nonlinear) in the example problems and then provide the design details subsequently.

The *minimum (error) variance design procedure* is:

1. Check that the innovations sequence is *zero-mean*.
2. Check that the innovations sequence is *white*.

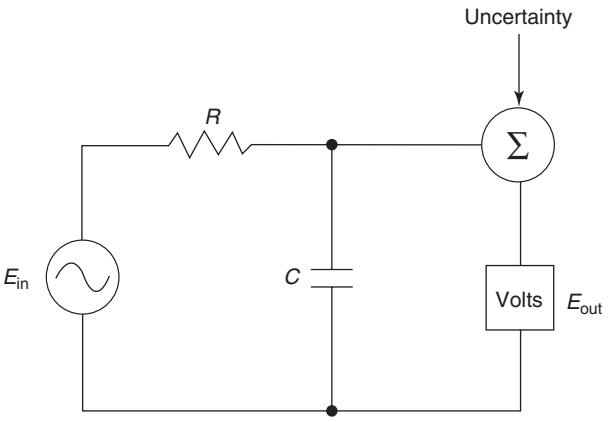


FIGURE 5.2 *RC*-circuit problem diagram.

3. Check that the innovations sequence is *uncorrelated* in time with input u.
4. Check that the innovations sequence is *lies within* the confidence limits constructed from R_{ee} predicted by the Bayesian processor.
5. Check that the innovations sequence variance is *reasonably close* to the estimated (sample) variance, $\hat{R}_{ee}$.
6. Check that the state estimation error, $\tilde{x}(t|t)$, *lies within* the confidence limits constructed from $\tilde{P}(t|t)$ predicted by the Bayesian processor.
7. Check that the state error variance is *reasonably close* to the estimated (sample) variance, $\hat{\tilde{P}}(t|t)$.

These is the basic "cookbook" approach to linear Bayesian processor design. Before we close this section, let us consider a simple linear example to demonstrate the ideas.

Example 5.1

Suppose we have the *RC* circuit as shown in Fig. 5.2. 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 *BP* 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 *BP* formulation; therefore, approximating the derivatives with first differences and substituting, 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 $x = e$, $y = e_{out}$, and $u = I_{in}$, we obtain

$$\begin{aligned} x(t) &= 0.97x(t-1) + 100u(t-1) + w(t-1) \\ y(t) &= 2x(t) + v(t) \end{aligned}$$

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 $\triangle V = 2(\frac{1}{2}\,4V)$. 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* is used to simulate this system [8]. The results are shown in Fig. 5.3. The simulated and true (mean) states (voltages) are shown in Fig. 5.3a 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 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 Fig. 5.3b. Again the 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, $[m_x(t) \pm 1.96\sqrt{P(t)}]$ and $[m_y(t) \pm 1.96\sqrt{R_{yy(t)}}]$ are shown in Fig. 5.3c.

With the data simulated, we now consider the design of the *BP*. In the ideal *BP* 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:

$$\begin{aligned} &\hat{x}(t|t-1) = 0.97\hat{x}(t-1|t-1)) + 100u(t-1) && \text{[Predicted State]} \\ &\tilde{P}(t|t-1) = 0.94\tilde{P}(t-1|t-1) + 0.0001 && \text{[Predicted Covariance]} \\ &e(t) = y(t) - 2\hat{x}(t|t-1) && \text{[Innovation]} \\ &R_{ee}(t) = 4\tilde{P}(t|t-1) + 4 && \text{[Innovations Covariance]} \end{aligned}$$

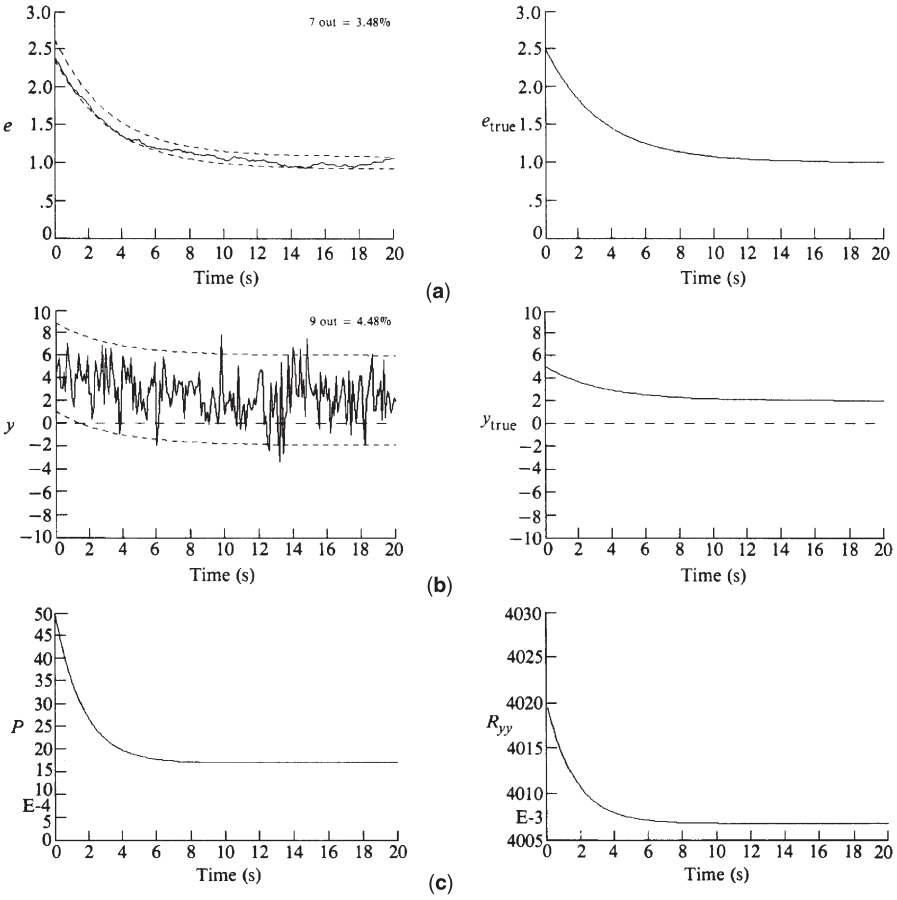


FIGURE 5.3 *RC* circuit problem Gauss-Markov simulation. (a) Simulated and true (mean) output voltage. (b) Simulated and true (mean) measurement. (c) Simulated state and measurement variances.

$$K(t) = 2\frac{\tilde{P}(t|t-1)}{4\tilde{P}(t|t-1)+4} \qquad \text{[Gain]}$$

$$\hat{x}(t|t) = \hat{x}(t|t-1) + K(t)e(t) \qquad \text{[Updated State]}$$

$$\tilde{P}(t|t) = \frac{\tilde{P}(t|t-1)}{\tilde{P}(t|t-1)+1} \qquad \text{[Updated Covariance]}$$

The estimator is also designed using *SSPACK_PC* and the results are shown in Fig. 5.4. In Fig. 5.4a 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,

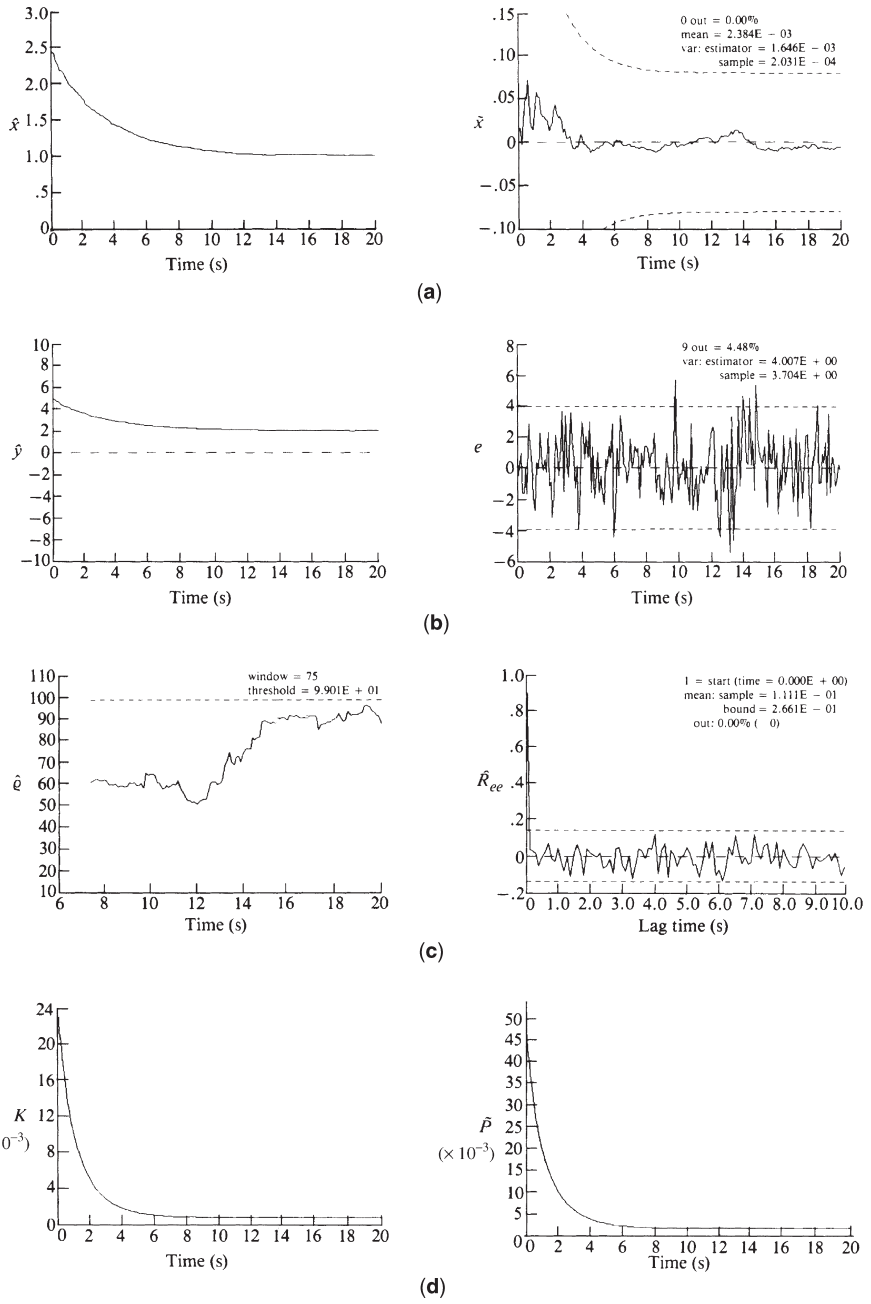


FIGURE 5.4 *BP* design for *RC* circuit problem. (a) Estimated state (voltage) and error. (b) Filtered voltage measurement and error (innovations). (c) *WSSR* and zero-mean/whiteness tests. (d) Gain and updated error covariance.

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 Fig. 5.4b, we see the filtered measurement ($\hat{y}(t|t-1)$) and corresponding innovation sequence along with the confidence limits provided by the processor. Here only 4.5% of the samples exceed the bounds and the variance predicted by the filter is close to the sample variance estimate ($4.0 \sim 3.7$). The weighted sum-squared residual (*WSSR*) statistic, zero-mean, and whiteness tests are shown in Fig. 5.4c. 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 updated error covariance as monotonically decreasing functions that reach a steady-state (constant) value at approximately 8 *sec*. This completes the example of an ideally "tuned" BP. △△△

5.4 LINEARIZED BAYESIAN PROCESSOR (LINEARIZED 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 Bayesian processor based on the underlying linearized state–space model. Many processes in practice 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].

Recall from the previous chapter that our *process* is characterized by a set of nonlinear stochastic vector difference equations in state–space form as

$$x(t) = a[x(t-1)] + b[u(t-1)] + w(t-1) \tag{5.23}$$

with the corresponding *measurement* model

$$y(t) = c[x(t)] + v(t) \tag{5.24}$$

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 $w \sim \mathcal{N}(0, R_{ww}(t))$, $v \sim \mathcal{N}(0, R_{vv}(t))$.

In Chapter 4 we linearized a deterministic nonlinear model using a first-order Taylor series expansion for the functions, a, b, and c and developed a *linearized Gauss-Markov perturbation model* valid for small deviations given by

$$\begin{aligned} \delta x(t) &= A[x^*(t-1)]\delta x(t-1) + B[u^*(t-1)]\delta u(t-1) + w(t-1) \\ \delta y(t) &= C[x^*(t)]\delta x(t) + v(t) \end{aligned} \tag{5.25}$$

with A, B and C the corresponding Jacobian matrices and w, v zero-mean, Gaussian.

We used linearization techniques to approximate the statistics of Eqs. 5.23 and 5.24 and summarized these results in an "approximate" Gauss-Markov model of Table 4.3 of the previous chapter. Using this perturbation model, we will now incorporate it to construct a Bayesian processor the embeds the $(A[\cdot], B[\cdot], C[\cdot])$ Jacobians linearized about the reference trajectory $[x^*, u^*]$. Each of the Jacobians are deterministic and time-varying, since they are updated at each time-step. Replacing the (A, B) matrices and $\hat{x}(t|t-1)$ in Table 5.1, respectively, by the Jacobians and $\delta\hat{x}(t|t-1)$, we obtain the state perturbation predicted estimate

$$\delta\hat{x}(t|t-1) = A[x^*(t-1)]\delta\hat{x}(t-1|t-1) + B[u^*(t-1)]\delta u(t-1) \tag{5.26}$$

For the Bayesian estimation problem, we are interested in the state estimate $\hat{x}(t|t-1)$ not its deviation $\delta\hat{x}(t|t-1)$. From the definition of the perturbation in Sec. 4.8, we have

$$\hat{x}(t|t-1) = \delta\hat{x}(t|t-1) + x^*(t) \tag{5.27}$$

where the reference trajectory $x^*(t)$ was defined previously as

$$x^*(t) = a[x^*(t-1)] + b[u^*(t-1)] \tag{5.28}$$

Substituting this relation along with Eq. 5.26 into Eq. 5.27 gives

$$\begin{aligned}\hat{x}(t|t-1) \;=\; & a[x^*(t-1)] + A[x^*(t-1)][\hat{x}(t-1|t-1) - x^*(t-1)] \\ & + b[u^*(t-1)] + B[u^*(t-1)][u(t-1) - u^*(t-1)]\end{aligned} \tag{5.29}$$

The corresponding *perturbed* innovation can also be found directly

$$\begin{aligned}\delta e(t) &= \delta y(t) - \delta\hat{y}(t|t-1) = (y(t) - y^*(t)) - (\hat{y}(t|t-1) - y^*(t)) \\ &= y(t) - \hat{y}(t|t-1) = e(t)\end{aligned} \tag{5.30}$$

Using the linear *BP* with deterministic Jacobian matrices results in

$$\delta\hat{y}(t|t-1) = C[x^*(t)]\delta\hat{x}(t|t-1) \tag{5.31}$$

and therefore using this relation and Eq. 4.142 for the reference measurement, we have

$$\hat{y}(t|t-1) = y^*(t) + C[x^*(t)]\delta\hat{x}(t|t-1) = c[x^*(t)] + C[x^*(t)]\delta\hat{x}(t|t-1) \tag{5.32}$$

Therefore it follows that the innovation is

$$e(t) = y(t) - c[x^*(t)] - C[x^*(t)][\hat{x}(t|t-1) - x^*(t)] \tag{5.33}$$

The updated estimate is easily found by substituting Eq. 5.27 to obtain

$$\begin{aligned} \delta\hat{x}(t|t) &= \delta\hat{x}(t|t-1) + K(t)e(t) \\ [\hat{x}(t|t) - x^*(t)] &= [\hat{x}(t|t-1) - x^*(t)] + K(t)e(t) \end{aligned} \tag{5.34}$$

which yields the identical update 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 $\delta\tilde{P}(t|\cdot) = \tilde{P}(t|\cdot)$ and therefore,

$$\delta\tilde{x}(t|\cdot) = \delta x(t) - \delta\hat{x}(t|\cdot) = [x(t) - x^*(t)] - [\hat{x}(t|\cdot) - x^*(t)] = x(t) - \hat{x}(t|\cdot) \tag{5.35}$$

The gain is just a function of the measurement linearization, $C[x^*(t)]$ completing the algorithm. We summarize the discrete linearized Bayesian processor (Kalman filter) in Table 5.2.

TABLE 5.2 Linearized *BP* (Kalman Filter) Algorithm

Prediction

$\hat{x}(t|t-1) = a[x^*(t-1)] + A[x^*(t-1)][\hat{x}(t-1|t-1) - x^*(t-1)]$
$+ \mathrm{b}[u^*(t-1)] + B[u^*(t-1)][u(t-1) - u^*(t-1)]$ (state prediction)

$\tilde{P}(t|t-1) = A[x^*(t-1)]\tilde{P}(t-1|t-1)A'[x^*(t-1)] + R_{ww}(t-1)$ (covariance prediction)

Innovation

$e(t) = y(t) - c[x^*(t)] - C[x^*(t)][\hat{x}(t|t-1) - x^*(t)]$ (innovation)

$R_{ee}(t) = C[x^*(t)]\tilde{P}(t|t-1)C'[x^*(t)] + R_{vv}(t)$ (innovation covariance)

Gain

$K(t) = \tilde{P}(t|t-1)C'[x^*(t)]R_{ee}^{-1}(t)$ (gain or weight)

Update

$\hat{x}(t|t) = \hat{x}(t|t-1) + K(t)e(t)$ (state update)

$\tilde{P}(t|t) = [I - K(t)C[x^*(t)]]\tilde{P}(t|t-1)$ (covariance update)

Initial Conditions

$\hat{x}(0|0) \quad \tilde{P}(0|0)$

Jacobians

$$A[x^*(t-1)] \equiv \left.\frac{da[x(t-1)]}{dx(t-1)}\right|_{x=x^*(t-1)} \qquad B[u^*(t-1)] \equiv \left.\frac{db[u(t-1)]}{du(t-1)}\right|_{u=u^*(t-1)}$$

$$C[x^*(t)] \equiv \left.\frac{dc[x(t)]}{dx(t)}\right|_{x=x^*(t)}$$

In a more formal framework, the *LZ-BP* 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 Sec. 5.3.

The *a posteriori probability* is given by

$$\Pr(x(t)|Y_t) = \frac{\Pr(y(t)|x(t)) \times \Pr(x(t)|Y_{t-1})}{\Pr(y(t)|Y_{t-1})} \tag{5.36}$$

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:

1. $\Pr(y(t)|x(t))$: $\quad \mathcal{N}(c[x(t)], R_{vv}(t))$
2. $\Pr(x(t)|Y_{t-1})$: $\quad \mathcal{N}(\hat{x}(t|t-1), \tilde{P}(t|t-1))$
3. $\Pr(y(t)|Y_{t-1})$: $\quad \mathcal{N}(\hat{y}(t|t-1), R_{ee}(t))$

Using the nonlinear models developed earlier, substituting the Gaussian probabilities and taking logarithms, we obtain the logarithmic *a posteriori* probability as

$$\begin{aligned}\ln \Pr(x(t)|Y_t) \;=\; & \ln \kappa - \frac{1}{2} v'(t) R_{vv}^{-1}(t) v(t) - \frac{1}{2} \tilde{x}'(t|t-1) \tilde{P}^{-1}(t|t-1) \tilde{x}(t|t-1) \\ & + \frac{1}{2} e'(t) R_{ee}^{-1}(t) e(t)\end{aligned} \tag{5.37}$$

The *MAP* estimate is then obtained by differentiating this equation, setting the result to zero and solving for $x(t)$, that is,

$$\nabla_x \ln \Pr(x(t)|Y_t)|_{x=\hat{X}_{map}} = \mathbf{0} \tag{5.38}$$

Before we attempt to derive the *MAP* estimate, we first linearize about a reference trajectory, $x^* \to x$ with the nonlinear measurement model approximated by a first order Taylor series

$$c[x(t)] \approx c[x^*(t)] + \frac{dc[x^*(t)]}{dx^*(t)} \delta x(t) = c[x^*(t)] + C[x^*(t)](x(t) - x^*(t))$$

Substituting this result into Eq. 5.37, we obtain

$$\begin{aligned}\ln \Pr(x(t)|Y_t) = \ln \kappa & - \frac{1}{2} (y(t) - c[x^*(t)] - C[x^*(t)](x(t) - x^*(t)))' R_{vv}^{-1}(t) \\ & \times (y(t) - c[x^*(t)] - C[x^*(t)](x(t) - x^*(t))) - \frac{1}{2} \tilde{x}'(t|t-1) \tilde{P}^{-1}(t|t-1) \tilde{x}(t|t-1) \\ & + \frac{1}{2} (y(t) - \hat{y}(t|t-1))' R_{ee}^{-1}(t) (y(t) - \hat{y}(t|t-1))\end{aligned} \tag{5.39}$$

Applying the *chain rule* and the gradient operator (see Chapter 3), we obtain the following expression. Note that the last term of Eq. 5.39 is not a function of $x(t)$, but just the data, so it is null.

$$\begin{aligned}\nabla_x \ln \Pr(x(t)|Y_t) = & -C'[x^*(t)]R_{vv}^{-1}(t)(y(t) - c[x^*(t)] - C[x^*(t)](x(t) - x^*(t))) \\ & - \tilde{P}^{-1}(t|t-1)[x(t) - \hat{x}(t|t-1)] = 0 \end{aligned} \tag{5.40}$$

Multiplying through and grouping like terms in $x(t)$ gives

$$\begin{aligned}& C'[x^*(t)]R_{vv}^{-1}(t)(y(t) - c[x^*(t)] + C[x^*(t)]x^*(t)) \\ & - [\tilde{P}^{-1}(t|t-1) + C'[x^*(t)]R_{vv}^{-1}(t)C[x^*(t)]]x(t) + \tilde{P}^{-1}(t|t-1)\hat{x}(t|t-1) = 0 \end{aligned} \tag{5.41}$$

Solving for $x(t) = \hat{X}_{\text{map}}(t)$ gives

$$\begin{aligned}\hat{X}_{\text{map}}(t) = & [\tilde{P}^{-1}(t|t-1) + C'[x^*(t)]R_{vv}^{-1}(t)C[x^*(t)]]^{-1} \\ & \times [\tilde{P}^{-1}(t|t-1)\hat{x}(t|t-1) + C'[x^*(t)]R_{vv}^{-1}(t) \\ & \times (y(t) - c[x^*(t)] + C[x^*(t)]x^*(t))] \end{aligned} \tag{5.42}$$

Using the matrix inversion manipulations of Eqs. 5.12–5.17 with $C(t) \longrightarrow C[x^*(t)]$, the first term in Eq. 5.42 becomes

$$\begin{aligned}& [\tilde{P}^{-1}(t|t-1) + C'[x^*(t)]R_{vv}^{-1}(t)C[x^*(t)]]^{-1} \\ & \quad = (I - \tilde{P}(t|t-1)C'[x^*(t)]R_{ee}^{-1}(t)C[x^*(t)])\tilde{P}^{-1}(t|t-1) \\ & \quad = (I - K(t)C[x^*(t)])\tilde{P}(t|t-1) \equiv \tilde{P}(t|t) \end{aligned} \tag{5.43}$$

where K is the Kalman gain, R_{ee} is the innovations covariance of the *LZ-BP* with this expression precisely the *updated error covariance*, $\tilde{P}(t|t)$, as in Table 5.2.

Solving this equation for the inverse of the predicted error covariance gives

$$\tilde{P}^{-1}(t|t-1) = \tilde{P}^{-1}(t|t) - C'[x^*(t)]R_{vv}^{-1}(t)C[x^*(t)] \tag{5.44}$$

and substituting into Eq. 5.42 using the results of Eq. 5.43 yields

$$\begin{aligned}\hat{X}_{\text{map}}(t) = & \tilde{P}(t|t)\,[(\tilde{P}^{-1}(t|t)\hat{x}(t|t-1) - C'[x^*(t)]R_{vv}^{-1}(t)C[x^*(t)]\hat{x}(t|t-1)) \\ & + C'[x^*(t)]R_{vv}^{-1}(t)(y(t) - c[x^*(t)] + C[x^*(t)]x^*(t))] \end{aligned} \tag{5.45}$$

Multiplying through by the updated error covariance and recognizing the expression for the Kalman gain gives

$$\begin{aligned}\hat{X}_{\text{map}}(t) = & \hat{x}(t|t-1) - K(t)C[x^*(t)]\hat{x}(t|t-1) + K(t)C[x^*(t)]x^*(t) \\ & + K(t)(y(t) - c[x^*(t)]) \end{aligned} \tag{5.46}$$

which leads to the final expression for the linearized *MAP* estimate as

$$\hat{X}_{\text{map}}(t) = \hat{x}(t|t) = \hat{x}(t|t-1) + K(t)(y(t) - c[x^*(t)] - C[x^*(t)](\hat{x}(t|t-1) - x^*(t)) \tag{5.47}$$

Compare this result to Table 5.1. The error covariances and predicted estimates follow as in the linear case. This completes the derivation. Next let us consider a discrete version of the nonlinear system example given in Jazwinski [1].

Example 5.2

Consider the discrete nonlinear process given by

$$x(t) = (1 - 0.05\Delta T)x(t-1) + 0.04\Delta T x^2(t-1) + w(t-1)$$

with corresponding measurement model

$$y(t) = x^2(t) + x^3(t) + v(t)$$

where $v(t) \sim \mathcal{N}(0, 0.09)$, $x(0) = 2.3$, $P(0) = 0.01$, $\Delta T = 0.01$ sec and $R_{ww} = 0$. The simulated measurement using *SSPACK_PC* [8] is shown in Fig. 5.5c. The *LZ-BP* is designed from the following Jacobians:

$$A[x(t-1)] = 1 - 0.05\Delta T + 0.08\Delta T x(t-1) \quad \text{and} \quad C[x(t)] = 2x(t) + 3x^2(t)$$

Observing the mean state, we develop a reference trajectory by fitting a line to the simulated state which is given by

$$x^*(t) = 0.067\, t + 2.0 \qquad 0 \le t \le 1.5 \text{ and } u^*(t) = u(t) = 0.0 \ \forall t$$

The *LZ-BP* algorithm is then given by

$$\begin{aligned}
\hat{x}(t|t-1) &= (1 - 0.05\Delta T)x^*(t-1) \\
&\quad + (1 - 0.05\Delta T + 0.08\Delta T x^*(t-1))[\hat{x}(t-1|t-1) - x^*(t-1)] \\
\tilde{P}(t|t-1) &= [1 - 0.05\Delta T + 0.08\Delta T x^*(t-1)]^2 \tilde{P}(t-1|t-1) \\
e(t) &= y(t) - (x^{*2}(t) - x^{*3}(t)) - (2x^*(t) + 3x^{*2}(t))[\hat{x}(t|t-1) - x^*(t)] \\
R_{ee}(t) &= [2\hat{x}(t|t-1) + 3\hat{x}^2(t|t-1)]^2 \tilde{P}(t|t-1) + 0.09 \\
K(t) &= \frac{\tilde{P}(t|t-1)[2x^*(t) + 3x^{*2}(t)]}{R_{ee}(t)} \\
\hat{x}(t|t) &= \hat{x}(t|t-1) + K(t)e(t)
\end{aligned}$$

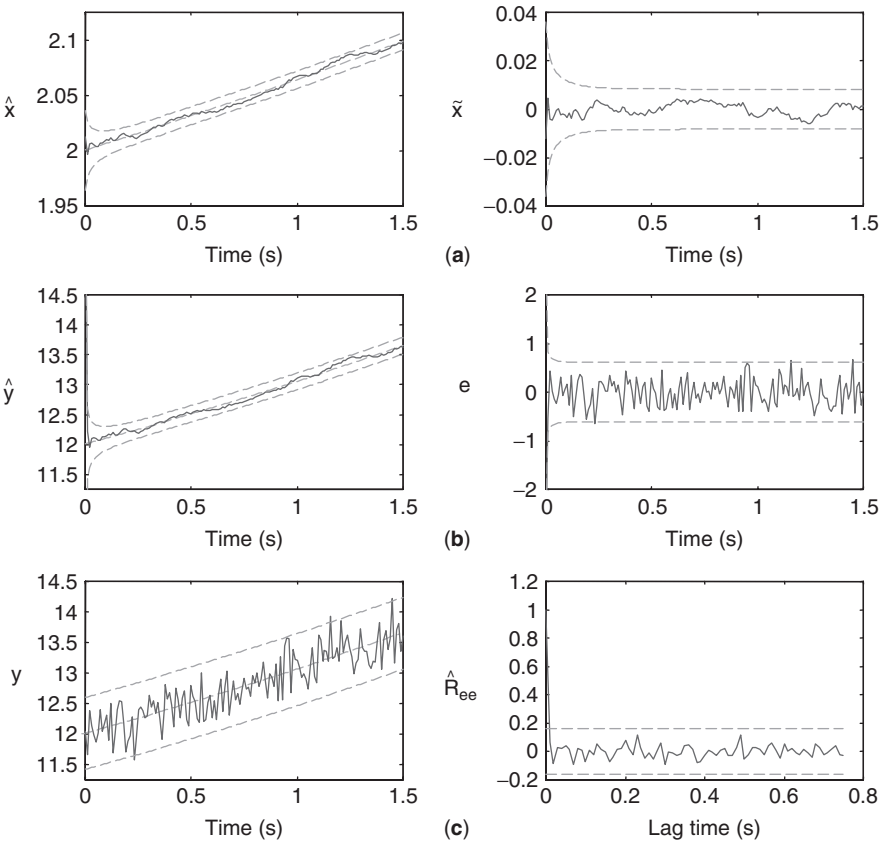


FIGURE 5.5 Linearized *BP* simulation. (a) Estimated state (0% out) and error (0% out). (b) Filtered measurement (1% out) and error (innovation) (2.6% out). (c) Simulated measurement and zero-mean/whiteness test ($3.9 \times 10^{-2} < 10.1 \times 10^{-2}$ and 0% out).

$$\tilde{P}(t|t) = (1 - K(t)[2x^*(t) + 3x^{*2}(t)])\tilde{P}(t|t-1)$$
$$\hat{x}(0|0) = 2.3 \quad \text{and} \quad \tilde{P}(0|0) = 0.01$$

A *LZ-BP* run is depicted in Fig. 5.5. 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 the filter is performing properly for this realization. The filtered measurement and innovations are shown in Fig. 5.5b 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 Fig. 5.5c indicating proper tuning. This completes the nonlinear filtering example. △△△

5.5 EXTENDED BAYESIAN PROCESSOR (EXTENDED KALMAN FILTER)

In this section we develop the extended Bayesian processor (*XBP*) or equivalently the extended Kalman filter (*EKF*). The *XBP* is *ad hoc* in nature, but has become one of the workhorses of (approximate) nonlinear filtering [1–11]. It has found applicability in a wide variety of applications such as tracking [15], navigation [1, 5], chemical processing [17], ocean acoustics [18], seismology [19] (see [20] for a detailed list). The *XBP* 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 makes the processor *ad hoc*. We must realize that the Jacobians used in the linearization process are deterministic (but time-varying), when a reference or perturbation trajectory is used. However, using the current state estimate is an approximation to the conditional mean, which is random, making these associated Jacobians and subsequent relations random. Therefore, although popularly ignored, most *XBP* 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-BP*. 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 *XBP* algorithm results. The reason for choosing to linearize about this estimate is that it represents the best information we have about the state and therefore most likely results in a better reference trajectory (state estimate). 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 α is $t-1$ or t, to linearize about instead of the reference trajectory $x^*(t)$, then the *XBP* evolves. That is, let

$$x^*(t) \equiv \hat{x}(t|\alpha) \quad \text{for } t-1 \leq \alpha \leq t \tag{5.48}$$

Then, for instance, when $\alpha = t-1$, the predicted perturbation is

$$\delta\hat{x}(t|t-1) = \hat{x}(t|t-1) - x^*(t)|_{x^*=\hat{x}(t|t-1)} = 0 \tag{5.49}$$

Thus, 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 update into the *LZ-BP* algorithm, it is easy to see that each of the difference terms $[\hat{x} - x^*]$ are null resulting in the *XBP* algorithm. That is, examining the *prediction* phase of the linearized algorithm, substituting the current available updated 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 of the *XBP*

$$\hat{x}(t|t-1) = a[\hat{x}(t-1|t-1)] + b[u(t-1)] \tag{5.50}$$

Now with the predicted estimate available, substituting it for the reference in Eq. 5.33, gives the innovation sequence as

$$\begin{aligned} e(t) &= y(t) - c[\hat{x}(t|t-1)] - C[\hat{x}(t|t-1)] \\ &\quad \times [\hat{x}(t|t-1) - \hat{x}(t|t-1)] = y(t) - c[\hat{x}(t|t-1)] \end{aligned} \tag{5.51}$$

where we have the new predicted or filtered measurement expression

$$\hat{y}(t|t-1) \equiv c[\hat{x}(t|t-1)] \tag{5.52}$$

The updated state estimate is easily obtained by substituting the predicted estimate for the reference ($\hat{x}(t|t-1) \rightarrow x^*(t)$) in Eq. 5.34

$$\begin{aligned} \delta\hat{x}(t|t) &= \delta\hat{x}(t|t-1) + K(t)e(t) \\ [\hat{x}(t|t) - \hat{x}(t|t-1)] &= [\hat{x}(t|t-1) - \hat{x}(t|t-1)] + K(t)e(t) \\ \hat{x}(t|t) &= \hat{x}(t|t-1) + K(t)e(t) \end{aligned} \tag{5.53}$$

The covariance and gain equations are identical to those in Table 5.2, 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 *XBP* or equivalently the *EKF* algorithm summarized in Table 5.3. Note that the covariance matrices, $\tilde{P}$, and the gain, K, are now functions of

TABLE 5.3 Extended *BP* (Kalman Filter) Algorithm

Prediction

$\hat{x}(t|t-1) = a[\hat{x}(t-1|t-1)] + b[u(t-1)]$ (state prediction)

$\tilde{P}(t|t-1) = A[\hat{x}(t|t-1)]\tilde{P}(t-1|t-1)A'[\hat{x}(t|t-1)] + R_{ww}(t-1)$ (covariance prediction)

Innovation

$e(t) = y(t) - c[\hat{x}(t|t-1)]$ (innovation)

$R_{ee}(t) = C[\hat{x}(t|t-1)]\tilde{P}(t|t-1)C'[\hat{x}(t|t-1)] + R_{vv}(t)$ (innovation covariance)

Gain

$K(t) = \tilde{P}(t|t-1)C'[\hat{x}(t|t-1)]R_{ee}^{-1}(t)$ (gain or weight)

Update

$\hat{x}(t|t) = \hat{x}(t|t-1) + K(t)e(t)$ (state update)

$\tilde{P}(t|t) = [I - K(t)C[\hat{x}(t|t-1)]]\tilde{P}(t|t-1)$ (covariance update)

Initial Conditions

$\hat{x}(0|0) \quad \tilde{P}(0|0)$

Jacobians

$$A[x(t|t-1)] \equiv \left.\frac{da[x(t-1)]}{d\hat{x}(t-1)}\right|_{x=\hat{x}(t|t-1)} \qquad C[\hat{x}(t|t-1)] \equiv \left.\frac{dc[x(t)]}{dx(t)}\right|_{x=\hat{x}(t|t-1)}$$

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 which should be averaged over the entire ensemble using this approach to get an "expected" state, etc. Note also in practice that this algorithm is usually implemented using sequential processing and *UD* (upper diagonal/square root) factorization techniques (see [21]).

The *XBP* can also be developed under (approximate) Gaussian assumptions using the *Bayesian approach* as before in the linear case. We briefly outline the derivation.

The *a posteriori probability* is given by

$$\Pr(x(t)|Y_t) = \frac{\Pr(y(t)|x(t)) \times \Pr(x(t)|Y_{t-1})}{\Pr(y(t)|Y_{t-1})} \tag{5.54}$$

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:

1. $\Pr(y(t)|x(t))$: $\quad \mathcal{N}(c[x(t)], R_{vv}(t))$
2. $\Pr(x(t)|Y_{t-1})$: $\quad \mathcal{N}(\hat{x}(t|t-1), \tilde{P}(t|t-1))$
3. $\Pr(y(t)|Y_{t-1})$: $\quad \mathcal{N}(\hat{y}(t|t-1), R_{ee}(t))$

Using the nonlinear models developed earlier, substituting the Gaussian probabilities and taking logarithms, we obtain the logarithmic *a posteriori* probability as

$$\begin{aligned} \ln \Pr(x(t)|Y_t) = \ln \kappa - \frac{1}{2} v'(t) R_{vv}^{-1}(t) v(t) - \frac{1}{2} \tilde{x}'(t|t-1) \tilde{P}^{-1}(t|t-1) \tilde{x}(t|t-1) \\ + \frac{1}{2} e'(t) R_{ee}^{-1}(t) e(t) \end{aligned} \tag{5.55}$$

The *MAP* estimate is then obtained by differentiating Eq. 5.55, setting it to zero and solving; that is,

$$\nabla_x \ln \Pr(x(t)|Y_t)|_{x=\hat{X}_{\text{map}}} = 0 \tag{5.56}$$

Applying the *chain rule* and the gradient operation (see Chapter 3), we obtain the expression. Note that the last term of Eq. 5.55 is not a function of $x(t)$, but just the data.

$$\begin{aligned} \nabla_x \ln \Pr(x(t)|Y_t) = \nabla_x c'[x(t)] R_{vv}^{-1}(t)(y(t) - c[x(t)]) \\ - \tilde{P}^{-1}(t|t-1)(x(t) - \hat{x}(t|t-1)) \end{aligned} \tag{5.57}$$

Using a first order Taylor series approximation for $c[x(t)]$ linearized about the predicted estimate $\hat{x}(t|t-1)$ and the usual definition for the Jacobian matrix $C[\hat{x}(t|t-1)]$

we have

$$\nabla_x \ln \Pr(x(t)|Y_t) = C'[\hat{x}(t|t-1)]R_{vv}^{-1}(t)[y(t) - c[\hat{x}(t|t-1)] - C[\hat{x}(t|t-1)](x(t) - \hat{x}(t|t-1))] - \tilde{P}^{-1}(t|t-1)[x(t) - \hat{x}(t|t-1)] \quad (5.58)$$

Identifying the innovation vector and grouping like-terms gives the expression

$$\begin{aligned}\nabla_x \ln \Pr(x(t)|Y_t) \;=\; & C'[\hat{x}(t|t-1)]R_{vv}^{-1}(t)e(t) \\ & - [C'[\hat{x}(t|t-1)]\, R_{vv}^{-1}(t)\, C[\hat{x}(t|t-1)] + \tilde{P}(t|t-1)]\, x(t) \\ & + [C'[\hat{x}(t|t-1)]\, R_{vv}^{-1}(t)\, C[\hat{x}(t|t-1)] + \tilde{P}(t|t-1)]\, \hat{x}(t|t-1)\end{aligned}$$

Setting this equation to zero and solving for $x(t) = \hat{X}_{\text{map}}(t)$ gives

$$\begin{aligned}\hat{X}_{\text{map}}(t) \;=\; & [C'[\hat{x}(t|t-1)]R_{vv}^{-1}(t)C[\hat{x}(t|t-1)] + \tilde{P}(t|t-1)]^{-1} \\ & \times [C'[\hat{x}(t|t-1)]R_{vv}^{-1}(t)C[\hat{x}(t|t-1)] + \tilde{P}(t|t-1)]\, \hat{x}(t|t-1) \\ & + [C'[\hat{x}(t|t-1)]R_{vv}^{-1}(t)C[\hat{x}(t|t-1)] \\ & \quad + \tilde{P}(t|t-1)]^{-1} C'[\hat{x}(t|t-1)]R_{vv}^{-1}(t)e(t)\end{aligned} \quad (5.59)$$

or

$$\begin{aligned}\hat{X}_{\text{map}}(t) = \hat{x}(t|t-1) + [C'[\hat{x}(t|t-1)]R_{vv}^{-1}(t)C[\hat{x}(t|t-1)] \\ + \tilde{P}(t|t-1)]^{-1} C'[\hat{x}(t|t-1)]R_{vv}^{-1}(t)e(t)\end{aligned} \quad (5.60)$$

Recognizing the similarity of this expression to the linear case with $C[\hat{x}(t|t-1)] \to C(t)$ and using the *matrix inversion lemma*, it is easy to show (see Eqns. 5.13–5.17) that our approximate updated error covariance satisfies

$$\tilde{P}(t|t) = [C'[\hat{x}(t|t-1)]R_{vv}^{-1}(t)C[\hat{x}(t|t-1)] + \tilde{P}^{-1}(t|t-1)]^{-1} \quad (5.61)$$

and therefore Eq. 5.60 can be simplified to give

$$\hat{X}_{\text{map}}(t) = \hat{x}(t|t-1) + \tilde{P}(t|t)C'[\hat{x}(t|t-1)]R_{vv}^{-1}(t)e(t) \quad (5.62)$$

Now recognizing the alternate form for the gain as in Eq. 5.21 gives the desired updated estimate as

$$\hat{x}(t|t) = \hat{X}_{\text{map}}(t) = \hat{x}(t|t-1) + K(t)e(t) \quad (5.63)$$

This completes the derivation. Consider the following discrete nonlinear example of the previous section.

Example 5.3

Consider the discrete nonlinear process and measurement system described in the previous example. The simulated measurement using *SSPACK_PC* [8] is shown in Fig. 5.6c. The *XBP* is designed from the following Jacobian:

$$A[x(t-1)] = 1 - 0.05\Delta T + 0.08\Delta T x(t-1) \quad \text{and} \quad C[x(t)] = 2x(t) + 3x^2(t)$$

The *XBP* algorithm is then given by

$$\begin{aligned}
\hat{x}(t|t-1) &= (1 - 0.05\Delta T)\hat{x}(t-1|t-1) + 0.04\Delta T\hat{x}^2(t-1|t-1) \\
\tilde{P}(t|t-1) &= [1 - 0.05\Delta T + 0.08\Delta T x(t-1)]^2\tilde{P}(t-1|t-1) \\
e(t) &= y(t) - \hat{x}^2(t|t-1) - \hat{x}^3(t|t-1) \\
R_{ee}(t) &= [2\hat{x}(t|t-1) + 3\hat{x}^2(t|t-1)]^2\tilde{P}(t|t-1) + 0.09 \\
K(t) &= (\tilde{P}(t|t-1)[2\hat{x}(t|t-1) + 3\hat{x}^2(t|t-1)])R_{ee}^{-1}(t) \\
\hat{x}(t|t) &= \hat{x}(t|t-1) + K(t)e(t) \\
\tilde{P}(t|t) &= (1 - K(t)[2\hat{x}(t|t-1) + 3\hat{x}^2(t|t-1)])\tilde{P}(t|t-1) \\
\hat{x}(0|0) &= 2.3 \quad \text{and} \quad \tilde{P}(0|0) = 0.01
\end{aligned}$$

A *XBP* run is depicted in Fig. 5.6. 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 Fig. 5.6b 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 Fig. 5.6c indicating proper tuning. Comparing the *XBP* to the *LZ-BP* of the previous section shows that it performs slightly worse in terms of 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 example. △△△

Next we consider one of the most popular applications of *XBP* approach—the *tracking problem* [1, 5, 15, 20]. The choice of the coordinate system for the tracker determines whether the nonlinearities occur either in the state equations (polar coordinates [22]) or in the measurement equations (Cartesian coordinates [23]). The following application depicts typical *XBP* performance in Cartesian coordinates.

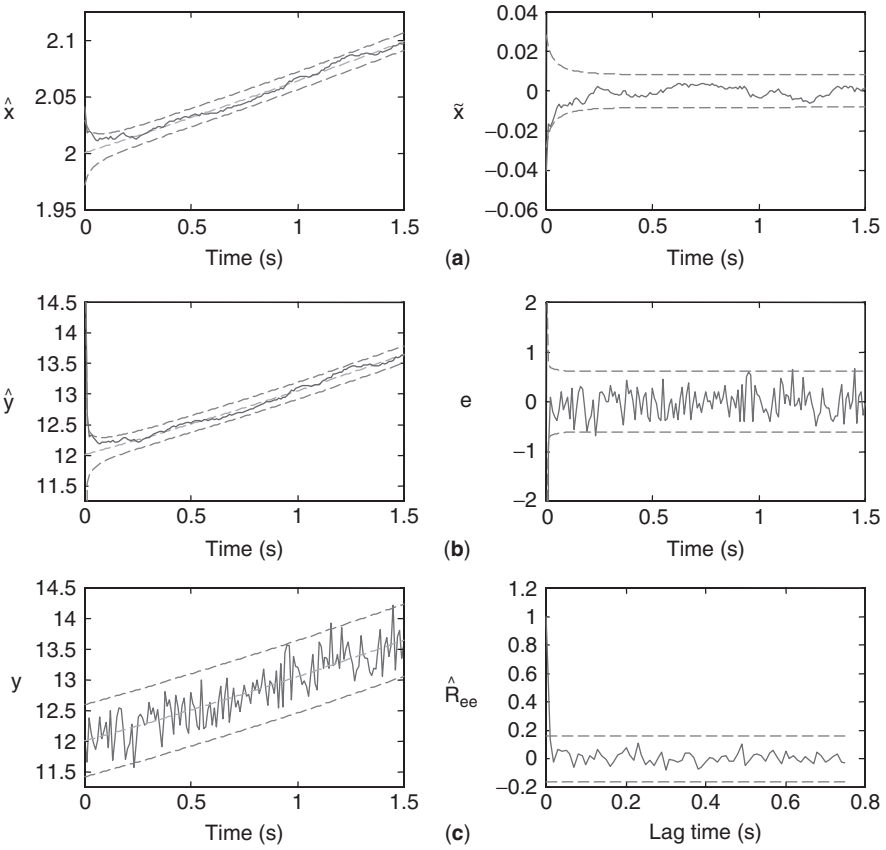


FIGURE 5.6 XBP (EKF) simulation. (a) Estimated state (1% out) and error (1% out). (b) Filtered measurement (1.3% out) and error (innovation) (3.3% out). (c) Simulated measurement and zero-mean/whiteness test ($6.3 \times 10^{-2} < 11.8 \times 10^{-2}$ and 0% out).

Example 5.4

Consider the following passive localization and tracking problem that frequently arises in sonar and navigation applications [15]. A maneuvering observer O monitors 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. 5.7. The velocity and position of the target relative to the observer are defined by

$$v_x(t) := v_{tx}(t) - v_{ox}(t) \qquad r_x(t) := r_{tx}(t) - r_{ox}(t)$$
$$v_y(t) := v_{ty}(t) - v_{oy}(t) \qquad r_y(t) := r_{ty}(t) - r_{oy}(t)$$

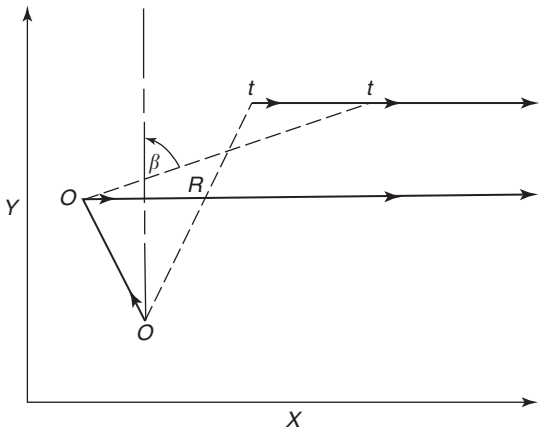


FIGURE 5.7 Observer/target ground track geometry for the XBP tracking application.

The velocity is related to position by

$$v(t) = \frac{d}{dt}r(t) \approx \frac{r(t) - r(t-1)}{\Delta T} \quad \text{for } \Delta T \text{ the sampling interval}$$

or

$$r(t) = r(t-1) + \Delta T v(t-1)$$

and

$$\begin{aligned} v(t) &= v(t-1) + [v(t) - v(t-1)] \\ v(t) &= [v_t(t-1) - v_o(t-1)] - [v_o(t) - v_o(t-1)] = v(t-1) - \Delta v_o(t-1) \end{aligned}$$

for a constant velocity target $v_t(t) = v_t(t-1) = \cdots = v_t$ and Δv 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 $x' := [\, r_x \;\; r_y \;\; v_x \;\; v_y \,]$ and input $u' := [-\Delta v_{ox} - \Delta v_{oy}]$, 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

$$x(t) = \begin{bmatrix} 1 & 0 & \Delta T & 0 \\ 0 & 1 & 0 & \Delta T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x(t-1) + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -\Delta v_{ox}(t-1) \\ \\ -\Delta v_{oy}(t-1) \end{bmatrix} + w(t-1)$$

with the nonlinear sensor 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 Fig. 5.8 for two legs of a tracking scenario. An impulse-incremental step change ($\Delta v_{ox} = -24$ knots and $\Delta v_{oy} = +10$ knots) was initiated at 0.5 h, resulting in a change of observer position and velocity depicted in the figure. The simulated bearing measurements are shown in Fig. 5.6d, The initial conditions for the run were $x'(0) := [0 \ 15\,\text{nm} \ 20\,\text{k} \ -10\,\text{k}]$ 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 $\Delta T = 0.33$ h.

The *XBP* algorithm of Table 5.3 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 Fig. 5.8. In a and b we see the respective x and y position estimates (velocity estimates are not shown) and corresponding tracking errors. Here we see that it takes approximately 1.8 h for the estimator to converge to the true target position (within ± 1 nm). The innovations sequence appears statistically zero-mean and white in Fig. 5.8c and d, indicating satisfactory performance. The filtered measurement in c is improved considerably over the unprocessed data in d. This completes the example. △△△

This completes the section on the extension of the *BP* to nonlinear problems. Next we investigate variants of the *XBP* for improved performance.

5.6 ITERATED-EXTENDED BAYESIAN PROCESSOR (ITERATED-EXTENDED KALMAN FILTER)

In this section we discuss an extension of the *XBP* of the previous section to the iterated-extended (*IX-BP*). We heuristically motivate the design and then discuss a

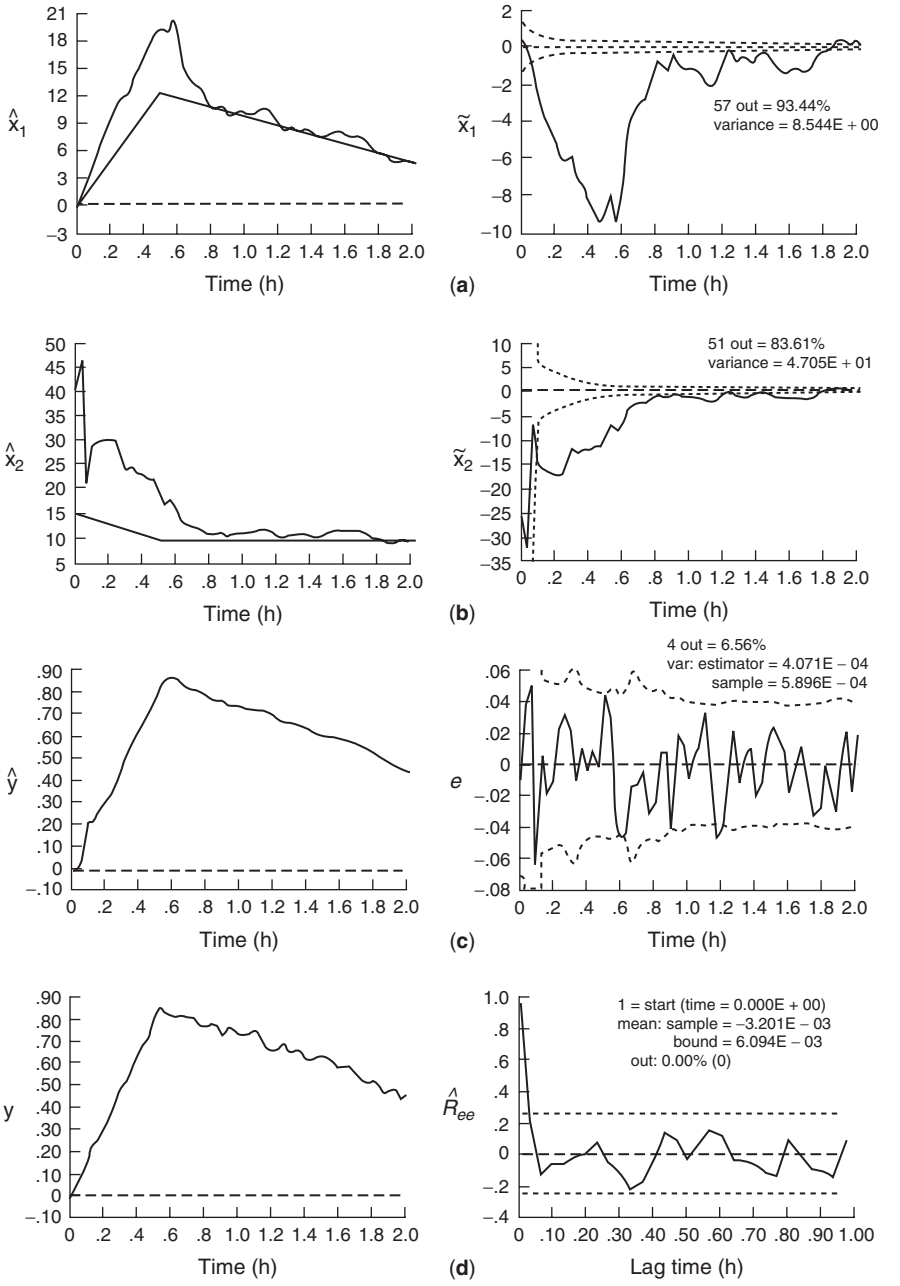


FIGURE 5.8 Extended *BP* (EKF) simulation for bearings-only tracking problem. (a) X-position estimate and error. (b) Y-position estimate and error. (c) Filtered measurement and error (innovation). (d) Simulated measurement and zero-mean/whiteness test ($3.2 \times 10^{-3} < 6.1 \times 10^{-3}$ and 0% out).

more detailed approach using Bayesian *MAP* estimation coupled with numerical optimization techniques to develop the processor. 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 as the new measurements are made available.

To develop the iterated-extended processor, we start with the linearized processor update relation substituting the "linearized" innovation of Eq. 5.33 of the *LZ-BP*, that is,

$$\hat{x}(t|t) = \hat{x}(t|t-1) + K(t;x^*(t))\,[y(t) - c[x^*(t)] - C[x^*(t)]\,(\hat{x}(t|t-1) - x^*(t))] \quad (5.64)$$

where we have explicitly shown the dependence of the gain (through the measurement Jacobian) on the reference trajectory, $x^*(t)$. The *XBP* algorithm linearizes about the most currently available estimate, $x^*(t) = \hat{x}(t|t-1)$ in this case. Theoretically, the updated estimate, $\hat{x}(t|t)$ is a better estimate and closer to the true trajectory. Suppose we continue and re-linearize about $\hat{x}(t|t)$ when it becomes available and then recompute the corrected estimate and so on. That is, define the $(i+1)^{th}$-iterated estimate as $\hat{x}_{i+1}(t|t)$, then the updated *iterator* equation becomes

$$\begin{aligned}\hat{x}_{i+1}(t|t) &= \hat{x}(t|t-1) + K(t;\hat{x}_i(t|t)) \\ &\quad \times [y(t) - c[\hat{x}_i(t|t)] - C[\hat{x}_i(t|t)](\hat{x}(t|t-1) - \hat{x}_i(t|t))] \end{aligned} \quad (5.65)$$

Now if we start with the 0^{th} iterate as the predicted estimate, that is, $\hat{x}_o \equiv \hat{x}(t|t-1)$, then the *XBP* results for $i = 0$. Clearly, the updated estimate in this iteration is given by

$$\hat{x}_1(t|t) = \hat{x}(t|t-1) + K_o(t)[y(t) - c[\hat{x}(t|t)] - C[\hat{x}(t|t-1)](\hat{x}(t|t-1) - \hat{x}(t|t-1))] \quad (5.66)$$

where the last term in this expression is null leaving the usual innovation. Note also that the *gain* is reevaluated on 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 updated estimate. The complete (updated) iterative loop is given by:

$$\begin{aligned} e_i(t) &= y(t) - c[\hat{x}_i(t|t)] \\ R_{e_i e_i}(t) &= C[\hat{x}_i(t|t)]\tilde{P}(t|t-1)C'[\hat{x}_i(t|t)] + R_{vv}(t) \\ K_i(t) &= \tilde{P}(t|t-1)C'[\hat{x}_i(t|t)]R^{-1}_{e_i e_i}(t) \\ \hat{x}_{i+1}(t|t) &= \hat{x}(t|t-1) + K_i(t)[e_i(t) - C[\hat{x}_i(t|t)](\hat{x}(t|t-1) - \hat{x}_i(t|t))] \\ \tilde{P}_i(t|t) &= (I - K_i(t)C[\hat{x}_i(t|t)])\tilde{P}(t|t-1) \end{aligned} \quad (5.67)$$

TABLE 5.4 Iterated Extended *BP* (Kalman Filter) Algorithm

Prediction

$\hat{x}(t|t-1) = a[\hat{x}(t-1|t-1)] - b[u(t-1)]$ (state prediction)

$\tilde{P}(t|t-1) = A[\hat{x}(t|t-1)]\tilde{P}(t-1|t-1)A'[\hat{x}(t|t-1)] - R_{ww}(t-1)$ (covariance prediction)

LOOP: $i = 1, \ldots, N_{iterations}$

Innovation

$e_i(t) = y(t) - c[\hat{x}_i(t|t)]$ (innovation)

$R_{e_ie_i}(t) = C[\hat{x}_i(t|t)]\tilde{P}(t|t-1)C'[\hat{x}_i(t|t)] - R_{vv}(t)$ (innovation covariance)

Gain

$K_i(t) = \tilde{P}(t|t-1)C'[\hat{x}_i(t|t)]R^{-1}_{e_ie_i}(t)$ (gain or weight)

Update

$\hat{x}_{i+1}(t|t) = \hat{x}(t|t-1) + K_i(t)[e_i(t) - C[\hat{x}_i(t|t)](\hat{x}(t|t-1) - \hat{x}_i(t|t))]$ (state update)

$\tilde{P}_i(t|t) = [I - K_i(t)C[\hat{x}_i(t|t)]]\tilde{P}(t|t-1)$ (covariance update)

Initial Conditions

$\hat{x}(0|0), \quad \tilde{P}(0|0), \quad \hat{x}_o(t|t) = \hat{x}(t|t-1)$

Jacobians

$$A[\hat{x}(t|t-1)] \equiv \left.\frac{da[x(t-1)]}{dx(t-1)}\right|_{x=\hat{x}(t|t-1)} \qquad C[\hat{x}_i(t|t)] \equiv \left.\frac{dc[x(t)]}{dx(t)}\right|_{x=\hat{x}_i(t|t)}$$

A typical stopping rule is:

$$\| \hat{x}_{i+1}(t|t) - \hat{x}_i(t|t) \| < \epsilon \quad \text{and} \quad \hat{x}_i(t|t) \to \hat{x}(t|t) \tag{5.68}$$

The *IX-BP* algorithm is summarized in Table 5.4.

The *IX-BP* can be useful in reducing the measurement function nonlinearity approximation errors 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-BP* from the Bayesian perspective, that is, we first formulate a parametric optimization problem to develop the generic structure of the iterator, then apply it to the underlying state estimation problem. Let us assume that we have a nonlinear cost function, $J(\Theta)$, 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 the Θ_i, that is,

$$J(\Theta) = J(\Theta_i) + (\Theta - \Theta_i)'[\nabla_\Theta J(\Theta_i)] + \frac{1}{2}(\Theta - \Theta_i)'[\nabla_{\Theta\Theta} J(\Theta_i)](\Theta - \Theta_i) + \text{H.O.T.} \tag{5.69}$$

where ∇_Θ is the N_Θ-gradient vector defined by

$$\nabla_\Theta J(\Theta_i) := \frac{\partial J(\Theta)}{\partial \Theta}\Big|_{\Theta=\Theta_i} \tag{5.70}$$

with the corresponding $N_\Theta \times N_\Theta$ *Hessian* matrix defined by

$$\nabla_{\Theta\Theta} J(\Theta_i) := \frac{\partial^2 J(\Theta)}{\partial \Theta^2}\Big|_{\Theta=\Theta_i} \tag{5.71}$$

Now if we approximate this expression by neglecting the H.O.T. and assume that Θ_i is close to the *true* parameter vector ($\Theta_i \approx \Theta_{\text{true}}$), then differentiating Eq. 5.69 using the chain rule, we obtain

$$\nabla_\Theta J(\Theta_i) = 0 + \nabla_\Theta \Theta'[\nabla_\Theta J(\Theta_i)] + \frac{1}{2}(2[\nabla_{\Theta\Theta} J(\Theta_i)](\Theta - \Theta_i)) = 0$$

or

$$[\nabla_{\Theta\Theta} J(\Theta_i)](\Theta - \Theta_i) = -[\nabla_\Theta J(\Theta_i)]$$

Solving for Θ and letting $\Theta \to \Theta_{i+1}$ we obtain the well-known *Newton-Rhapson* iterator (*NRI*) as [1, 15, 16]

$$\Theta_{i+1} = \Theta_i - [\nabla_{\Theta\Theta} J(\Theta_i)]^{-1}[\nabla_\Theta J(\Theta_i)] \tag{5.72}$$

This is the form that our *IX-BP* will assume. Now let us 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

$$\Pr(x(t)|Y_t) = \frac{1}{\eta} \times \Pr(y(t)|x(t)) \times \Pr(x(t)|Y_{t-1}) \tag{5.73}$$

where η 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. $\Pr(x(t)|Y_{t-1})$: $\quad \mathcal{N}(\hat{x}(t|t-1), \tilde{P}(t|t-1))$

Maximizing the *a posteriori* probability is equivalent to minimizing its logarithm, therefore, we have that

$$\begin{aligned} J(x(t)) &= \ln \Pr(y(t)|x(t)) + \ln \Pr(x(t)|Y_{t-1}) \\ &= -\frac{1}{2}(y(t) - c[x(t)])' R_{vv}^{-1}(t)(y(t) - c[x(t)]) \\ &\quad -\frac{1}{2}(x(t) - \hat{x}(t|t-1))' \tilde{P}^{-1}(t|t-1)(x(t) - \hat{x}(t|t-1)) \end{aligned} \tag{5.74}$$

Minimizing this expression, we differentiate with respect to $x(t)$ using the gradient operator as before to obtain

$$\nabla_x J(x(t)) = [\nabla_x c'[x(t)]]R_{vv}^{-1}(t)(y(t)-c[x(t)])-\tilde{P}^{-1}(t|t-1)(x(t)-\hat{x}(t|t-1)) \quad (5.75)$$

With a slight abuse, we can simplify the notation by defining the relations

$$C[x_i(t)] := \frac{\partial c[x(t)]}{\partial x(t)}|_{x(t)=x_i(t)}, e_i(t) := y(t) - c[x_i(t)], \text{ and } \tilde{x}_i(t) := x_i(t) - \hat{x}(t|t-1)$$

Therefore, letting $x \to x_i$, we can write Eq. 5.75 as

$$\nabla_x J(x_i(t)) = C'[x_i(t)]R_{vv}^{-1}(t)e_i(t) - \tilde{P}^{-1}(t|t-1)\tilde{x}_i(t|t-1) \quad (5.76)$$

which is the same form of the linear *MAP* estimator of Eq. 5.10 with $C[x_i(t)] \to C(t)$.

Continuing with the *NRI* derivation, we differentiate Eq. 5.76 again to obtain the Hessian

$$\nabla_{xx} J(x_i(t)) = C'[x_i(t)]R_{vv}^{-1}(t)C[x_i(t)] + \tilde{P}^{-1}(t|t-1) \quad (5.77)$$

but applying the *matrix inversion lemma* as in Sec. 5.4 (see Eq. 5.13 for details), it is shown that

$$[\nabla_{xx} J(x_i(t))]^{-1} = (I - K_i(t)C[x_i(t)])\tilde{P}(t|t-1) \equiv \tilde{P}_i(t|t) \quad (5.78)$$

for

$$K_i(t) := K_i(t; x_i(t)) = \tilde{P}(t|t-1)C'[x_i(t)]R_{e_ie_i}^{-1}(t) \quad (5.79)$$

with

$$R_{e_ie_i}(t) = C[x_i(t)]\tilde{P}(t|t-1)C'[x_i(t)] + R_{vv}(t) \quad (5.80)$$

Now equating Eqs. 5.76 and 5.78 and solving for $\tilde{P}^{-1}(t|t-1)$, we obtain

$$\tilde{P}^{-1}(t|t-1) = \tilde{P}^{-1}(t|t) - C'[x_i(t)]R_{vv}^{-1}(t)C[x_i(t)] \quad (5.81)$$

which can be substituted back into Eq. 5.76 to give

$$\nabla_x J(x_i(t)) = C'[x_i(t)]R_{vv}^{-1}(t)e_i(t) - [\tilde{P}^{-1}(t|t) - C'[x_i(t)]R_{vv}^{-1}(t)C[x_i(t)]]\tilde{x}_i(t|t-1) \quad (5.82)$$

The *NRI* can now be written in terms of the *iterated state* at time t as

$$x_{i+1}(t) = x_i(t) - [\nabla_{xx} J(x_i(t))]^{-1}\nabla_x J(x_i(t)) \quad (5.83)$$

Using the expressions for the Hessian and gradient, we have

$$\begin{aligned} x_{i+1}(t) \;=\; & x_i(t) + \tilde{P}_i(t|t) \\ & \times(C'[x_i(t)]R_{vv}^{-1}(t)e_i(t) - [\tilde{P}^{-1}(t|t) - C'[x_i(t)]R_{vv}^{-1}(t)C[x_i(t)]]\tilde{x}_i(t|t-1)) \end{aligned} \quad (5.84)$$

Multiplying through by the Hessian and recognizing the alternate expression for the gain (see Eq. 5.22), we obtain the expression

$$x_{i+1}(t) = x_i(t) + K_i(t)e_i(t) - (I - K_i(t)C[x_i(t)])\tilde{x}_i(t|t-1) \tag{5.85}$$

Now multiplying through by $\tilde{x}(t|t-1)$, factoring out the gain and performing the additions, we have

$$x_{i+1}(t) = \hat{x}(t|t-1) + K_i(t)[e_i(t) - C[x_i(t)](\hat{x}(t|t-1) - x_i(t))] \tag{5.86}$$

Defining the iterate in terms of the corrected state estimate, that is, $x_i(t) \to \hat{x}_i(t|t)$ gives the *NRI* iterator of Eq. 5.67 and Table 5.4.

So we see that for strong measurement nonlinearities the *IX-BP* can be used for little cost to the *XBP*. A further extension of these results is called the *iterator-smoother XBP* in which the entire processor is iterated to mitigate strong nonlinearities in the predicted estimates [1]. Here the measurement is relinearized and then a "smoothed" state estimate is calculated and used in the prediction loop. This completes the discussion of the processor, next we demonstrate its performance.

Example 5.5

Consider the discrete nonlinear process and measurement system described in the previous example. The simulated measurement using *SSPACK_PC* [8] is shown in Fig. 5.9c. The *IX-BP* is designed from the following Jacobian:

$$A[x(t-1)] = 1 - 0.05\Delta T + 0.08\Delta T x(t-1) \quad \text{and} \quad C[x(t)] = 2x(t) + 3x^2(t)$$

The *IX-BP* algorithm is then given by

$$\begin{aligned}
\hat{x}(t|t-1) &= (1 - 0.05\Delta T)\hat{x}(t-1|t-1) + 0.04\Delta T\hat{x}^2(t-1|t-1) \\
\tilde{P}(t|t-1) &= [1 - 0.05\Delta T + 0.08\Delta T x(t-1)]^2\tilde{P}(t-1|t-1) \\
\\
e_i(t) &= y(t) - \hat{x}_i^2(t|t) - \hat{x}_i^3(t|t) \\
R_{e_ie_i}(t) &= [2\hat{x}_i(t|t) + 3\hat{x}_i^2(t|t)]^2\tilde{P}(t|t-1) + 0.09 \\
\\
K_i(t) &= \tilde{P}(t|t-1)[2\hat{x}_i(t|t) + 3\hat{x}_i^2(t|t)]/R_{e_ie_i}(t) \\
\\
\hat{x}_{i+1}(t|t) &= \hat{x}(t|t-1) + K_i(t)[e_i(t) - [2\hat{x}_i(t|t) + 3\hat{x}_i^2(t|t)](\hat{x}(t|t-1) - \hat{x}_i(t|t)) \\
\tilde{P}_i(t|t) &= (1 - K_i(t)[2\hat{x}_i(t|t) + 3\hat{x}_i^2(t|t)])\tilde{P}(t|t-1) \\
\\
\hat{x}(0|0) &= 2.3 \quad \text{and} \quad \tilde{P}(0|0) = 0.01
\end{aligned}$$

A *IX-BP* run is depicted in Fig. 5.9. Here we see that the state estimate ($\sim$0% lie within limits) begins tracking the true state instantaneously. The estimation error is

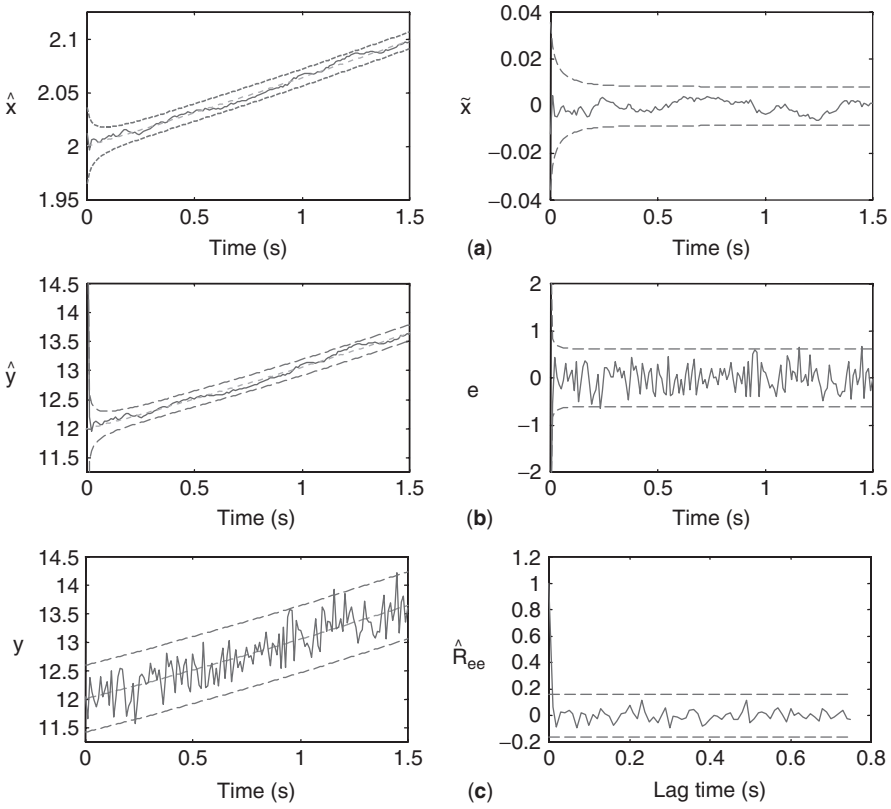


FIGURE 5.9 Iterated-extended *BP* (IEKF) simulation. (a) Estimated state (∼0% out) and error (∼0% out). (b) Filtered measurement (∼1% out) and error (innovation) (∼2.6% out). (c) Simulated measurement and zero-mean/whiteness test ($4 \times 10^{-2} < 10.7 \times 10^{-2}$ and 0% out).

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 Fig. 5.9b. The innovations are zero mean ($4 \times 10^{-2} < 10.7 \times 10^{-2}$) and white (0% lie outside limits) as shown in Fig. 5.9c indicating proper tuning and matching the *LZ-BP* 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-BP* 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-BP*. This completes the nonlinear filtering example. △△△

Next we consider some more practical approaches to designing estimators.

5.7 PRACTICAL ASPECTS OF CLASSICAL BAYESIAN PROCESSORS

In this section we heuristically provide an intuitive feel for the operation of the Bayesian processor using the state–space model and *GM* assumptions. These results coupled with the theoretical points developed in [10] lead to the proper adjustment or "tuning" of the *BP*. Tuning the processor is considered an art, but with proper statistical tests, the performance can readily be evaluated and adjusted. As mentioned previously, this approach is called the *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 *BP* is properly tuned. Once tuned, then all of the statistics (conditional means and variances) are valid and may be used as reasonable estimates. Here we discuss how the parameters can be adjusted and what statistical tests must be performed to evaluate *BP* performance.

Heuristically, the sequential *BP* can be viewed simply by its update equation

$$\hat{X}_{\text{new}} = \overbrace{\underbrace{\hat{X}_{\text{old}}}_{\text{State–space model}}}^{\text{Prediction}} + \overbrace{\underbrace{K \times E_{\text{new}}}_{\text{Measurement}}}^{\text{Update}}$$

where $\hat{X}_{\text{old}} \approx f(\text{model})$ and $E_{\text{new}} \approx f(\text{measurement})$.

Using this model of the *BP*, 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 5.1. Consider the new estimate under the following cases:

$$K \longrightarrow \text{small} \qquad \hat{X}_{\text{new}} = \hat{X}_{\text{old}} = f(\text{model})$$
$$K \longrightarrow \text{large} \qquad \hat{X}_{\text{new}} = KE_{\text{new}} = f(\text{measurement})$$

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* (Fig. 5.10).

Let us investigate the gain matrix and see if its variations are consistent with these heuristic notions. First, it was shown in Eq. 5.22 that the alternate form of the gain equation is given by

$$K(t) = \tilde{P}(t|t)C'(t)R_{vv}^{-1}(t)$$

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;

Condition	Gain	Parameter
Believe *model*	Small	$\widetilde{P}$ small (model adequate)
		R_{vv} large (measurement noisy)
Believe *measurement*	Large	R_{vv} small (measurement good)
		$\widetilde{P}$ large (model inadequate)

FIGURE 5.10 Bayesian processor heuristic notions.

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 *BP* algorithm of Table 5.1.

Summarizing, a *BP* (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 *BP*.

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 *BP* operation. A *necessary and sufficient condition* for a *BP* to be optimal is that the innovation sequence is zero-mean and white (see [4] for the proof). These are the first properties that must be evaluated to ensure 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

$$\hat{m}_e(i) = \frac{1}{N}\sum_{t=1}^{N} e_i(t) \quad \text{for } i = 1, \ldots, N_y \tag{5.87}$$

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 zero [10]. We test that the mean of the i^{th} component of the innovation vector $e_i(t)$ is

$$\begin{aligned} H_o: &\quad m_e(i) = 0 \\ H_1: &\quad m_e(i) \neq 0 \end{aligned}$$

As our test statistic we use the sample mean. At the α-significance level, the probability of rejecting the null hypothesis H_o is given by

$$\Pr\left(\left|\frac{\hat{m}_e(i) - m_e(i)}{\sqrt{R_{ee}(i)/N}} > \frac{\tau_i - m_e(i)}{\sqrt{R_{ee}(i)/N}}\right|\right) = \alpha \tag{5.88}$$

Therefore, the *zero-mean test* [10] on each component innovation e_i is given by

$$\hat{m}_e(i) \underset{\text{Accept } H_o}{\overset{\text{Reject } H_o}{\gtrless}} \tau_i \tag{5.89}$$

Under the null hypothesis H_o, each $m_e(i)$ is zero. Therefore, at the 5% significance level ($\alpha = 0.05$), we have that the threshold is

$$\tau_i = 1.96\sqrt{\frac{\hat{R}_{ee}(i)}{N}} \tag{5.90}$$

where $\hat{R}_{ee}(i)$ is the *sample variance* (assuming ergodicity) estimated by

$$\hat{R}_{ee}(i) = \frac{1}{N}\sum_{t=1}^{N} e_i^2(t) \tag{5.91}$$

Under the same assumptions, we can perform a *whiteness test* [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

$$\hat{R}_{ee}(i,k) = \frac{1}{N}\sum_{t=k+1}^{N} (e_i(t) - \hat{m}_e(i))(e_i(t+k) - \hat{m}_e(i)) \tag{5.92}$$

We actually use the *normalized covariance* test statistic

$$\hat{\rho}_{ee}(i,k) = \frac{\hat{R}_{ee}(i,k)}{\hat{R}_{ee}(i)} \tag{5.93}$$

Asymptotically for large N, it can be shown that (see [10–14]) that

$$\hat{\rho}_{ee}(i,k) \sim \mathcal{N}(0, 1/N)$$

Therefore, the 95% confidence interval estimate is

$$I_{\rho_{ee}} = \hat{\rho}_{ee}(i,k) \pm \frac{1.96}{\sqrt{N}} \quad \text{for } N > 30 \tag{5.94}$$

Hence, under the null hypothesis, 95% of the $\hat{\rho}_{ee}(i,k)$ values must lie within this confidence interval, 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. 5.94 is very useful for detecting model inaccuracies from individual component innovations. However, for complex systems with a large number of measurement channels, it becomes computationally burdensome to investigate each innovation component-wise. A statistic capturing *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 [10, 14]. The *WSSR* test statistic is given by

$$\hat{\rho}(\ell) := \sum_{k=\ell-N+1}^{\ell} e'(k) R_{ee}^{-1}(k) e(k) \quad \text{for } \ell \geq N \tag{5.95}$$

and is based on the hypothesis test

$$\begin{aligned} H_o: &\quad \rho(\ell) \ \textit{is} \text{ white} \\ H_1: &\quad \rho(\ell) \ \textit{is not} \text{ white} \end{aligned}$$

given by

$$\hat{\rho}(\ell) \begin{array}{l} > \text{Reject } H_o \\ \\ < \text{Accept } H_o \end{array} \tau \tag{5.96}$$

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 [4] for more details). At the α-significance level, the probability of rejecting the null hypothesis is given by

$$\Pr\left(\left| \frac{\rho(\ell) - N_y N}{\sqrt{2N_y N}} > \frac{\tau - N_y N}{\sqrt{2N_y N}} \right| \right) = \alpha \tag{5.97}$$

For a level of significance of $\alpha = 0.05$, we have

$$\tau = N_y N + 1.96\sqrt{2N_y N} \tag{5.98}$$

Thus, 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 *BP* algorithm directly, they can be used for both *stationary* as well as *nonstationary* processes. In fact, in practice for a large number of measurement

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	Whiteness	Gaussian
	$R_{ee}(t,k)$	Sample cross-covariance	Cross-covariance	Ergodic, gaussian
	$R_{eu}(t,k)$	Sample cross-covariance	Cross-covariance	Ergodic, gaussian
Covariances	Innovation	Sample variance	$R_{ee} = \hat{R}_{ee}$	Ergodic
	Innovation	R_{ee}	Confidence interval about $\{e(t)\}$	
	Estimation error	Sample variance	$\tilde{P} = \hat{P}$	Ergodic, X_{true} known
	Estimation error	$\tilde{P}$	Confidence interval about $\{\tilde{x}(t\|t)\}$	X_{true} known

FIGURE 5.11 State–space *BP* tuning tests.

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.

Other sets of "reasonableness" tests can be performed using the covariances estimated by the *BP* algorithm and sample variances estimated using Eq. 5.92. The *BP* 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, $\tilde{P}$ is constant, the estimates can be compared to ensure that they are reasonable. Thus we have

$$\hat{R}_{ee}(i) \approx R_{ee}(i) \quad \text{and} \quad \hat{\tilde{P}}(i) \approx \tilde{P}(i) \tag{5.99}$$

Plotting the $\pm 1.96\sqrt{R_{e_ie_i}(t)}$ and $\pm 1.96\sqrt{\tilde{P}_i(t|t)}$ 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 *BP* 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 Fig. 5.11 and examine an *RLC*-circuit design problem in the following section to demonstrate the approach in more detail.

5.8 CASE STUDY: RLC CIRCUIT PROBLEM

Consider the design of an estimator for a series *RLC* circuit (second-order system) excited by a pulse train. The circuit diagram is shown in Fig. 5.12. Using Kirchhoff's

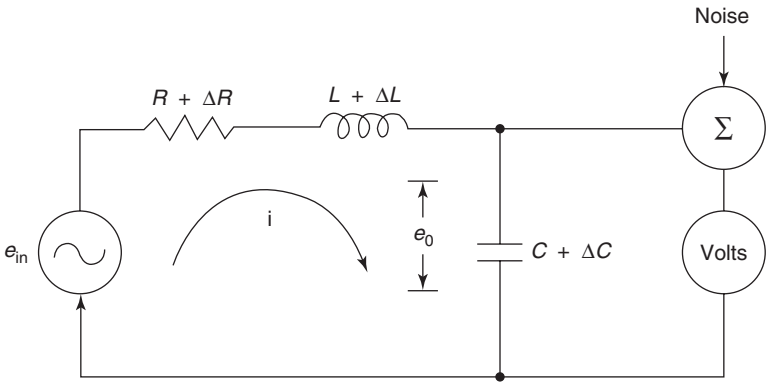


FIGURE 5.12 *RLC* circuit problem.

voltage law, we can obtain the circuit equations with $i = C(de/dt)$:

$$\frac{d^2e}{dt^2} + \frac{R}{L}\frac{de}{dt} + \frac{1}{LC}e = \frac{1}{LC}e_{in}$$

where e_{in} is a unit pulse train. This equation is that of a second-order system that characterizes the electrical *RLC* circuit, or a mechanical vibration system, or a hydraulic flow system, etc. The dynamic equations can be placed in state–space form by choosing $x := [e \mid de/dt]'$ and $u = e_{in}$:

$$\frac{dx}{dt} = \begin{bmatrix} 0 & 1 \\ -\frac{1}{LC} & -\frac{R}{L} \end{bmatrix} x + \begin{bmatrix} 0 \\ -\frac{1}{LC} \end{bmatrix} u + \begin{bmatrix} 0 \\ -\frac{1}{LC} \end{bmatrix} w$$

where $w \sim N(0, R_{ww})$ is used to model component inaccuracies.

A high-impedance voltmeter is placed in the circuit to measure the capacitor voltage e. We assume that it is a digital (sampled) device contaminated with noise of variance R_{vv}; that is,

$$y(t) = e(t) + v(t)$$

where $v \sim N(0, R_{vv})$. For our problem we have the following parameters: $R = 5\,\mathrm{K}\Omega, L = 2.5\,\mathrm{H}, C = 0.1\,\mu\mathrm{F}$, and $T = 0.1\,\mathrm{ms}$ (the problem will be scaled in milliseconds). We assume that the component inaccuracies can be modeled using $R_{ww} = 0.01$, characterizing a deviation of $\pm 0.1\,\mathrm{V}$ uncertainty in the circuit representation. Finally, we assume that the precision of the voltmeter measurements are $(e \pm 0.2\,\mathrm{V})$, the two standard deviation value, so that $R_{vv} = 0.01\,(\mathrm{V})^2$. Summarizing the circuit model, we have the continuous-time representation

$$\frac{dx}{dt} = \begin{bmatrix} 0 & 1 \\ -4 & -2 \end{bmatrix} x + \begin{bmatrix} 0 \\ -4 \end{bmatrix} u + \begin{bmatrix} 0 \\ -4 \end{bmatrix} w$$

and the discrete-time measurements

$$y(t) = [1 \quad 0]x(t) + v(t)$$

where

$$R_{ww} = \frac{(0.1)^2}{T} = 0.01(\text{V})^2 \quad \text{and} \quad R_{vv} = 0.01(\text{V})^2$$

Before we design the discrete Bayesian processor, we must convert the system or process model to a sampled-data (discrete) representation. Using *SSPACK_PC* [8], this is accomplished automatically with the Taylor series approach to approximating the matrix exponential. For an error tolerance of 1×10^{-12}, a 15-term series expansion yields the following discrete-time Gauss-Markov model:

$$x(t) = \begin{bmatrix} 0.98 & 0.09 \\ -0.36 & 0.801 \end{bmatrix} x(t-1) + \begin{bmatrix} -0.019 \\ -0.36 \end{bmatrix} u(t-1) + \begin{bmatrix} -0.019 \\ -0.36 \end{bmatrix} w(t-1)$$

$$y(t) = [1 \mid 0]x(t) + v(t)$$

where

$$R_{ww} = 0.01(\text{V})^2 \quad \text{and} \quad R_{vv} = 0.01(\text{V})^2$$

Using *SSPACK_PC* with initial conditions $x(0) = 0$ and $P = \text{diag}(0.01, 0.04)$, the simulated system is depicted in Fig. 5.13. In Fig. 5.13a–c we see the simulated states and measurements with corresponding confidence limits about the mean (true) values. In each case, the simulation satisfies the statistical properties of the *GM* model. The corresponding true (mean) trajectories are also shown along with the pulse train excitation. Note that the measurements are merely a noisier (process and measurement noise) version of the voltage x_1.

A discrete Bayesian processor was designed using *SSPACK_PC* to improve the estimated voltage $\hat{x}_1$. The results are shown in Fig. 5.14. In a through c we see the filtered states and measurements as well as the corresponding estimation errors. The true (mean) states are superimposed as well to indicate the tracking capability of the estimator. The estimation errors lie within the bounds (3 percent out) for the second state, but the error covariance is slightly underestimated for the first state (14 percent out). The predicted and sample variances are close ($0.002 \approx 0.004$ and $0.028 \approx 0.015$) in both cases. The innovations lie within the bounds (3 percent out) with the predicted sample variances close ($0.011 \approx 0.013$). The innovations are statistically zero-mean ($0.0046 \ll 0.014$) and white (5 percent out, *WSSR* below threshold),[4] indicating a well-tuned estimator. This completes the *RLC* problem.

[4] *WSSR* is the weighted-sum squared residual statistic which aggregates the innovation vector information over a window to perform a vector-type whiteness test (see [2] for details).

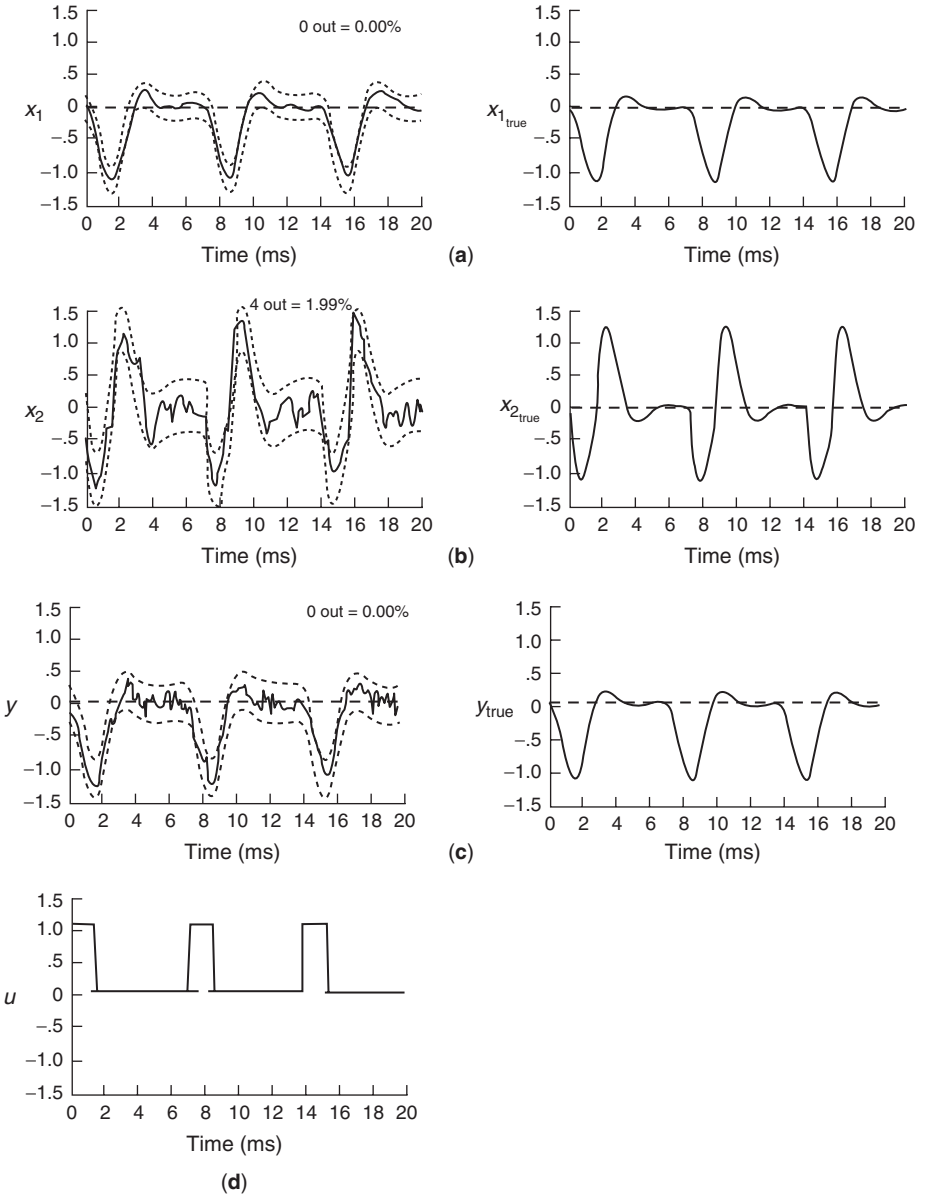


FIGURE 5.13 Gauss-Markov simulation of *RLC* circuit problem. (a) Simulated and true state (voltage). (b) Simulated true state (current). (c) Simulated and true measurement. (d) Pulse-train excitation.

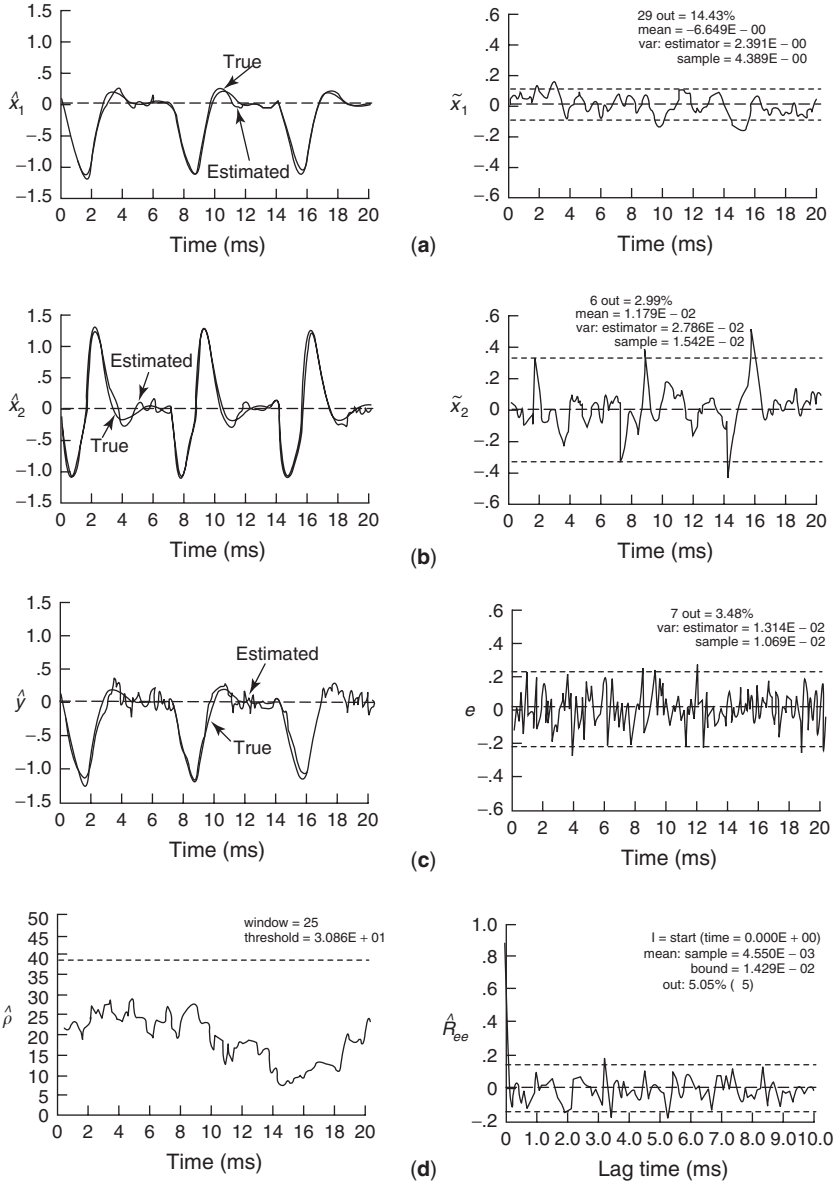


FIGURE 5.14 Bayesian processor design for *RLC* circuit problem. (a) Estimated state (voltage) and error. (b) Estimated state (current) and error. (c) Filtered and true measurement (voltage) and error (innovation). (d) *WSSR* and whiteness test.

5.9 SUMMARY

In this chapter we have introduced the concepts of classical linear and nonlinear Bayesian signal processing using state–space models. After developing the idea of linearizing a nonlinear state–space system, we developed the linearized Bayesian processor (*LZ-BP*). It was shown that the resulting processor provides an approximate solution (time-varying) to the nonlinear state estimation problem. We then developed the extended Bayesian processor (*XBP*) or equivalently the extended Kalman filter (*EKF*), as a special case of the *LZ-BP* linearizing about the most currently available estimate. Next we investigated a further enhancement of the *XBP* by introducing a local iteration of the nonlinear measurement system using a Newton-Rhapson method. Here the processor is called the iterated-extended Bayesian processor (*IX-BP*) and is shown to produce improved estimates at a small computational cost in most cases. Examples were developed throughout to demonstrate the concepts (see **http://www.techni-soft.net** for more details).

MATLAB NOTES

SSPACK_PC [8] is a third-party toolbox in *MATLAB* that can be used to design classical Bayesian processors. This package incorporates the major *nonlinear BP* algorithms discussed in this chapter—all implemented in the *UD*-factorized form [21] for stable and efficient calculations. It performs the discrete approximate Gauss-Markov simulations using (**SSNSIM**) and both extended (*XBP*) and iterated-extended (*IX-BP*) processors using (**SSNEST**). The linearized Bayesian processor (*LZ-BP*) 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 post-processor (**SSPOST**) which is used to analyze and display the results of the simulations and processing.

REBEL is a recursive Bayesian estimation package in *MATLAB* available on the web which performs similar operations including the new statistical-based unscented algorithms including the *UBP* (Chapter 6) including the unscented transformations [24]. It also includes the new particle filter designs (Chapter 7) discussed in [25] (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. Sage and J. Melsa, *Estimation Theory with Applications to Communications and Control* (New York: McGraw-Hill, 1971).

3. A. Gelb, *Applied Optimal Estimation* (Boston: MIT Press, 1975).
4. B. Anderson and J. Moore, *Optimal Filtering* (Englewood Cliffs, NJ: Prentice-Hall, 1979).
5. P. Maybeck, *Stochastic Models, Estimation, and Control* (New York: Academic Press, 1979).
6. M. Grewal and A. Andrews, *Kalman Filtering: Theory and Practice* (Englewood Cliffs, NJ: Prentice-Hall, 1993).
7. J. Mendel, *Lessons in Estimation Theory for Signal Processing, Communications and Control* (Englewood Cliffs, NJ: Prentice-Hall, 1995).
8. 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).
9. J. Candy, *Signal Processing: The Model-Based Approach* (New York: McGraw-Hill, 1986).
10. J. Candy, *Model-Based Signal Processing* (Hoboken, NJ: Wiley/IEEE Press, 2006).
11. D. Simon, *Optimal State Estimation: Kalman, H_∞ and Nonlinear Approaches* (Hoboken, NJ: John Wiley, 2006).
12. T. Kailath, A. Sayed and B. Hassibi, *Linear Estimation* (Englewood Cliffs, NJ: Prentice-Hall, 2000).
13. R. Mehra and J. Peschon, "An innovations approach to fault detection and diagnosis in dynamic systems," *Automatica*, **7**, 637–640, 1971.
14. A. Wilsky, "A survey of failure detection in dynamic systems," *Automatica*, **12**, 601–611, 1976.
15. Y. Bar-Shalom and X. Li, *Estimation and Tracking: Principles, Techniques, and Software* (Norwood, MA: Artech House, 1993).
16. B. Bell and F. Cathey, "The iterated Kalman filter update as a Gauss-Newton method," *IEEE Trans. Autom. Contr.*, **AC-38**, 2, 294–297, 1993.
17. J. Candy and R. Rozsa, "Safeguards for a plutonium-nitrate concentrator-an applied estimation approach,"*Automatica*, **16**, 615–627, 1980.
18. J. Candy and E. Sullivan, "Ocean Acoustic Signal Processing: A Model-Based Approach," *J. Acoust. Soc. Am.*, **92**, 3185–3201, 1992.
19. J. Mendel, J. Kormylo, J. Lee, and F. Ashirafi, "A novel approach to seismic signal processing and modeling," *Geophysics*, **46**, 1398–1414, 1981.
20. H. Sorenson Ed., "Special Issue on Applications of Kalman Filtering," *IEEE Trans. Auto. Contr.*, **AC-28**, 3, 253–427, 1983.
21. G. Bierman, *Factorization Methods of Discrete Sequential Estimation* (New York: Academic Press, 1977).
22. V. Aidala and S. Hammel, "Utilization of modified polar-coordinates for bearings-only tracking," *IEEE Trans. Autom. Contr.*, **AC-28**, 283–294, 1983.
23. V. Aidala, "Kalman filter behavior in bearings-only velocity and position estimation," *IEEE Trans. Aerosp. Electron. Syst.*, **AES-15**, 29–39, 1979.
24. R. van der Merwe and E. Wan, *REBEL: Recursive Bayesian Estimation Library University of Oregon* (see **http://choosh.ece.ogi.edu/rebel** for more details), 2002.
25. S. Haykin and N. de Freitas, Eds., "Sequential state estimation: from Kalman filters to particle filters," *Proc. IEEE*, **92**, 3, 399–574, 2004.

Problems

5.1 Derive the *continuous-time BP* by starting with the discrete equations of Table 5.1 and using the following sampled-data approximations:

$$\begin{aligned} A &= e^{A_c \Delta t} \approx I + A_c \Delta t \\ B &= B_c \Delta t \\ W &= W_c \Delta t \\ R_{ww} &= R_{w_c w_c} \Delta t \end{aligned}$$

5.2 Suppose we are given a continuous-time Gauss-Markov model characterized 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 *BP* using first difference approximations for derivatives and the discrete (sampled) system matrices derived above.

5.3 The covariance correction equation of the *BP* 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.

5.4 Prove that a necessary and sufficient condition for a *linear BP* to be optimal is that the corresponding innovations sequence is zero-mean and white.

5.5 A bird watcher is counting the number of birds migrating to and from a particular nesting area. Suppose the number of migratory birds, $m(t)$ is modeled by a first order *ARMA* model:

$$m(t) = -0.5m(t-1) + w(t) \quad \text{for } w \sim \mathcal{N}(10, 75)$$

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

$$y(t) = 0.5r(t) + m(t) + v(t) \quad \text{for } w \sim \mathcal{N}(0, 0.1)$$

(*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 $\hat{x}(2|2)$?

5.6 Suppose we are given a measurement device that not only acquires the current state but also the state 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 $\tilde{v}(t) \sim N(\mu_{\tilde{v}}, R_{\tilde{v}\tilde{v}})$.

(*d*) Are w and v correlated? If so, use the prediction form to develop the *MBP* algorithm for this system.

5.7 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:* $\ddot{x}(t) + \epsilon \dot{x}(t)[1 - \dot{x}^2(t)] + x(t) = m(t)$
- *Hill equation:* $\ddot{x}(t) - \alpha x(t) + \beta p(t)x(t) = m(t)$

(*a*) Develop the *LZ-BP*.

(*b*) Develop the *XBP*.

(*c*) Develop the *IX-BP*.

5.8 Suppose 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-BP* for this process.

(*b*) Develop the *XBP* for this process.

(*c*) Develop the *IX-BP* for this process.

(*d*) Suppose the parameters ω and α are unknown develop the XBP such that the parameters are jointly estimated along with the states. (*Hint*: Augment the states and parameters to create a new state vector).

5.9 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-BP* for this process.

(*c*) Choose $x^*(t) = \hat{x}(t)$ whenever $\hat{x}(t)$ is available during the recursion. Therefore, develop the continuous-discrete *XBP*.

5.10 Suppose we assume that a target is able to maneuver, that is, we assume that the target velocity satisfies a first order *AR* model given by:

$$v_\tau(t) = -\alpha v_\tau(t-1) + w_\tau(t-1) \quad \text{for } w \sim \mathcal{N}(0, R_{w_\tau w_\tau})$$

(*a*) Develop the Cartesian tracking model for this process.

(*b*) Develop the corresponding *XBP* assuming all parameters are known *a priori*.

(*c*) Develop the corresponding *XBP* assuming α is unknown.

5.11 Nonlinear processors (*LZ-BP*, *XBP*, *IX-BP*) can be used to develop neural networks used in many applications. Suppose 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-BP* for this process.

(*b*) Develop the *XBP* for this process.

(*c*) Develop the *IX-BP* for this process.

5.12 The Mackey-Glass time delay differential equation is given by

$$\begin{aligned}\dot{x}(t) &= \frac{\alpha x(t-\tau)}{1+x(t-\tau)^N} - \beta x(t) + w(t)\\ y(t) &= x(t) + v(t)\end{aligned}$$

where α, β 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 $x(0) = 1.2$

(*a*) Develop the *LZ-BP* for this process.

(*b*) Develop the *XBP* for this process.

(*c*) Develop the *IX-BP* for this process.

5.13 Consider the problem of estimating a random signal from an AM modulator characterized by

$$\begin{aligned}s(t) &= \sqrt{2P}a(t)\sin\omega_c t\\ r(t) &= s(t) + v(t)\end{aligned}$$

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 *BP*.

(*d*) Assume the carrier frequency, ω_c is unknown. Develop the *XBP* for this process.

6

MODERN BAYESIAN STATE–SPACE PROCESSORS

6.1 INTRODUCTION

In this chapter we discuss an extension of the approximate nonlinear Bayesian suite of processors that takes a distinctly different approach to the nonlinear Gaussian problem. Instead of attempting to improve on the linearized approximation in the nonlinear *XBP* (*EKF*) schemes discussed in the previous section or increasing the order of the Taylor series approximations [1–11] a modern *statistical* (linearization) *transformation* approach is developed. It is founded on the idea that "it is easier to approximate a probability distribution, than to approximate an arbitrary nonlinear function of transformation" [3, 12–21]. The classical nonlinear Bayesian processors discussed so far are based on linearizing nonlinear functions of the state and measurements to provide estimates of the underlying statistics (using Jacobians), while the statistical transformation approach is based on selecting a set of sample points that capture certain properties of the underlying distribution. This transformation is essentially a "statistical linearization" technique that incorporates the uncertainty of the prior random variable when linearizing [12]. This set of sample points 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 *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 [22]. Here the samples are *not* drawn at random, but according to a specific *deterministic* algorithm. Once this transformation is performed, the resulting processor, the sigma-point Bayesian processor (*SPBP*) or equivalently the unscented Kalman filter (*UKF*) evolves. It is a recursive processor that resolves some of the approximation issues [14] and deficiencies of the *XBP* of the previous

Bayesian Signal Processing. By James V. Candy

sections. We first develop the idea of nonlinearly transforming a random vector with known probability distribution and then apply it to a Gaussian problem leading to the *SPBP* algorithm. We then apply the modern processor to the previous nonlinear state estimation problem and compare its performance to the classical.

6.2 SIGMA-POINT (UNSCENTED) TRANSFORMATIONS

A completely different approach to nonlinear estimation evolves from the concept of statistical linearization [3, 16, 19, 24]. Instead of approximating the nonlinear process and measurement dynamics of the underlying system using a truncated Taylor series representation that leads to the classical forms of estimation (*LZKF*, *EKF*, *IEKF*, etc.), the statistical approximation or equivalently *statistical linearization* method provides an alternative that takes into account the uncertainty or probabilistic spread of the prior random vector. The basic idea is to approximate (linearize) a nonlinear function of a random vector while preserving its first and second moments; therefore, this approach requires *a priori* knowledge of its distribution resulting in a more statistically accurate transformation.

6.2.1 Statistical Linearization

Following Gelb [3], statistical linearization evolves from the idea of propagating an N_x-dimensional random vector $\mathbf{x}$ with *PDF*, $\mathrm{p}_X(\mathbf{x})$, through an arbitrary nonlinear transformation $\mathbf{a}[\cdot]$ to generate a new random vector,

$$\mathbf{y} = \mathbf{a}[\mathbf{x}] \tag{6.1}$$

Expanding this function in a Taylor series about $\mathbf{x}_i$ gives

$$\mathbf{a}[\mathbf{x}] = \mathbf{a}[\mathbf{x}_i] + (\mathbf{x} - \mathbf{x}_i)' \nabla_{\mathbf{x}} \mathbf{a}[\mathbf{x}_i] + \text{H.O.T.} \tag{6.2}$$

where $\nabla_{\mathbf{x}}$ is the N_x-gradient vector defined by

$$\nabla_{\mathbf{x}} \mathbf{a}[\mathbf{x}_i] := \left. \frac{\partial \mathbf{a}[\mathbf{x}]}{\partial \mathbf{x}} \right|_{\mathbf{x}=\mathbf{x}_i} \tag{6.3}$$

Neglecting the H.O.T and simplifying with the appropriate definitions, we obtain the "regression form" of $\mathbf{y}$ regressing on $\mathbf{x}$

$$\mathbf{y} = \mathbf{a}[\mathbf{x}] \approx A\mathbf{x} + \mathbf{b} \tag{6.4}$$

where both A, $\mathbf{b}$ are to be determined given $\mathbf{x} \sim \mathrm{p}_X(\mathbf{x})$. Estimation of A, $\mathbf{b}$ follows from the traditional linear algebraic perspective by defining the approximation or *linearization error* as:

$$\epsilon := \mathbf{y} - A\mathbf{x} - \mathbf{b} = \mathbf{a}[\mathbf{x}] - A\mathbf{x} - \mathbf{b} \tag{6.5}$$

along with the corresponding cost function

$$J_\epsilon := E\{\epsilon'\epsilon\} \tag{6.6}$$

that is minimized with respect to the unknowns $A, \mathbf{b}$. Performing the usual differentiation of the cost function, $J_\epsilon \to J_\epsilon(A, \mathbf{b})$, with respect to the unknowns, setting the results to zero and solving generates the *MMSE* result discussed previously in Chapter 2. That is, we first minimize with respect to $\mathbf{b}$

$$\nabla_{\mathbf{b}} J_\epsilon(A, \mathbf{b}) = \nabla_{\mathbf{b}} E\{\epsilon'\epsilon\} = E\{\nabla_{\mathbf{b}}(y - A\mathbf{x} - \mathbf{b})'\epsilon\} = 0 \tag{6.7}$$

Now using the chain rule of Eq. 2.11 with $a' = (y - A\mathbf{x} - \mathbf{b})'$ and $b = \epsilon$ we obtain

$$\nabla_{\mathbf{b}} J_\epsilon(A, \mathbf{b}) = E\{\nabla_{\mathbf{b}}(y - A\mathbf{x} - \mathbf{b})'\epsilon\} = -2E\{(y - A\mathbf{x} - \mathbf{b})\} = 0$$

Solving for $\mathbf{b}$, we obtain

$$\mathbf{b} = \mu_y - A\mu_x \tag{6.8}$$

Furthermore, substituting for $\mathbf{b}$ into the cost and differentiating with respect to A, setting the result to zero and solving yields

$$\nabla_A J_\epsilon(A, \mathbf{b})|_{\mathbf{b}=\mu_y - A\mu_x} = E\{A(x - \mu_x)(x - \mu_x)' + (y - \mu_y)(x - \mu_x)'\} = 0$$

giving

$$AE\{(x - \mu_x)(x - \mu_x)'\} = E\{(y - \mu_y)(x - \mu_x)'\}$$

or

$$AR_{xx} = R_{yx}$$

that has the *MMSE* solution

$$A = R_{yx}R_{xx}^{-1} = R'_{xy}R_{xx}^{-1} \tag{6.9}$$

Now suppose we linearize the function through this relation by constructing a linear regression between a selected set of N_x-points and the nonlinear transformation, $\mathbf{a}[\mathbf{x}]$, at these selected points. That is, defining the set of the selected points as $\{\mathcal{X}_i, \mathcal{Y}_i\}$ with

$$\mathcal{Y}_i = \mathbf{a}[\mathcal{X}_i]$$

Following the same approach as before by defining the "pointwise" linearization error as

$$\epsilon_i = \mathcal{Y}_i - A\mathcal{X}_i - \mathbf{b} = \mathbf{a}[\mathcal{X}_i] - A\mathcal{X}_i - \mathbf{b} \tag{6.10}$$

with

$$\mu_\epsilon = 0, \quad \text{and} \quad R_{\epsilon\epsilon} = R_{yy} - AR_{xx}A' \tag{6.11}$$

Performing a *weighted* minimization (as above) on the pointwise linearization error with a set of regression weights, $\{W_i\}; i = 1, \ldots, N_x$ yields the following solution:

$$A = R'_{xy} R_{xx}^{-1}, \quad \mathbf{b} = \mu_y - A\mu_x \tag{6.12}$$

where the weighted statistics are given by

$$\begin{aligned} \mu_x &= \sum_{i=1}^{N_x} W_i \mathcal{X}_i \\ R_{xx} &= \sum_{i=1}^{N_x} W_i (\mathcal{X}_i - \mu_x)(\mathcal{X}_i - \mu_x)' \end{aligned} \tag{6.13}$$

and for the posterior

$$\begin{aligned} \mu_y &= \sum_{i=1}^{N_x} W_i \mathcal{Y}_i \\ R_{yy} &= \sum_{i=1}^{N_x} W_i (\mathcal{Y}_i - \mu_y)(\mathcal{Y}_i - \mu_y)' \end{aligned} \tag{6.14}$$

with the corresponding cross-covariance

$$R_{xy} = \sum_{i=1}^{N_x} W_i (\mathcal{X}_i - \mu_x)(\mathcal{Y}_i - \mu_y)' \tag{6.15}$$

With this underlying regression and pointwise transformation in place, the next step is to determine the corresponding set of regression (sigma) points $\{\mathcal{X}_i\}$ and their corresponding weights, $\{W_i\}$.

6.2.2 Sigma-Point Approach

The *sigma-point transformation* (*SPT*) or equivalently *unscented transformation* (*UT*) is a technique for calculating the statistics of a random vector that has been nonlinearly transformed. The approach is illustrated in Fig. 6.1. Here the set of samples or so-called *sigma points* are chosen so that they capture the specific properties of the underlying distribution. In the figure we consider $\mathrm{p}_X(x)$ to be a $2D$-Gaussian, then the σ-points are located along the major and minor axes of the covariance ellipse capturing the essence of this distribution. In general, the goal is to construct a set of σ-points possessing the same statistics as the original distribution such that when nonlinearly transformed to the new space, then new set of points sufficiently capture the posterior statistics. The transformation occurs on a point-by-point basis, since it is simpler to match statistics of individual points rather than the entire *PDF*. The

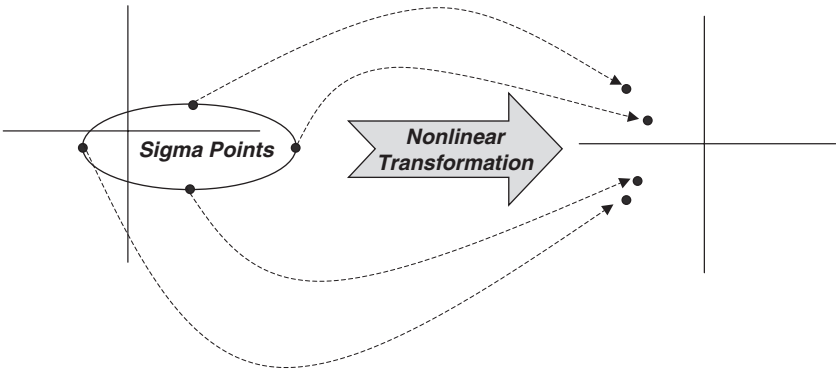


FIGURE 6.1 Unscented transformation. A set of distribution points shown on an error ellipsoid are selected and transformed into a new space where their underlying statistics are estimated.

statistics of the transformed points are then calculated to provide the desired estimates of the transformed distribution.

Following the development of Julier [15], consider propagating an N_x-dimensional random vector, $\mathbf{x}$, through an arbitrary nonlinear transformation $\mathbf{a}[\cdot]$ to generate a new random vector,

$$\mathbf{y} = \mathbf{a}[\mathbf{x}] \tag{6.16}$$

The set of σ-points, $\{\mathcal{X}_i\}$, consists of $N_\sigma + 1$ vectors with appropriate weights, $\{W_i\}$, given by $\Sigma = \{\mathcal{X}_i, W_i; i = 0, \ldots, N_\sigma\}$. The weights can be positive or negative but *must satisfy* the *normalization constraint*

$$\sum_{i=0}^{N_\sigma} W_i = 1$$

so that the estimate of the statistics remains unbiased. The problem then becomes:

GIVEN the sigma-points, $\Sigma = \{\mathcal{X}_i, W_i;\ i = 0, \ldots, N_\sigma\}$, and the nonlinear transformation $\mathbf{a}[\mathbf{x}]$, **FIND** the statistics of the transformed samples,

$$\mu_y = E\{\mathbf{y}\} \quad \text{and} \quad \mathbf{R}_{yy} = \text{Cov}(\mathbf{y})$$

The *sigma-point (unscented) transformation* (*SPT*) approach to approximate the statistics, $(\mu_y, \mathbf{R}_{yy})$ is:

1. Determine the number, weights and locations of the σ-point set, Σ, based on the unique characteristics of the prior distribution

2. Nonlinearly *transform* each point to obtain the set of new (posterior) σ-points, $\{\mathcal{Y}_i\}$:

$$\mathcal{Y}_i = \mathbf{a}[\mathcal{X}_i] \tag{6.17}$$

3. Estimate the posterior *mean* by its weighted average[1]

$$\mu_{\mathbf{y}} = \sum_{i=0}^{N_\sigma} W_i \mathcal{Y}_i \tag{6.18}$$

4. Estimate the posterior *covariance* by its weighted outer product

$$\mathbf{R}_{yy} = \sum_{i=0}^{N_\sigma} W_i (\mathcal{Y}_i - \mu_y)(\mathcal{Y}_i - \mu_y)' \tag{6.19}$$

One set of σ-points that satisfies the above conditions consists of a symmetric set of $N_\sigma = 2N_x + 1$ points that lie on the $\sqrt{N_x}$-th covariance contour [13]:

$$\begin{aligned} \mathcal{X}_o &= \mu_x, & W_o &= \frac{1}{N_x} \\ \mathcal{X}_i &= \mu_x + \sqrt{N_x}\,\sigma_i, & W_i &= \frac{1}{2N_x} \\ \mathcal{X}_{i+N_x} &= \mu_x - \sqrt{N_x}\,\sigma_i, & W_{i+N_x} &= \frac{1}{2N_x} \end{aligned}$$

where $\sqrt{N_x}\,\sigma_i$ is the i^{th} standard deviation scaled by $\sqrt{N_x}$ and W_i is the weight associated with the i^{th} σ-point.

Thus, the σ-point transformation can be considered a *statistical linearization method* that provides an optimal (*MMSE*) linear approximation to a general nonlinear transformation taking into account the prior second-order statistics of the underlying random variable, that is, its mean and covariance [19]. It can be accomplished using the weighted statistical linear regression approach (*WSLR*) [16, 19], resulting in the weight-constrained estimates above.

Therefore, in contrast to random sampling, selection of the "deterministic" σ-points requires resolving the following critical issues:

- N_σ, the *number* of σ-points;
- W_i, the *weights* assigned to each σ-point; and
- $\mathcal{X}_i$, the *location* of the σ-points.

That is, we must answer the questions of: How many (points)?, What (weights)? and Where (located)?

[1] Note that this estimate is actually a weighted statistical linear regression (*WSLR*) of the random variable [19].

The σ-points should be *selected* or constrained to capture the "most important" statistical properties of the random vector, $\mathbf{x}$. Let the underlying prior $\mathrm{p}_X(\mathbf{x})$ be its density function, then the σ-points capture the properties by satisfying the *necessary (constraint) condition*

$$\mathbf{g}[\Sigma, \mathrm{p}_X(\mathbf{x})] = \mathbf{0} \tag{6.20}$$

Since it is possible to meet this constraint condition and still have some degree of freedom in the choice of σ-points, assigning a *penalty* function

$$\mathbf{p}[\Sigma, \mathrm{p}_X(\mathbf{x})] \tag{6.21}$$

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. In general, the σ-point set relative to this problem that is the most desirable is the set that conforms to the necessary conditions of Eqs. 6.20 and 6.21. The σ-points must *satisfy*

$$\min_{\Sigma} \mathbf{p}[\Sigma, \mathrm{p}_X(\mathbf{x})] \ni \mathbf{g}[\Sigma, \mathrm{p}_X(\mathbf{x})] = \mathbf{0} \tag{6.22}$$

The decision as to which properties of the random vector $\mathbf{x}$ to capture precisely or approximate depends on the particular application. For our problems, we wish to match the *moments* of the underlying distribution of σ-points with those of $\mathbf{x}$.

Summarizing, to apply the *SPT* approach: (1) a set of σ-points, Σ, are constructed that are "deterministically" constrained to possess the identical statistics of the prior; (2) each σ-point is nonlinearly transformed to the new space, $\mathcal{Y}_i$; and (3) the statistics of the transformed set are approximated using *WSLR* techniques. The following example illustrates this approach.

Example 6.1

Consider a scalar random variable, x, for which we would like to propagate its first two moments (μ_x, R_{xx}) through a nonlinear transformation, $\mathcal{Y}_i = \mathbf{a}[\mathcal{X}_i]$. The corresponding set of constraint equations are defined by:

$$\mathbf{g}(\Sigma, \mathrm{p}_X(x)) = \begin{bmatrix} \sum_{i=0}^{N_\sigma} W_i - 1 \\ \sum_{i=0}^{N_\sigma} W_i \mathcal{X}_i - \mu_x \\ \sum_{i=0}^{N_\sigma} W_i (\mathcal{X}_i - \mu_x)(\mathcal{X}_i - \mu_x)' - R_{xx} \end{bmatrix} = \mathbf{0}$$

with posterior

$$\mu_y = \sum_{i=0}^{N_\sigma} W_i \mathcal{Y}_i$$

$$R_{yy} = \sum_{i=0}^{N_\sigma} W_i (\mathcal{Y}_i - \mu_y)(\mathcal{Y}_i - \mu_y)'$$

Suppose $x \sim \mathcal{N}(0,1)$, then the set of σ-points are defined by $N_\sigma = 2$ or 3 points with $\mathcal{X}_0 = \sigma_0 = 0$, $\mathcal{X}_1 = -\sigma_1$, and $\mathcal{X}_2 = \sigma_2$. The constraint equations become

$$\begin{aligned} W_0 + W_1 + W_2 - 1 &= 0 \\ -W_1\sigma_1 + W_2\sigma_2 - 0 &= 0 \\ W_1\sigma_1^2 + W_2\sigma_2^2 - 1 &= 0 \end{aligned}$$

for W_0, a free parameter whose value affects the 4^{th} and higher moments. These relations are not uniquely satisfied (4 unknowns, 3 equations); therefore, we must add another constraint to the set based on the property that the *skew* is zero for the Gaussian

$$-W_1\sigma_1^3 + W_2\sigma_2^3 = 0$$

Solving these equations and using the symmetry property of the Gaussian gives:

$$\sigma_1 = 1/2\sqrt{W_1}, \quad W_1 = (1 - W_0)/2, \quad \sigma_2 = \sigma_1, \quad W_2 = W_1 \qquad \triangle\triangle\triangle$$

This example illustrates the use of the *constraint* equations to determine the σ-points from properties of the prior. Next, let us incorporate the nonlinear transformation and penalty function to demonstrate the entire selection procedure. Following Julier [13] consider the following one-dimensional example.

Example 6.2

As before, suppose we have a scalar random variable x, Gaussian distributed with mean μ_x and variance σ_x^2, and we would like to know the statistics (mean and variance) of y which nonlinearly transforms x according to

$$y = a[x] = x^2$$

Here the true mean and variance are

$$\mu_y = E\{y\} = E\{x^2\} = \sigma_x^2 + \mu_x^2$$

and

$$\begin{aligned} \sigma_y^2 &= E\{y^2\} - \mu_y^2 = E\{x^4\} - \mu_y^2 = (3\sigma_x^4 + 6\sigma_x^2\mu_x^2 + \mu_x^4) - (\sigma_x^4 + 2\sigma_x^2\mu_x^2 + \mu_x^4) \\ &= 2\sigma_x^4 + 4\sigma_x^2\mu_x^2 \end{aligned}$$

According to *SPT* of Eqns. 6.17–6.19 the number of σ-points is $N_\sigma = 3$. Since this is a scalar problem ($N_x = 1$) only 3 points are required: the two σ-points and the mean,

therefore, we have

$$\{\mathcal{X}_0, \mathcal{X}_1, \mathcal{X}_2\} = \{\mu_x, \mu_x - \sqrt{1+\kappa}\,\sigma_x, \mu_x + \sqrt{1+\kappa}\,\sigma_x\}$$
$$\{W_0, W_1, W_2\} = \{1/(1+\kappa), 1/2(1+\kappa), 1/2(1+\kappa)\}$$

and κ is chosen as a scaling factor to be determined. Propagating these samples through $a[\cdot]$ gives the transformed samples, say $\mathcal{X}_i'$ that lie at

$$\{\mathcal{X}_0', \mathcal{X}_1', \mathcal{X}_2'\} = \left\{\mu_x^2, (\mu_x - \sqrt{1+\kappa}\,\sigma_x)^2, (\mu_x + \sqrt{1+\kappa}\,\sigma_x)^2\right\}$$

The mean of y is given by

$$\mu_y = \frac{1}{2(1+\kappa)}(2\kappa\mu_x^2 + 2\mu_x^2 + 2(1+\kappa)\sigma_x^2) = \mu_x^2 + \sigma_x^2$$

which is precisely the true mean. Next the covariance is given by

$$\sigma_y^2 = \frac{1}{2(1+\kappa)}\left(\sum_{i=1}^{2}((\mathcal{X}_i' - \mu_y)^2 + 2\kappa\sigma_x^4)\right) = \kappa\sigma^4 + 4\mu_x^2\sigma_x^2$$

To find the solution, κ must be specified. The kurtosis of the true distribution is $2\sigma_x^4$ and that of the σ-points is σ_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. △△△

Next using the underlying principles of the *SPT*, we develop the multivariate Gaussian case in more detail.

6.2.3 SPT for Gaussian Prior Distributions

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 *XBP* (*EKF*). This parameterization captures the prior *mean* and *covariance* 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. As illustrated in Fig. 6.2, we see samples of the true prior distribution of x and the corresponding nonlinearly transformed distribution of y. Using the same transformation, the selected σ-points are transformed as well closely preserving the dominant moments of the original distribution (see Julier [13] for more details).

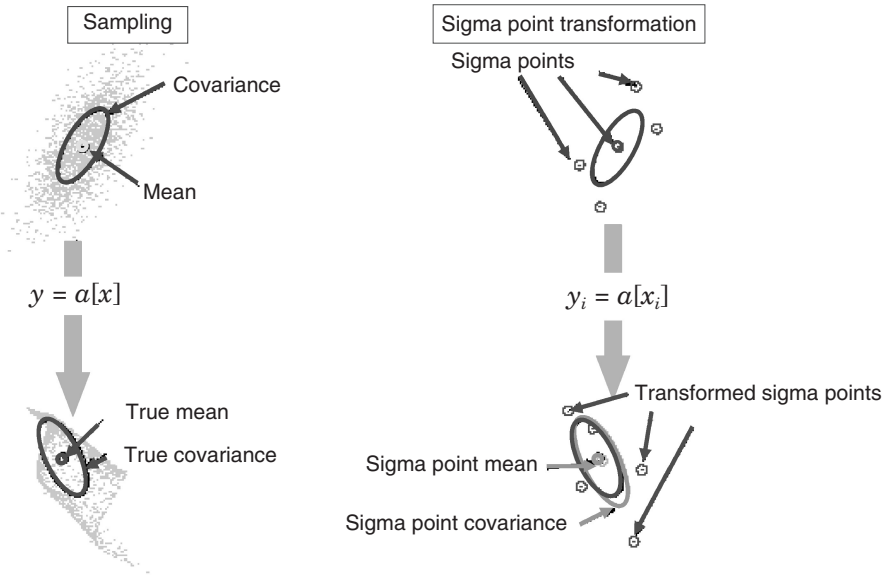


FIGURE 6.2 Sigma point (unscented) transformation approximation of the original distribution moments after nonlinear transformation.

It is important to recognize that the *SPT* has specific properties when the underlying distribution is Gaussian [16]. The Gaussian has two properties which play a significant role in the form of the σ-points *selected*. First, since the distribution is *symmetric*, the σ-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

$$\mathbf{x} = \mu_{\mathbf{x}} + U\mathbf{s} \quad \text{for } U \text{ the matrix square root of } R_{xx} \tag{6.23}$$

where $\mathbf{s} \sim \mathcal{N}(0, I)$. Therefore, in the Gaussian case, the second order *SPT* uses a set of σ-points which capture 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 its covariance is given by

$$E\{s_i^2\} = 1 \quad \forall i \tag{6.24}$$

Also from the symmetry properties of the distribution, all *odd*-ordered moments are zero.

The minimum number of points whose distribution obeys these conditions has two types of σ-points: (1) a single point at the origin of the s-axis with weight, W_o; and (2) $2N_x$ symmetrically distributed points on the coordinate s-axis a distance r from the origin all having the same weight, W_1. Thus, there are $N_\sigma = 2N_x + 1$ σ-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 **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

$$\mathbf{g}[\sigma, \mathrm{p}_X(\mathbf{s})] = \begin{pmatrix} 2W_1 r^2 - 1 \\ W_o + 2N_x W_1 - 1 \end{pmatrix} = \mathbf{0} \tag{6.25}$$

The solution to these equations is given by

$$r = \frac{1}{\sqrt{2W_1}} \quad \text{and} \quad W_o = 1 - 2N_x W_1, \qquad W_o \text{ free} \tag{6.26}$$

By reparameterizing $W_1 := \frac{1}{2(N_x+\kappa)}$, then it can be shown that after pre-multiplying by U, the matrix square root of R_{xx}, that the σ-points for **x** are:

$$\begin{aligned} \mathcal{X}_o &= \mu_x, & W_o &= \frac{\kappa}{(N_x+\kappa)} \\ \mathcal{X}_i &= \mu_x + (\sqrt{(N_x+\kappa)R_{xx}})_i, & W_i &= \frac{1}{2(N_x+\kappa)} \\ \mathcal{X}_{i+N_x} &= \mu_x - (\sqrt{(N_x+\kappa)R_{xx}})_i, & W_{i+N_x} &= \frac{1}{2(N_x+\kappa)} \end{aligned}$$

where κ is a scalar, $(\sqrt{(N_x+\kappa)\,R_{xx}})_i$ is the i^{th} row or column of the matrix square root of $(N_x+\kappa)R_{xx}$ and W_i is the weight associated with the i^{th} σ-point. The parameter κ is free; however, it can be selected to minimize the mismatch between the fourth-order moments of the σ-points and the true distribution [16]. From the properties of the Gaussian, we have that

$$E\{s_i^4\} = 3 \quad \forall\, i \tag{6.27}$$

The penalty function penalizes the discrepancy between the σ-points and the true value along one direction (s_1), in this case, due to the symmetry. Therefore, we have that

$$\mathbf{p}[\sigma, \mathrm{p}_X(\mathbf{s})] = |2W_1 r^4 - 3| \quad \text{giving } W_1 = \frac{3}{2r^4} = \frac{1}{6} \quad \text{or} \quad \kappa = N_x - 3 \tag{6.28}$$

It is clear that the ability to minimize **p** depends on the number of degrees of freedom that are available after the constraint **g** is satisfied for the given set of σ-points; the kurtosis cannot be matched exactly without developing a larger set of σ-points (see Julier [16] for details).

We summarize the sigma-point processor under the *multivariate* Gaussian assumptions: Given an N_x-dimensional Gaussian distribution having covariance R_{xx} we can generate a set of $O(N_\sigma)$ σ-points having the same sample covariance from the columns (or rows) of the matrices $\pm\sqrt{(N_x+\kappa)R_{xx}}$. Here κ is the scaling factor discussed previously. This set is zero mean, but if the original distribution has mean μ_x, then simply adding μ_x to each of the σ-points yields a symmetric set of $N_\sigma = 2N_x + 1$ 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 σ-points capable of capturing the essential statistical information. The basic *SPT* technique for a multivariate Gaussian distribution [16] is therefore:

1. Determine the set of $N_\sigma = 2N_x + 1$ σ-points from the rows or columns of $\pm\sqrt{(N_x + \kappa)\, R_{xx}}$. For the nonzero mean case compute, $\mathcal{X}_i = \sigma + \mu_x$;

$$\begin{aligned} \mathcal{X}_o &= \mu_x, & W_o &= \frac{\kappa}{(N_x + \kappa)} \\ \mathcal{X}_i &= \mu_x + \left(\sqrt{(N_x + \kappa)R_{xx}}\right)_i, & W_i &= \frac{1}{2(N_x + \kappa)} \\ \mathcal{X}_{i+N_x} &= \mu_x - \left(\sqrt{(N_x + \kappa)R_{xx}}\right)_i, & W_{i+N_x} &= \frac{1}{2(N_x + \kappa)} \end{aligned}$$

   where κ is a scalar, $\left(\sqrt{(N_x + \kappa)R_{xx}}\right)_i$ is the i^{th} row or column of the matrix square root of $(N_x + \kappa)R_{xx}$ and W_i is the weight associated with the i^{th} σ-point;
2. Nonlinearly *transform* each point to obtain the set of new σ-points: $\mathcal{Y}_i = \mathbf{a}[\mathcal{X}_i]$
3. Estimate the posterior *mean* of the new samples by its weighted average (regression)

$$\mu_{\mathbf{y}} = \sum_{i=0}^{2N_x} W_i \mathcal{Y}_i$$

4. Estimate the posterior *covariance* of the new samples by its weighted outer product (regression)

$$\mathbf{R}_{yy} = \sum_{i=0}^{2N_x} W_i (\mathcal{Y}_i - \mu_{\mathbf{y}})(\mathcal{Y}_i - \mu_{\mathbf{y}})'$$

There is a wealth of properties of this processor:

1. The transformed statistics of $\mathbf{y}$ are captured *precisely* up to the second order.
2. The σ-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 (*WSLR*) and it is *not* necessary to evaluate any Jacobian as required by the *XBP* methods.
4. κ is a "tuning" parameter used to tune the higher order moments of the approximation that can be used to reduce overall prediction errors. For $\mathbf{x}$ a multivariate Gaussian, $N_x + \kappa = 3$ is a useful heuristic.
5. A modified form for $\kappa \rightarrow \lambda = \alpha^2(N_x + \kappa) - N_x$ (scaled transform) can be used to overcome a nonpositive definiteness of the covariance. Here α controls the

spread of the σ-points around μ_x and is typically set to a value $0.01 \leq \alpha \leq 1$ with κ a secondary scaling parameter set to 0 or $3 - N_x$ and β is an extra degree of freedom to incorporate any extra prior knowledge of the $p_X(x)$ with $\beta = 2$ for Gaussian distributions. In this case the weights change and are given by: $W_0^{(m)} = \frac{\lambda}{N_x+\lambda}$, $W_0^{(c)} = \frac{\lambda}{N_x+\lambda} + (1 - \alpha^2 + \beta)$, and $W_i^{(m)} = \frac{1}{(N_x+\lambda)}$ [17, 21, 24].

6. Note that although statistical linearization offers a convenient way to interpret the subsequent *sigma-point approach*, it does not indicate some of its major advantages, especially since it is possible to extend the approach to incorporate more points capturing and accurately propagating higher order moments [24].

Next we apply the σ-point approach to the nonlinear filtering problem by defining the terms in the *SPT* and showing where the statistical approximations are utilized.

6.3 SIGMA-POINT BAYESIAN PROCESSOR (UNSCENTED KALMAN FILTER)

The *SPBP* or *UKF* 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. Different from the *XBP* (*EKF*), the σ-point processor 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 using a statistical linearization approach leading to a set of regression equations for the states and measurements. In the sigma-point processor, the state is still represented as Gaussian, but it is specified using the minimal set of *deterministically* selected samples or σ-points. These points completely capture the true mean and covariance of the prior Gaussian distribution. When they are propagated through the 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. This is the statistical linearization using the weighted (statistical) linear regression approximation (*WSLR*) discussed previously [16].

We use the *XBP* (*EKF*) 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

$$x(t) = \mathbf{a}[x(t-1)] + \mathbf{b}[u(t-1)] + w(t-1) \tag{6.29}$$

with corresponding measurement model

$$y(t) = \mathbf{c}[x(t)] + v(t) \tag{6.30}$$

for $w \sim \mathcal{N}(0, R_{ww})$ and $v \sim \mathcal{N}(0, R_{vv})$. It was demonstrated previously (Sec. 5.3) that the critical conditional Gaussian distribution for the *state variable* statistics was the prior

$$\Pr(x(t)|Y_{t-1}) = \mathcal{N}(\hat{x}(t|t-1), \tilde{P}(t|t-1))$$

with the measurement statistics specified by

$$\Pr(y(t)|Y_{t-1}) = \mathcal{N}(\hat{y}(t|t-1), R_{\xi\xi}(t|t-1))$$

where $\hat{x}(t|t-1)$, and $\tilde{P}(t|t-1)$ are the respective predicted state and error covariance based upon the data up to time $(t-1)$ and $\hat{y}(t|t-1)$, $R_{\xi\xi}(t|t-1)$ are the predicted measurement and residual covariance. The idea is to use the "prior" statistics and perform the *SPT* (under Gaussian assumptions) using *both* the process and measurement nonlinear transformations (models) as specified above yielding the corresponding set of σ-points in the new space. The predicted means are weighted sums of the transformed σ-points and the covariances are merely weighted sums of their mean-corrected, outer products, that is, they are specifically the *WSLR* discussed in Sec. 6.2.

To develop the sigma-point processor we must:

- PREDICT the next state and error covariance, $(\hat{x}(t|t-1), \tilde{P}(t|t-1))$, by *SPT* transforming the prior, $\mathcal{N}(\hat{x}(t-1|t-1), \tilde{P}(t-1|t-1))$, including the process noise using the σ-points, $\mathcal{X}_i(t|t-1)$ and $\mathcal{X}_i(t-1|t-1)$, respectively;
- PREDICT the measurement and residual covariance, $[\hat{y}(t|t-1), R_{\xi\xi}(t|t-1)]$ by using the *SPT* transformed σ-points $\mathcal{Y}_i(t|t-1)$ and performing the weighted regressions; and
- PREDICT the cross-covariance, $R_{\tilde{x}\xi}(t|t-1)$ in order to calculate the corresponding gain for the subsequent update step.

We use these steps as our road map to develop the sigma-point processor depicted in the block diagram of Fig. 6.3. We start by defining the set, Σ, and selecting the corresponding, $N_\sigma = 2N_x + 1$ σ-points and weights according to the multivariate Gaussian distribution selection procedure developed in the previous section. That is, the N_σ points are defined by substituting the σ-points for the *SPT* transformation of the "prior" *state* information specified by $\mu_x = \hat{x}(t-1|t-1)$ and $R_{xx} = \tilde{P}(t-1|t-1)$. We define the set of σ-points and weights as:

$$\begin{aligned}
\mathcal{X}_o &= \mu_x = \hat{x}(t-1|t-1), & W_o &= \frac{\kappa}{(N_x+\kappa)} \\
\mathcal{X}_i &= \mu_x + \left(\sqrt{(N_x+\kappa)R_{xx}}\right)_i \\
&= \hat{x}(t-1|t-1) + \left(\sqrt{(N_x+\kappa)\tilde{P}(t-1|t-1)}\right)_i, & W_i &= \frac{1}{2(N_x+\kappa)} \\
\mathcal{X}_{i+N_x} &= \mu_x - \left(\sqrt{(N_x+\kappa)R_{xx}}\right)_i \\
&= \hat{x}(t-1|t-1) - \left(\sqrt{(N_x+\kappa)\tilde{P}(t-1|t-1)}\right)_i, & W_{i+N_x} &= \frac{1}{2(N_x+\kappa)}
\end{aligned}$$

Next we perform the state prediction-step to obtain, $\{\mathcal{X}_i(t|t-1), \hat{x}(t|t-1)\}$, transforming each σ-point using the nonlinear process model to the new space, that is, to

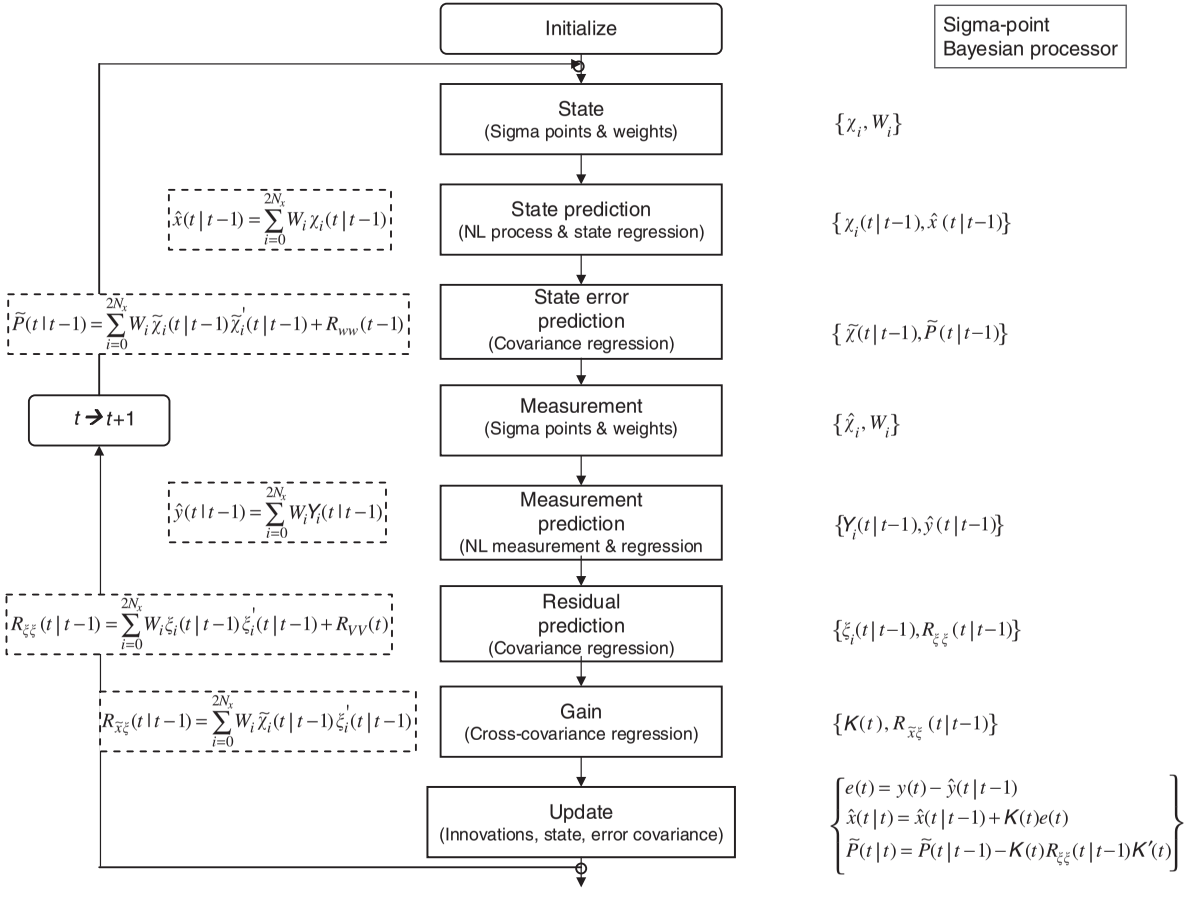


FIGURE 6.3 Sigma-point Bayesian processor block diagram with sigma-points and regressions.

obtain the one-step *state prediction* as

$$\mathcal{X}_i(t|t-1) = \mathbf{a}[\mathcal{X}_i(t-1|t-1)] + \mathbf{b}[u(t-1)] \tag{6.31}$$

The *WSLR* approximation step to obtain the predicted mean follows from the statistical linearization transformation model using the following relations: $\mathbf{y} \to x(t)$, $\mathbf{x} \to x(t-1)$, $A \to A(t-1)$, $\mathbf{b} \to \mathbf{b}(t-1)$ and $\epsilon \to \epsilon(t-1)$ of Eq. 6.4

$$x(t) = \underbrace{A(t-1)x(t-1) + \mathbf{b}(t-1)}_{\text{Linear Approximation}} + \underbrace{\epsilon(t-1)}_{\text{Linearization Error}} \tag{6.32}$$

Conditioning Eq. 6.32 on Y_{t-1} and taking expectations, we have

$$\begin{aligned}\hat{x}(t|t-1) = E\{x(t)|Y_{t-1}\} \;=\; & E\{A(t-1)x(t-1)|Y_{t-1}\} + E\{\mathbf{b}(t-1)|Y_{t-1}\} \\ & + E\{\epsilon(t-1)|Y_{t-1}\}\end{aligned}$$

or

$$\hat{x}(t|t-1) = A(t-1)\hat{x}(t-1|t-1) + \mathbf{b}(t-1) \tag{6.33}$$

Using this linearized form, with the conditional means replacing the unconditional, that is, $\mu_y \to \hat{x}(t|t-1)$ and $\mu_x \to \hat{x}(t-1|t-1)$, we have from Eq. 6.8 that

$$\begin{aligned}\mathbf{b}(t-1) &\to \hat{x}(t|t-1) - A(t-1)\hat{x}(t-1|t-1) \\ &= \sum_{i=0}^{2N_x} W_i \mathcal{X}_i(t|t-1) - A(t-1)\hat{x}(t-1|t-1)\end{aligned} \tag{6.34}$$

where we substituted the regression of Eq. 6.14 for μ_y. Substituting this expression for $\mathbf{b}$ above gives the required regression relation

$$\begin{aligned}\hat{x}(t|t-1) \;&=\; A(t-1)\hat{x}(t-1|t-1) + \sum_{i=0}^{2N_x} W_i \mathcal{X}_i(t|t-1) - A(t-1)\hat{x}(t-1|t-1) \\ &= \sum_{i=0}^{2N_x} W_i \mathcal{X}_i(t|t-1)\end{aligned} \tag{6.35}$$

yielding the *predicted state* estimate using the *WSLR* approach shown in Fig. 6.3.

Next let us define the *predicted state estimation error* as

$$\tilde{\mathcal{X}}_i(t|t-1) := \mathcal{X}_i(t|t-1) - \hat{x}(t|t-1) \tag{6.36}$$

therefore, the corresponding *predicted state error covariance*,

$$\tilde{P}(t|t-1) := \text{cov}[\tilde{\mathcal{X}}_i(t|t-1)] = \text{cov}[\mathcal{X}_i(t|t-1) - \hat{x}(t|t-1)]$$

can be calculated from

$$\begin{aligned}\tilde{P}(t|t-1) &= \text{cov}[(A(t-1)\mathcal{X}_i(t-1|t-1)+\mathbf{b}(t-1)+\epsilon_i(t-1)) \\ &\quad -(A(t-1)\hat{x}(t-1|t-1)+\mathbf{b}(t-1))] \\ &= \text{cov}[A(t-1)\tilde{\mathcal{X}}_i(t-1|t-1)+\epsilon_i(t-1)]\end{aligned}$$

Performing this calculation gives the expression

$$\tilde{P}(t|t-1) = A(t-1)\tilde{P}(t|t-1)A'(t-1)+R_{\epsilon\epsilon}(t-1) \tag{6.37}$$

which can also be written in the regression form by using the relations from Eq. 6.11 with $y \to \mathcal{X}_i$, $R_{yy} \to \tilde{P}_{\mathcal{X}\mathcal{X}}$. Therefore, substituting for $A\tilde{P}A'$ above, we have

$$\begin{aligned}\tilde{P}(t|t-1) &= \tilde{P}_{\mathcal{X}\mathcal{X}}(t|t-1)-R_{\epsilon\epsilon}(t-1)+R_{\epsilon\epsilon}(t-1) \\ &= \sum_{i=0}^{2N_x} W_i\tilde{\mathcal{X}}_i(t|t-1)\tilde{\mathcal{X}}_i'(t|t-1)\end{aligned} \tag{6.38}$$

In the case of additive, zero-mean, white Gaussian noise with covariance, $R_{ww}(t-1)$, we have the final *WSLR*

$$\tilde{P}(t|t-1) = \sum_{i=0}^{2N_x} W_i\tilde{\mathcal{X}}_i(t|t-1)\tilde{\mathcal{X}}_i'(t|t-1)+R_{ww}(t-1) \tag{6.39}$$

completing the *state error prediction* step of Fig. 6.3.

Here, the Bayesian processor approximates the predicted density, $\mathcal{N}(\hat{x}(t|t-1), \tilde{P}(t|t-1))$, where

$$\text{p}_X(x(t)|Y_{t-1}) = \int \text{p}_X(x(t)|x(t-1)\text{p}_X(x(t-1)|Y_{t-1})\,dx(t-1) \tag{6.40}$$

Next we calculate a new set of σ-points to reflect the prediction-step and perform the *SPT* in the *measurement* space as

$$\hat{\mathcal{X}}_i(t|t-1) = \{\mathcal{X}_i(t|t-1),\ \mathcal{X}_i(t|t-1)+\kappa\sqrt{R_{ww}(t-1)},\ \mathcal{X}_i(t|t-1)-\kappa\sqrt{R_{ww}(t-1)}\} \tag{6.41}$$

We linearize the measurement function similar to the nonlinear process function in the previous steps. We use the nonlinear measurement model to propagate the "new" σ-points to the transformed space producing the *measurement prediction* step shown in Fig. 6.3.

$$\mathcal{Y}_i(t|t-1) = \mathbf{c}[\hat{\mathcal{X}}_i(t|t-1)] \tag{6.42}$$

The *WSLR* is then performed to obtain the *predicted measurement*, that is,

$$y(t) = C(t)x(t)+\mathbf{b}(t)+\epsilon(t) \tag{6.43}$$

Conditioning on Y_{t-1}, as before, and taking expectations of Eq. 6.43 gives

$$\hat{y}(t|t-1) = E\{y(t)|Y_{t-1}\} = C(t)\hat{x}(t|t-1) + \mathbf{b}(t) \tag{6.44}$$

Using this linearized form with the conditional means replacing the unconditional, that is, $\mu_y \to \hat{y}(t|t-1), A \to C, \mathbf{b} \to \mathbf{b}$ and $\mu_x \to \hat{x}(t|t-1)$, we have from Eq. 6.8 that

$$\mathbf{b}(t) \to \hat{y}(t|t-1) - C(t)\hat{x}(t|t-1) = \sum_{i=0}^{2N_x} W_i \mathcal{Y}_i(t|t-1) - C(t)\hat{x}(t|t-1) \tag{6.45}$$

where we substituted the regression of Eq. 6.14 for μ_y. Substituting this expression for $\mathbf{b}$ above gives the *WSLR* relation

$$\begin{aligned}\hat{y}(t|t-1) &= C(t)\hat{x}(t|t-1) + \sum_{i=0}^{2N_x} W_i \mathcal{Y}_i(t|t-1) - C(t)\hat{x}(t|t-1) \\ &= \sum_{i=0}^{2N_x} W_i \mathcal{Y}_i(t|t-1)\end{aligned} \tag{6.46}$$

yielding the *predicted measurement* estimate of Fig. 6.3.

Similarly, the *residual* (measurement) error is defined by:

$$\xi_i(t|t-1) := \mathcal{Y}_i(t|t-1) - \hat{y}(t|t-1) \tag{6.47}$$

and therefore the *residual (predicted) covariance* can be expressed as

$$R_{\xi\xi}(t|t-1) = \text{cov}[\xi_i(t|t-1)] = \text{cov}[\mathcal{Y}_i(t|t-1) - \hat{y}(t|t-1)]$$

Substituting the linearized model above, we obtain

$$\begin{aligned}R_{\xi\xi}(t|t-1) &= \text{cov}[(C(t)\mathcal{X}_i(t|t-1) + \mathbf{b}(t) + \epsilon_i(t)) \\ &\qquad - (C(t)\hat{x}(t|t-1) + \mathbf{b}(t))] \\ &= \text{cov}[C(t-1)\tilde{\mathcal{X}}_i(t|t-1) + \epsilon_i(t)]\end{aligned}$$

Performing this calculation gives the expression

$$R_{\xi\xi}(t|t-1) = C(t)\tilde{P}(t|t-1)C'(t) + R_{\epsilon\epsilon}(t) \tag{6.48}$$

which can also be written in the regression form by using the relations from Eq. 6.11 with $y \to \xi_i, R_{yy} \to \tilde{P}_{\xi\xi}$. Therefore, substituting for $C\tilde{P}C'$ above, we have

$$R_{\xi\xi}(t|t-1) = \tilde{P}_{\xi\xi}(t|t-1) - R_{\epsilon\epsilon}(t) + R_{\epsilon\epsilon}(t) = \sum_{i=0}^{2N_x} W_i \xi_i(t|t-1)\xi_i'(t|t-1) \tag{6.49}$$

In the case of additive, zero-mean, white Gaussian noise with covariance, $R_{vv}(t)$ we have the final *WSLR*

$$R_{\xi\xi}(t|t-1) = \sum_{i=0}^{2N_x} W_i \xi_i(t|t-1)\xi_i'(t|t-1) + R_{vv}(t) \tag{6.50}$$

completing the measurement *residual prediction* step of Fig. 6.3.

The *gain* is estimated from

$$\mathcal{K}(t) = R_{\tilde{x}\xi}(t|t-1)R_{\xi\xi}^{-1}(t|t-1) \tag{6.51}$$

where the *cross error covariance* is approximated using the *WSLR* as (above)

$$R_{\tilde{x}\xi}(t|t-1) = \sum_{i=0}^{2N_x} W_i \tilde{\mathcal{X}}_i(t|t-1)\xi_i'(t|t-1) \tag{6.52}$$

With this in mind it is clear that the *posterior distribution* can be calculated from

$$\mathrm{p}_X(x(t)|Y_t) \sim \mathcal{N}(\hat{x}(t|t), \tilde{P}(t|t)) = \int \mathrm{p}_X(x(t)|x(t-1) \times \mathrm{p}_X(x(t-1)|Y_{t-1})\, dx(t-1) \tag{6.53}$$

where we have the "usual" (Kalman) *update* relations of Fig. 6.3 starting with the *innovations*

$$e(t) = y(t) - \hat{y}(t|t-1) \tag{6.54}$$

and the *state update*

$$\hat{x}(t|t) = \hat{x}(t|t-1) + \mathcal{K}(t)e(t) \tag{6.55}$$

along with the corresponding *state error covariance update* given by

$$\tilde{P}(t|t) = \tilde{P}(t|t-1) - \mathcal{K}(t)R_{\xi\xi}(t|t-1)\mathcal{K}'(t) \tag{6.56}$$

This completes the *SPBP* algorithm which is summarized in Table 6.1. We have shown how the *SPT* coupled with the *WSLR* can be used to develop this technique in its "plain vanilla" form. We note in passing that there are *no* Jacobians calculated and the nonlinear models are employed directly to transform the σ-points to the new space. Also in the original problem definition (see Sec. 6.2) both process and noise sources, (w, v) were assumed *additive*. The *SPT* 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 $\overline{x}(t) = [x(t) \;\; w(t-1) \;\; v(t)]'$. We chose to ignore the general case to keep the development of the sigma-point processor simple. For more details of the general process see the following references [17, 21, 24]. Let us apply the sigma-point

TABLE 6.1 Discrete Sigma-Point Bayesian Processor (Unscented Kalman Filter) Algorithm

State: Sigma Points and Weights

$$\mathcal{X}_o = \hat{x}(t-1|t-1), \qquad W_o = \frac{\kappa}{(N_x+\kappa)}$$

$$\mathcal{X}_i = \hat{x}(t-1|t-1) + \left(\sqrt{(N_x+\kappa)\,\tilde{P}(t-1|t-1)}\right)_i, \qquad W_i = \frac{1}{2(N_x+\kappa)}$$

$$\mathcal{X}_{i+N_x} = \hat{x}(t-1|t-1) - \left(\sqrt{(N_x+\kappa)\,\tilde{P}(t-1|t-1)}\right)_i, \qquad W_{i+N_x} = \frac{1}{2(N_x+\kappa)}$$

State Prediction

$$\mathcal{X}_i(t|t-1) = \mathbf{a}[\mathcal{X}_i(t-1|t-1)] + \mathbf{b}[u(t-1)] \qquad \text{(nonlinear state process)}$$

$$\hat{x}(t|t-1) = \sum_{i=0}^{2N_x} W_i \mathcal{X}_i(t|t-1) \qquad \text{(state regression)}$$

State Error Prediction

$$\tilde{\mathcal{X}}_i(t|t-1) = \mathcal{X}_i(t|t-1) - \hat{x}(t|t-1) \qquad \text{(state error)}$$

$$\tilde{P}(t|t-1) = \sum_{i=0}^{2N_x} W_i \tilde{\mathcal{X}}_i(t|t-1)\tilde{\mathcal{X}}_i'(t|t-1) + R_{ww}(t-1) \qquad \text{(error covariance prediction)}$$

Measurement: Sigma Points and Weights

$$\hat{\mathcal{X}}_i(t|t-1) = \{\mathcal{X}_i(t|t-1),\ \mathcal{X}_i(t|t-1) + \kappa\sqrt{R_{ww}(t-1)},\ \mathcal{X}_i(t|t-1) - \kappa\sqrt{R_{ww}(t-1)}\}$$

Measurement Prediction

$$\mathcal{Y}_i(t|t-1) = \mathbf{c}[\hat{\mathcal{X}}_i(t|t-1)] \qquad \text{(nonlinear measurement)}$$

$$\hat{y}(t|t-1) = \sum_{i=0}^{2N_x} W_i \mathcal{Y}_i(t|t-1) \qquad \text{(measurement regression)}$$

Residual Prediction

$$\xi_i(t|t-1) = \mathcal{Y}_i(t|t-1) - \hat{y}(t|t-1) \qquad \text{(predicted residual)}$$

$$R_{\xi\xi}(t|t-1) = \sum_{i=0}^{2N_x} W_i \xi_i(t|t-1)\xi_i'(t|t-1) + R_{vv}(t) \qquad \text{(residual covariance regression)}$$

Gain

$$R_{\tilde{x}\xi}(t|t-1) = \sum_{i=0}^{2N_x} W_i \tilde{\mathcal{X}}_i(t|t-1)\xi_i'(t|t-1) \qquad \text{(cross-covariance regression)}$$

$$\mathcal{K}(t) = R_{\tilde{x}\xi}(t|t-1)R_{\xi\xi}^{-1}(t|t-1) \qquad \text{(gain)}$$

State Update

$$e(t) = y(t) - \hat{y}(t|t-1) \qquad \text{(innovation)}$$

$$\hat{x}(t|t) = \hat{x}(t|t-1) + \mathcal{K}(t)e(t) \qquad \text{(state update)}$$

$$\tilde{P}(t|t) = \tilde{P}(t|t-1) - \mathcal{K}(t)R_{\xi\xi}(t|t-1)\mathcal{K}'(t) \qquad \text{(error covariance update)}$$

Initial Conditions

$$\hat{x}(0|0) \quad \tilde{P}(0|0)$$

processor to the nonlinear trajectory estimation problem and compare its performance to the other nonlinear processors discussed previously.

Example 6.3

We revisit the nonlinear trajectory estimation problem of the previous examples with the dynamics specified by the discrete nonlinear process given by

$$x(t) = (1 - 0.05\Delta T)x(t-1) + 0.04\Delta T x^2(t-1) + w(t-1)$$

and corresponding measurement model

$$y(t) = x^2(t) + x^3(t) + v(t)$$

Recall that $v(t) \sim \mathcal{N}(0, 0.09)$, $x(0) = 2.0$, $P(0) = 0.01$, $\Delta T = 0.01\,sec$ and $R_{ww} = 0$. The simulated measurement is shown in Fig. 6.4*b*. The sigma-point processor (*SPBP*) and *XBP* (*EKF*) and *IX-BP* (*IEKF*) (3 iterations) were applied to this problem. We used the square-root implementations of the *XBP* and *IX-BP* in *SSPACK_PC* [10] and compared them to the sigma-point processor in *REBEL* [24]. The results are shown

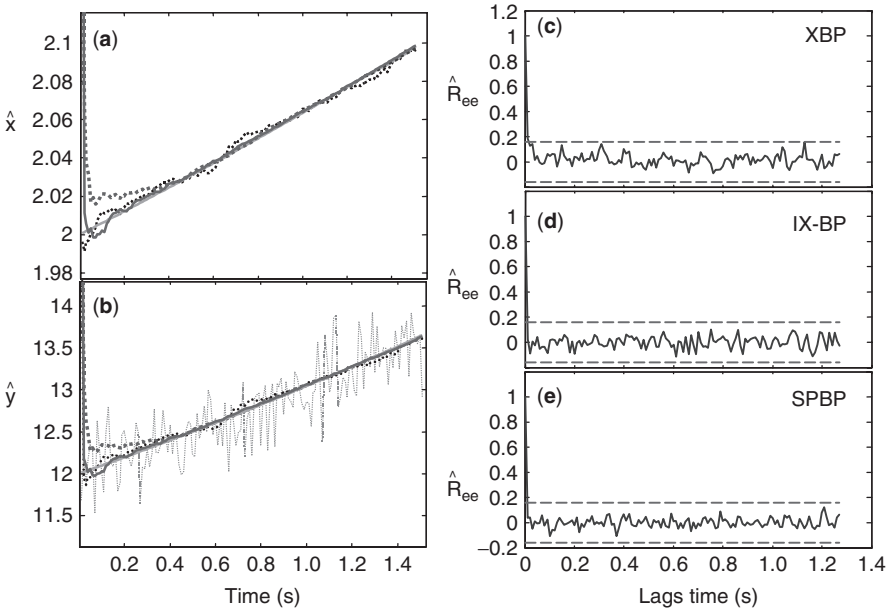


FIGURE 6.4 Nonlinear trajectory estimation. (a) Trajectory (state) estimates using the *XBP* (thick dotted), *IX-BP* (thin dotted) and *SPBP* (thick solid). (b) Filtered measurement estimates using the *XBP* (thick dotted), *IX-BP* (thin dotted) and *SPBP* (thick solid). (c) Zero-mean/whiteness tests for *XBP* ($1.04 \times 10^{-1} < 1.73 \times 10^{-1}$/1% out). (d) Zero-mean/whiteness tests for *IX-BP* ($3.85 \times 10^{-2} < 1.73 \times 10^{-1}$/0% out). (e) Zero-mean/whiteness tests for *SPBP*: ($5.63 \times 10^{-2} < 1.73 \times 10^{-1}$/0% out).

in Fig. 6.4 where we see the corresponding trajectory (state) estimates in *a* and the "filtered" measurements in *b*. From the figures, it appears that all of the estimates are quite reasonable with the sigma-point processor estimate (thick solid line) converging most rapidly to the true trajectory (dashed line). The *XBP* (thick dotted line) appears slightly biased while the *IX-BP* (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 *XBP* and *IX-BP* 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 sigma-point processor is certainly comparable at $(5.63 \times 10^{-2} < 1.73 \times 10^{-1}/0\%$ out). This completes the example. △△△

6.3.1 Extensions of the Sigma-Point Processor

Recently there have been a number of developments in the nonlinear estimation area that are based on the sigma-point (or similar) transformation [29, 30, 32]. Next we briefly mention these approaches keeping in mind that they are members of the "sigma-point family".

The *central difference* Bayesian processor (*CDBP*) or unscented Kalman filter (*UKF*) is based on the Stirling approximation interpolation formula that is essentially a second order Taylor series expansion of the nonlinear random function about its mean. The central differences are used to approximate the first and second order terms of the series. In this sense the processor implicitly employs the *WSLR* used to derive the *SPBP* as before. The resulting σ-points are dependent on the half-step size, $\triangle_x$, rather than the other parameters discussed previously. The *CDBP* is slightly more accurate than the *SPBP*, and also has the advantage of only requiring a single parameter $\triangle_x$ to adjust the spread of the σ-points. For more details, see [29].

Just as with the Kalman filter implementations [10], the *SPBP* also admits the numerically stable "square-root" forms for prediction and updated state covariance matrices. These methods are based on employing the QR decomposition and Cholesky updating. Again this approach offers a slightly more accurate processor as well as reduced computational costs while maintaining their numerical stability. See [29] for further details.

6.4 QUADRATURE BAYESIAN PROCESSORS

The grid-based quadrature Bayesian processor (*QBP*) or equivalently quadrature Kalman filter (*QKF*) is another alternative σ-point approach [30–32]. It uses the Gauss-Hermite numerical integration rule as its basic building block to precisely calculate the sequential Bayesian recursions of Chapter 2 under the Gaussian assumptions. For scalars, the *Gauss-Hermite rule* is given

$$\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-x^2} dx = \sum_{i=1}^{M} W_i \times f(\Delta x_i) \tag{6.57}$$

where equality holds for all polynomials, $f(\cdot)$, of degree up to $2M-1$ and the quadrature (grid) points Δx_i with corresponding weights W_i are determined according to the rule. The *QBP* uses the *WSLR* to linearize the nonlinear transformations through a set of Gauss-Hermite quadrature points *rather* than introducing the set of σ-points. Details of this approach can be found in [32]. We briefly show the approach following the usual recursive forms of the *BP* of the previous chapter. For our nonlinear estimation problem, we have both nonlinear process (a$[\cdot]$) and measurement (c$[\cdot]$) equations that can be expressed in terms of the Gauss-Hermite rule above.

Suppose we have the conditional mean and covariance at time $(t-1)$ based on all of the data up to the same time step. Then the corresponding conditional distribution is $\Pr(x(t-1)|Y_{t-1}) \sim \mathcal{N}(\hat{x}(t-1|t-1), \tilde{P}(t-1|t-1))$. The *QBP* is then given by the following set of recursive Bayesian equations:

$$x_i = \sqrt{\tilde{P}(t-1|t-1)}' \, \Delta x_i + \hat{x}(t-1|t-1) \tag{6.58}$$

where $\sqrt{\tilde{P}}$ is the matrix square root (Cholesky factor: $\tilde{P} = \sqrt{\tilde{P}}' \sqrt{\tilde{P}}$). The *prediction-step*: $\Pr(x(t)|Y_{t-1}) \sim \mathcal{N}(\hat{x}(t|t-1), \tilde{P}(t|t-1))$ is given by

$$\begin{aligned}
\hat{x}(t|t-1) &= \sum_{i=1}^{M} W_i \mathrm{a}[x_i] \\
\tilde{P}(t|t-1) &= \sum_{i=1}^{M} W_i (\mathrm{a}[x_i] - \hat{x}(t|t-1))(\mathrm{a}[x_i] - \hat{x}(t|t-1))' + R_{ww}(t-1)
\end{aligned} \tag{6.59}$$

The *update-step*: $\Pr(x(t)|Y_t) \sim \mathcal{N}(\hat{x}(t|t), \tilde{P}(t|t))$ follows a similar development

$$\overline{x}_i = \sqrt{\tilde{P}(t|t-1)}' \, \Delta x_i + \hat{x}(t|t-1) \tag{6.60}$$

and

$$\begin{aligned}
\hat{x}(t|t) &= \hat{x}(t|t-1) + \mathcal{K}(t)(y(t) - \overline{y}(t)) \quad \text{for } \overline{y}(t) = \sum_{i=1}^{M} W_i \mathrm{c}[\overline{x}_i] \\
\tilde{P}(t|t) &= \tilde{P}(t|t-1) - \mathcal{K}(t)\mathcal{P}_{XY}
\end{aligned} \tag{6.61}$$

where

$$\begin{aligned}
\mathcal{K}(t) &= \mathcal{P}_{XY}(\mathcal{P}_{YY} + R_{vv}(t))^{-1} \\
\mathcal{P}_{XY} &= \sum_{i=1}^{M} W_i (\hat{x}(t|t-1) - \overline{x}_i)(y(t) - \mathrm{c}[\overline{x}_i])' \\
\mathcal{P}_{YY} &= \sum_{i=1}^{M} W_i (y(t) - \mathrm{c}[\overline{x}_i])(y(t) - \mathrm{c}[\overline{x}_i])'
\end{aligned} \tag{6.62}$$

As with the *SPBP*, the *QBP* does *not* linearize the process or measurement models as in the case of the classical nonlinear processors. It calculates the weighted quadrature points in state-space over a fixed grid to estimate the unknown distribution (see [32–34] for more details).

Another powerful approach to nonlinear estimation is the Gaussian sum (mixture) processor [36], which we discuss in more detail next.

6.5 GAUSSIAN SUM (MIXTURE) BAYESIAN PROCESSORS

Another general approach to the Bayesian processing problem is the Gaussian sum (*G-S*) approximation leading to a processor. It has been shown [4, 35, 36] that *any* non-Gaussian distribution can be approximated by a sum of Gaussian distributions, that is,

$$\mathrm{p}_X(x) \approx \sum_{i=1}^{N_g} \mathcal{W}_i \mathrm{p}_i(x) = \sum_{i=1}^{N_g} \mathcal{W}_i \mathcal{N}(x : \mu_x(i), \Sigma_x(i)) \quad \text{for} \sum_{i=1}^{N_g} \mathcal{W}_i = 1 \text{ and } \mathcal{W}_i \geq 0 \, \forall i \tag{6.63}$$

a mixture of Gaussian distributions with $\{\mathcal{W}_i\}$ the set of mixing coefficients or weights. Clearly, this approach can be implemented with a bank of parallel classical processors (*LZ-BP*, *XBP*, *IX-BP*) or with any of the modern *SPBP* family just discussed in order to estimate the mean and variance of the individual N_g-processors. In addition, the particle filtering algorithms to be discussed in the next chapter can also be incorporated into a Gaussian mixture framework—that is what makes this approach important [37]. Before we develop the processor, let us investigate some of the underlying properties of the Gaussian sum or equivalently Gaussian mixture (*G-M*) that makes it intriguing.

The fundamental problem of approximating a probability distribution or density evolves from the idea of *delta families* of functions, that is, families of functions that converge to a delta or impulse function as the parameter that uniquely characterizes that family converges to a limiting value. Properties of such families are discussed in [35]. The most *important* result regarding Gaussian sums is given in a theorem, which states that a probability density function formed by

$$\mathrm{p}_X(x) = \int_{-\infty}^{\infty} \mathrm{p}(\alpha)\delta(x - \alpha)\, d\alpha \tag{6.64}$$

converges uniformly to $\mathrm{p}_X(x)$ [35]. The Gaussian density forms a delta family with parameter σ represented by

$$\delta_\sigma(x) = \mathcal{N}(x : 0, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{x^2}{2\sigma^2}\right\} \tag{6.65}$$

which satisfies the requirement of a positive delta family as $\sigma \to 0$, that is, as the variance parameter approaches zero in the limit, the Gaussian density approaches a

delta function; therefore, we have that

$$\mathrm{p}_X(x) = \int_{-\infty}^{\infty} \mathrm{p}(\alpha)\mathcal{N}_\sigma(x-\alpha)\,d\alpha \tag{6.66}$$

This is the *key result* that forms the underlying basis of Gaussian sum or mixture approximations, similar to the idea of an empirical probability distribution

$$\hat{\Pr}(x) \approx \sum_{i=1}^{N_x} \mathcal{W}_i \delta(x - x_i) \quad \Leftrightarrow \quad \hat{\Pr}(x) \approx \sum_{i=1}^{N_g} \mathcal{W}_i \mathcal{N}(x_i : \mu_x(i), \Sigma_x(i)) \tag{6.67}$$

as the variance $\sigma_x \to 0$. The Gaussian converges to an impulse located at its mean, μ_x. The Gaussian mixture distributions have some interesting properties that we list below (see [35, 36] for more details):

- Mean: $\mu_g = \sum_{i=1}^{N_g} \mathcal{W}_i \mu_x(i)$
- Covariance: $\Sigma_g = \sum_{i=1}^{N_g} \mathcal{W}_i(\Sigma_x(i) + \mu_x(i))$
- Skewness: $\gamma_g = \sum_{i=1}^{N_g} \mathcal{W}_i(\mu_x(i) - \mu_g)(3\Sigma_x(i) + (\mu_x(i) - \mu_g)^2 \times \Sigma_g^{\frac{-3}{2}})$
- Kurtosis: $\kappa_g = \sum_{i=1}^{N_g} \mathcal{W}_i(3\Sigma_x^2(i) + 6(\mu_x(i) - \mu_g)^2\Sigma_x(i) + \mu_g^4)\Sigma_g^{-2} - 3$

Next, let us see how this property can be utilized to develop a Bayesian processor. The processor we seek evolves from the sequential Bayesian paradigm of Sec. 2.5 given by

$$\Pr(x(t)|Y_t) = \frac{\Pr(y(t)|x(t)) \times \Pr(x(t)|Y_{t-1})}{\Pr(y(t)|Y_{t-1})} \tag{6.68}$$

with the corresponding prediction-step

$$\Pr(x(t)|Y_{t-1}) = \int \Pr(x(t)|x(t-1)) \times \Pr(x(t-1)|Y_{t-1})\,dx(t-1) \tag{6.69}$$

Assume at time t that we approximate the predictive distribution by the Gaussian mixture

$$\Pr(x(t)|Y_{t-1}) \approx \sum_{i=1}^{N_g} \mathcal{W}_i(t-1)\mathcal{N}(x(t) : \overline{x}_i(t), \overline{P}_i(t)) \tag{6.70}$$

Then substituting this expression into the filtering posterior of Eq. 6.68 we obtain

$$\Pr(x(t)|Y_t) = \frac{\Pr(y(t)|x(t))}{\Pr(y(t)|Y_{t-1})} \times \sum_{i=1}^{N_g} \mathcal{W}_i(t-1)\mathcal{N}(x(t) : \overline{x}_i(t), \overline{P}_i(t)) \tag{6.71}$$

For clarity in this development we constrain both process and measurement models to additive Gaussian noise sources[2] resulting in the nonlinear state-space approximate

[2] This is not necessary but enables the development and resulting processor relations to be much simpler. In the general case both noise sources can be modeled by Gaussian mixtures as well (see [35] for details).

GM representation of Sec. 4.8, that is, the process model

$$x(t) = \mathbf{a}[x(t-1)] + w(t-1) \quad w \sim \mathcal{N}(0, R_{ww}(t)) \tag{6.72}$$

and the measurement model

$$y(t) = \mathbf{c}[x(t)] + v(t) \quad v \sim \mathcal{N}(0, R_{vv}(t)) \tag{6.73}$$

with *Gaussian prior*, $x(0) \sim \mathcal{N}(x(0), P(0))$. Applying the fundamental convergence theorem of Eq. 6.64, the Gaussian mixture distribution of the posterior can be approximated by

$$\Pr(x(t)|Y_t) \approx \sum_{i=1}^{N_g} \mathcal{W}_i(t)\mathcal{N}(x(t) : x_i(t), P_i(t)) \tag{6.74}$$

and converges uniformly in $x(t)$ and $y(t)$ as $P_i(t) \to 0$ for $i = 1, \ldots, N_g$. Therefore, $\{x_i(t), P_i(t)\}$ can be estimated from any of the classical (nonlinear) processors developed in Chapter 5 or the modern σ-point processor of this chapter. We choose to use the *SPBP* technique[3] of Table 6.1 to provide a "bank" of N_g-parallel processors required to estimate each member of the Gaussian mixture such that

$$\begin{aligned}
\overline{y}_i(t) &= f(\overline{x}_i(t)) \\
e_i(t) &= y(t) - \overline{y}_i(t) \\
R_{e_ie_i}(t) &= g(\overline{P}_i(t), R_{vv}(t)) \\
x_i(t) &= \overline{x}_i(t) + \mathcal{K}_i(t)e_i(t) \\
P_i(t) &= \overline{P}_i(t) - \mathcal{K}_i(t)R_{e_ie_i}(t)\mathcal{K}_i'(t)
\end{aligned}$$

where $f(\cdot)$, $g(\cdot)$ are functions that are derived from the *SPBP* of Table 6.1 and the weights[4] (mixing coefficients) of the individual mixture Gaussians are updated

$$\mathcal{W}_i(t) = \frac{\mathcal{W}_i(t-1) \times \mathcal{N}(y(t) : \overline{y}_i(t), R_{e_ie_i}(t))}{\sum_{i=1}^{N_g} \mathcal{W}_i(t-1) \times \mathcal{N}(y(t) : \overline{y}_i(t), R_{e_ie_i}(t))} \tag{6.75}$$

Once we have performed the parallel *SPBP* estimation using the Gaussian mixtures, we can estimate the statistic of interest from the posterior: the conditional mean as

$$\begin{aligned}
\hat{x}(t|t) &= E\{x(t)|Y_t\} = \int x(t)\hat{\Pr}(x(t)|Y_t)\,dx(t) \\
&\approx \sum_{i=1}^{N_g} \mathcal{W}_i(t) \int x(t)\mathcal{N}(x(t) : x_i(t), P_i(t))\,dx(t)
\end{aligned}$$

[3] We symbolically use the *SPBP* algorithm and ignore the details (sigma-points, etc.) of the implementation in this presentation to avoid the unnecessary complexity.

[4] The detailed derivation of these expressions can be found in Alspach [36] and Anderson [4]. Both references use the uniform convergence theorem to develop the posterior representation given above.

Now using the sifting property of the implied delta function, the Gaussian sum converges uniformly ($P_i \to 0, \mathcal{N}(\cdot) \to \delta(\cdot)$) to give

$$\hat{x}(t|t) = \sum_{i=1}^{N_g} \mathcal{W}_i(t) x_i(t) \tag{6.76}$$

Defining the estimation error as

$$\tilde{x}(t|t) := x(t) - \hat{x}(t|t)$$

and the corresponding error covariance as $\tilde{P}(t|t) := \text{cov}(\tilde{x}(t|t))$ as before in Chapter 5, the approximated state error covariance is given by

$$\tilde{P}(t|t) = \sum_{i=1}^{N_g} \mathcal{W}_i(t)[\overline{P}_i(t) + \text{cov}(x_i(t) - \hat{x}(t|t))] \tag{6.77}$$

Thus, updating consists of a bank of N_g-parallel *SPBP* to estimate the means and covariance $\{x_i(t), P_i(t)\}$ required for the conditional statistics of Eqs. 6.76 and 6.77. This set of relations constitute the *update-step* of the Gaussian sum processor. Next let us briefly develop the prediction-step.

With the availability of the posterior $\hat{\Pr}(x(t)|Y_t)$ and the process model of Eq. 6.72, the one-step prediction distribution can also be estimated as a Gaussian mixture. Using the filtering posterior and *SPBP* relations of Table 6.1, we have that

$$\Pr(x(t+1)|Y_t) \approx \sum_{i=1}^{N_g} \mathcal{W}_i(t) \int x(t) \mathcal{N}(x(t+1) : \overline{x}_i(t+1), \overline{P}_i(t+1)) \tag{6.78}$$

and using the N_g-*SPBP*, we have

$$\begin{aligned} \overline{x}_i(t+1) &= \mathbf{a}[x_i(t)] \\ \overline{P}_i(t+1) &= h(P_i(t)) + R_{ww}(t) \end{aligned}$$

for $h(\cdot)$ a function of the *SPBP* parameters in Table 6.1. These relations lead to the one-step prediction conditional mean and covariance (as before)

$$\begin{aligned} \hat{x}(t+1|t) &= \sum_{i=1}^{N_g} \mathcal{W}_i(t) x_i(t+1) \\ \tilde{P}(t+1|t) &= \sum_{i=1}^{N_g} \mathcal{W}_i(t)[\overline{P}_i(t+1 + \text{cov}(x_i(t+1) - \hat{x}(t+1|t))] \end{aligned} \tag{6.79}$$

to complete the algorithm.

Next we consider the application of nonlinear processors to a tracking problem.

6.6 CASE STUDY: 2D-TRACKING PROBLEM

In this section we investigate the design of nonlinear *BP* to solve a two-dimensional (2D) tracking problem. The hypothetical scenario discussed will demonstrate the applicability of these processors to solve such a problem and demonstrate the "basic" thinking behind constructing such a problem and solution. In contrast to the "bearings-only" problem discussed in Chapter 5, let us investigate the tracking of a large tanker entering a busy harbor with a prescribed navigation path. In this case the pilot on 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 frequently signaling accurate information about its current position. The objective is to safely dock the tanker without any incidents. We observe that the ship's path should track the prescribed trajectory (Cartesian coordinates) shown in Fig. 6.5 which corresponds to the instantaneous *XY*-positions (versus time) shown.

Our fictitious measurement instrument (e.g., low ground clutter phased array radar or a satellite communications receiver) is assumed to instantly report on the tanker

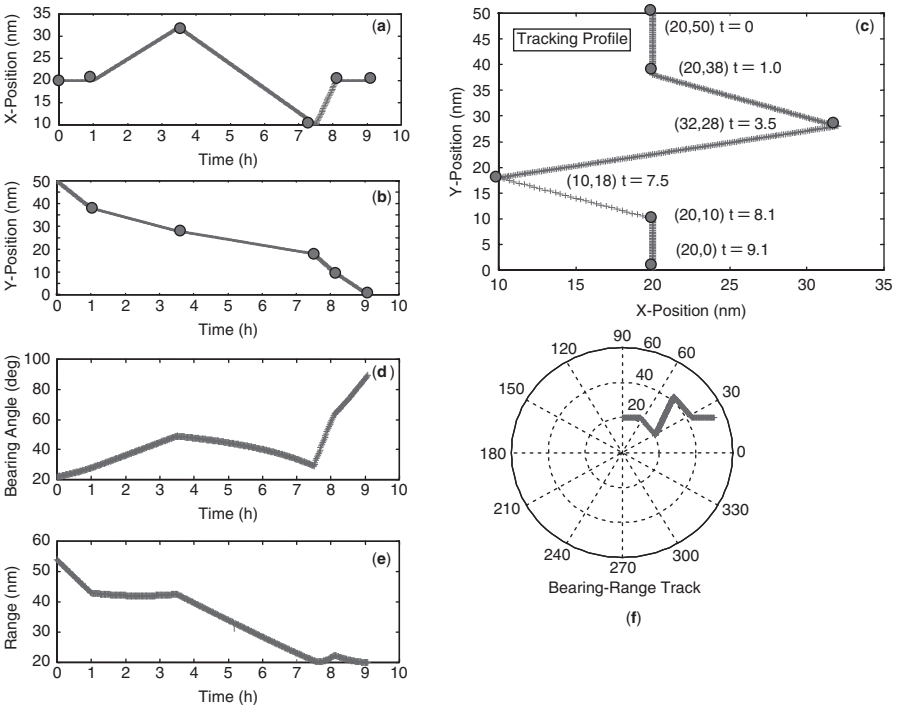


FIGURE 6.5 Tanker ground track geometry for the harbor docking application: (a) Instantaneous X-position (nm). (b) Instantaneous Y-position (nm). (c) Filed XY-path (nm). (d) Instantaneous bearing (deg). (e) Instantaneous range (nm). (f) Polar bearing-range track from sensor measurement.

position in bearing and range with high accuracy. 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 Sec. 5.5) 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 $u' := [-\Delta V_{x_o} - \Delta V_{y_o}]$.

Using 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

$$\begin{aligned} x(t) = &\begin{bmatrix} 1 & 0 & \Delta T & 0 \\ 0 & 1 & 0 & \Delta T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x(t-1) \\ &+ \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -\Delta V_{x_o}(t-1) \\ -\Delta V_{y_o}(t-1) \end{bmatrix} + w(t-1) \end{aligned} \tag{6.80}$$

with the corresponding measurement model given by

$$y(t) = \begin{bmatrix} \arctan \frac{x_2(t)}{x_1(t)} \\ \sqrt{x_1^2(t) + x_2^2(t)} \end{bmatrix} + v(t)$$

for $w \sim \mathcal{N}(0, R_{ww})$ and $v \sim \mathcal{N}(0, R_{vv})$.

The *SSPACK_PC* software [23] was used to simulate this Gauss-Markov system for the tanker path and the results are shown in Fig. 6.6. In this scenario, we assume the instrument is capable of making measurements every $\Delta T = 0.02\,hr$ with a bearing precision of ± 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, for example, V_y going from $-12\,k$ to $-4\,k$ is an incremental change of $+8\,k$ corresponding to $\Delta V_{y_o} = [8\ \ 1.5\ \ -10.83\ \ 3.33]$ knots and $\Delta V_{x_o} = [0\ \ 4.8\ \ -10.3\ \ 22.17]$ knots. These impulses (changes) occur at time fiducials of $t = [0\ \ 1\ \ 3.5\ \ 7.5\ \ 8.1\ \ 9.1]\ hr$ corresponding to the filed harbor path depicted in the figure. Note that the velocity changes are impulses of height $(\Delta V_x, \Delta V_y)$ 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.

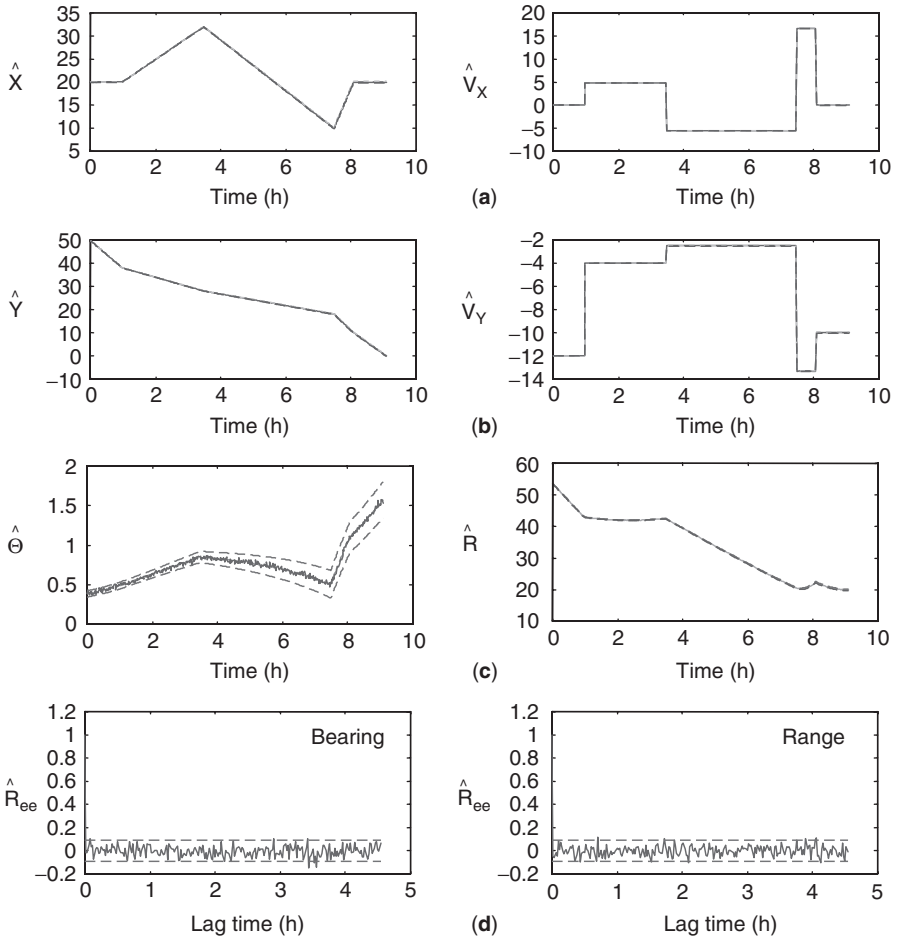


FIGURE 6.6 XBP (EKF) design for harbor docking problem (input known). (a) X-position and velocity estimates with bounds (0% out). (b) Y-position and velocity estimates with bounds (0% and 3% out). (c) Bearing and range estimates with bounds (1% and 2% out). (d) Innovations zero-mean/whiteness tests for bearing ($6 \times 10^{-4} < 26 \times 10^{-1}$ and 3% out) and range ($2 \times 10^{-4} < 42 \times 10^{-4}$ and 5% out).

The simulated bearing measurements are generated using the initial conditions $x'(0) := [20\,\text{nm}\ \ 50\,\text{nm}\ \ 0\,\text{k}\ \ -12\,\text{k}]$ and $R_{ww} = \text{diag}[1 \times 10^{-6}, 1 \times 10^{-6}]$ with the corresponding initial covariance given by $\tilde{P}(0) = \text{diag}[1 \times 10^{-6}, 1 \times 10^{-6}]$. The Jacobian matrices derived from the Gauss-Markov model above are shown below:

$$A[x] = A \quad \text{and} \quad C[x] = \begin{bmatrix} \frac{x_2(t)}{R^2(t)} & \frac{-x_1(t)}{R^2(t)} & 0 & 0 \\ \frac{x_1(t)}{R(t)} & \frac{x_2(t)}{R(t)} & 0 & 0 \end{bmatrix}$$

The *XBP*, *IXBP*, *LZ-BP*, and *SPBP* were executed under the constraint that all of the *a priori* information for the tanker harbor path is "known". Each of the processors performed almost identically. A representative realization output for the *XBP* is shown in Fig. 6.6. In *a* and *b* we observe the estimated states X ($\sim$0% out), Y ($\sim$0% out), V_x ($\sim$0% out), V_y ($\sim$3% out)[5]. Note that the velocities are piecewise constant functions with step changes corresponding to the impulsive incremental velocities. The filtered measurements: bearing ($\sim$1% out) and range ($\sim$2% out) are shown in Fig. 6.6*c* with the resulting innovations zero-mean/whiteness tests depicted in *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.

Next, we investigate the realistic 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* now the unknown input which is a solution to the deconvolution problem [25]. It is also a problem that is ill-conditioned especially, since $u(t)$ is impulsive.

In any case we ran the nonlinear *BP* algorithms over the simulated data and the best results were obtained using the *LZ-BP* and *SPBP*. This is expected, since we used the exact state reference trajectories or filed paths, but not the input. Note that the other nonlinear *BP* have no knowledge of this trajectory inhibiting their performance in this problem. The results are shown in Fig. 6.7, where we observe the state estimates as before. We note that the position estimates appear reasonable, primarily because of the reference trajectories. The *LZ-BP* is able to compensate for the unknown impulsive input with a time 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-BP* response even with the exact step changes available. These lags are more vividly shown in the bearing estimate of Fig. 6.7c 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-BP* 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).

[5] Here "% out" means the percentage of samples exceeding the confidence bounds.

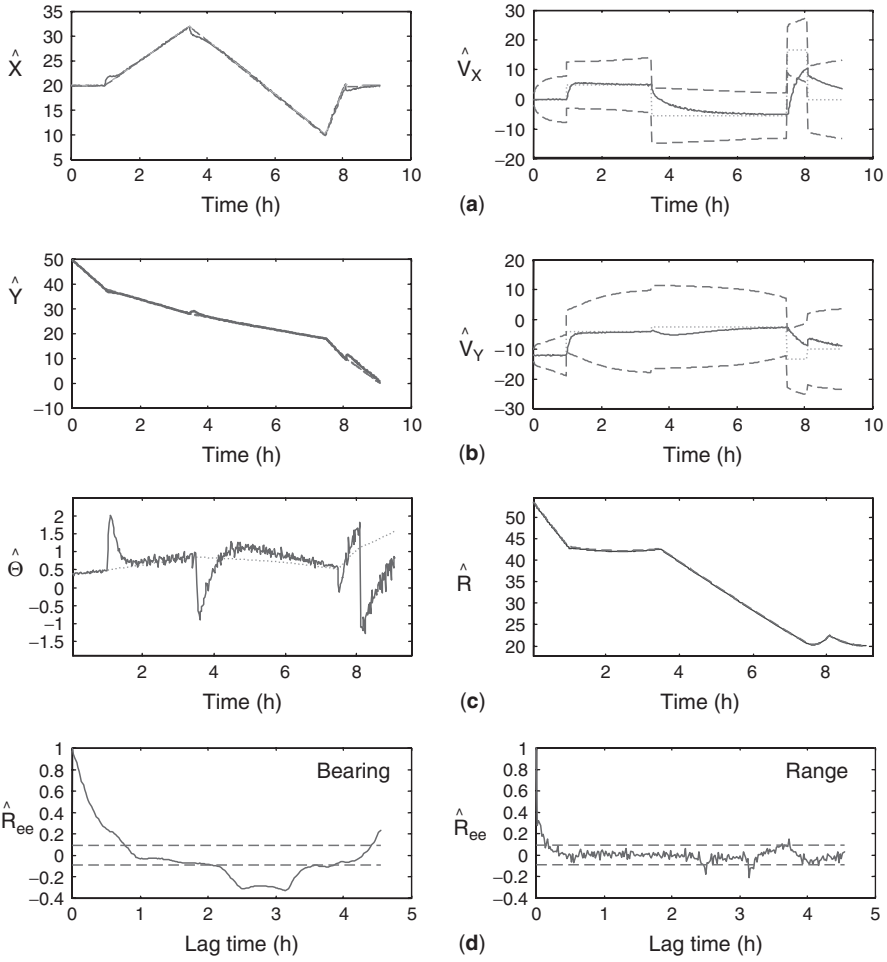


FIGURE 6.7 LZBP design for harbor docking problem (input unknown). (a) X-position and velocity estimates with bounds (68% and 4% out). (b) Y-position and velocity estimates with bounds (49% and 1% out). (c) Bearing and range estimates with bounds (0% and 3% out). (d) Innovations zero-mean/whiteness tests 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).

We also designed the *SPBP* (*UKF*) to investigate its performance on this problem and its results were quite good[6] as shown in Fig. 6.8. The processor does not perform a model linearization but a *statistical linearization* instead, it is clear from the figure shown that it performs better than any of the other processors for this problem. In Fig. 6.8a–d, we see that the *XY* position estimates "track" the data very well while

[6] We used noisier simulation data for this run than that for the *LZ-BP* with $R_{vv} = \text{diag} = [4 \times 10^{-4}$ $5 \times 10^{-1}]$ providing a more realistic measurement uncertainty.

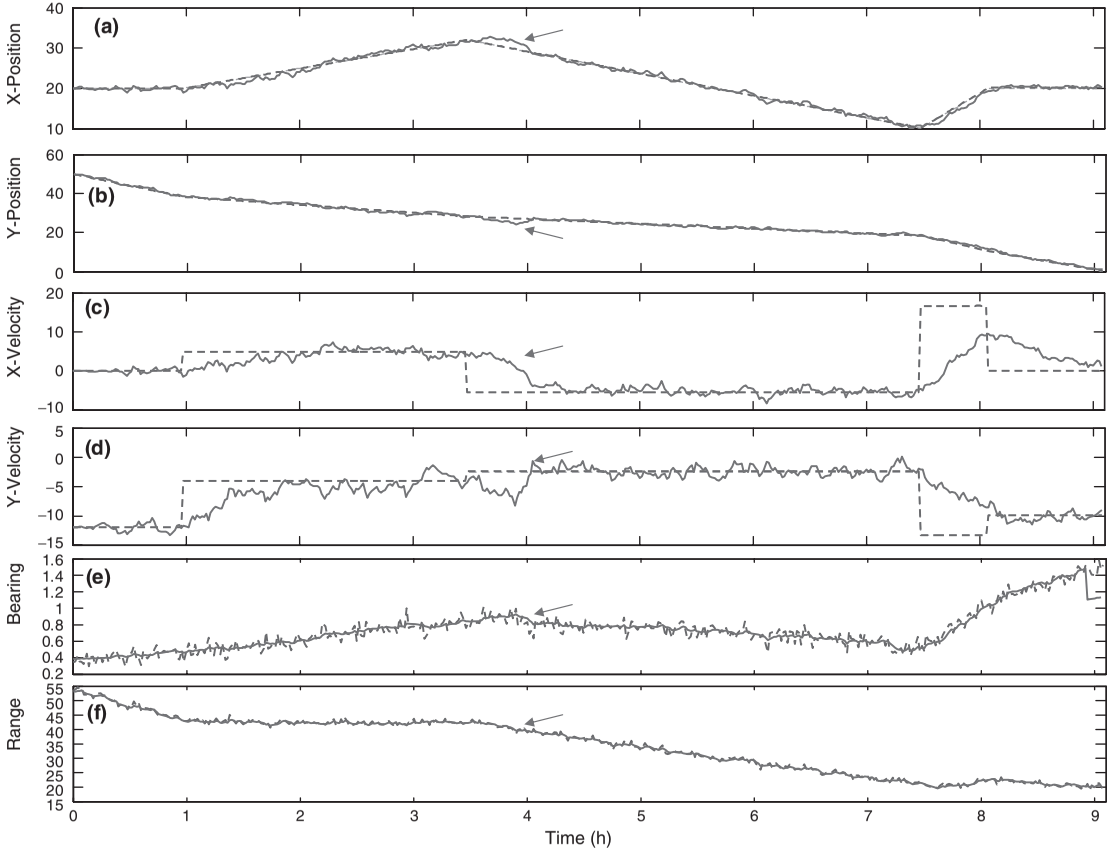


FIGURE 6.8 SPBP (UKF) design for harbor docking problem (input unknown). (a) X-position estimate. (b) Y-position estimate. (c) X-velocity estimate. (d) Y-velocity estimate. (e) Bearing estimate (zero mean/whiteness: $3.3 \times 10^{-3} < 1.2 \times 10^{-1}$ and 4.7% out). (f) Range estimate (zero mean/whiteness: $6.5 \times 10^{-3} < 1.2 \times 10^{-1}$ and 4.7% out).

the XY-velocity estimates are somewhat nosier due to the abrupt changes (steps) tuning values of the process noise covariance terms. In order to be able to track the step changes, the process noise covariance could be increased even further at the cost of nosier estimates. The *SPBP* tracks the estimated bearing and range reasonably well as shown in figure with a slight loss of bearing track toward the end of the time sequence. These results are demonstrated by the zero-mean/whiteness test results of the corresponding innovations sequences. The bearing innovation statistics are: $3.3 \times 10^{-3} < 1.2 \times 10^{-1}$ and 4.7% out and the corresponding range innovation statistics given by: $6.5 \times 10^{-3} < 1.2 \times 10^{-1}$ and 4.7% out. Both indicate a tuned processor. These are the best results of all of the nonlinear processors applied. This completes the case study.

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 *SPBP* appears to provide superior performance, but in the real-world tracking problem (as discussed in Sec. 5.5) when this information on the target is not available, then the *XBP* and *IX-BP* 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 [26, 27] or in the form of a set of models, each with its own independent processor [8]. One might also consider using estimator/smoothers as in the seismic case [7] because of the unknown impulsive input. In any case, this is a challenging problem that much work has been accomplished, the interested reader should consult Ref. [8] and the references cited within.

6.7 SUMMARY

In this chapter we have developed the "modern" sigma-point Bayesian processor (*SPBP*) or equivalently, the unscented Kalman filter (*UKF*), from the basic principles of weighted linear stochastic linearization (*WSLR*) and σ-point transformations (*SPT*). We extended the results for multivariate Gaussian distributions and calculated the corresponding σ-points and weights. Once determined, we developed the *SPBP* by extrapolating the *WSLR* and *SPT* approaches coupled to the usual Kalman filter recursions. Grid-based quadrature and Gaussian sum (*G-S*) processors were also discussed and developed using the *SPBP* formulation to demonstrate a distribution approximation approach leading to the particle filter formulation of the next chapter. Examples were developed throughout to demonstrate the concepts ending in a hypothetical investigation based on tracking a tanker entering a busy harbor. We summarized the results by applying some of the processors to the case study implementing a 2D-tracking filter.

MATLAB NOTES

SSPACK_PC is a third-party toolbox in *MATLAB* that can be used to design model-based signal processors [10, 23]. This package incorporates the major *nonlinear MBP* algorithms discussed in this chapter—all implemented in the *UD*-factorized form [38] 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 post-processor (**SSPOST**) which 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 *UKF* including the unscented transformations [24]. It also has included the new particle filter designs as discussed in [28] (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. Sage and J. Melsa, *Estimation Theory with Applications to Communications and Control* (New York: McGraw-Hill, 1971).
3. A. Gelb, Ed., *Applied Optimal Estimation* (Boston: MIT Press, 1975).
4. B. Anderson and J. Moore, *Optimal Filtering* (Englewood Cliffs, NJ: Prentice-Hall, 1979).
5. P. Maybeck, *Stochastic Models, Estimation, and Control* (New York: Academic Press, 1979).
6. M. Grewal and A. Andrews, *Kalman Filtering: Theory and Practice* (Englewood Cliffs, NJ: Prentice-Hall, 1993).
7. J. Mendel, *Lessons in Estimation Theory for Signal Processing, Communications and Control* (Englewood Cliffs, NJ: Prentice-Hall, 1995).
8. Y. Bar-Shalom and X. Li, *Estimation and Tracking: Principles, Techniques, and Software* (Norwood, MA: Artech House, 1993).
9. B. Bell and F. Cathey, "The iterated kalman filter update as a Gauss-Newton method," *IEEE Trans. Autom. Contr.*, **AC-38**, 2, 294–297, 1993.
10. J. Candy, *Model-Based Signal Processsing* (Hoboken, NJ: John Wiley/IEEE Press, 2006).
11. D. Simon, *Optimal State Estimation: Kalman, H_∞ and Nonlinear Approaches* (Hoboken, NJ: John Wiley, 2006).

12. S. Julier, J. Uhlmann and H. F. Durrant-Whyte, "A new approach for filtering in nonlinear systems," *Proc. Am. Contr. Confr.*, 1995.
13. S. Julier and J. Uhlmann, *A General Method for Approximating Nonlinear Transformations of Probability Distributions*, **Univ. of Oxford Report**, 1996.
14. S. Julier, J. 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.
15. S. Julier and J. Uhlmann, "Unscented filtering and nonlinear estimation," *Proc. IEEE*, **92**, 3, 401–422, 2004.
16. S. Julier and J. Uhlmann, "A new extension of the Kalman filter to nonlinear systems," *Proc. SPIE Confr.*, Orlando, FL, 1997.
17. E. Wan and R. van der Merwe, "The unscented Kalman filter for nonlinear estimation," *Proc. IEEE Sumpos. Adaptive Sys. for Sig. Proc., Comm. and Contr.*, Lake Louise, Alberta, 2000.
18. R. van der Merwe, A. Doucet, N. de Freitas and E. Wan, "The unscented particle filter," *Cambridge University Tech. Report*, CUED/F-INFENG-380, 2000.
19. T. Lefebvre, H. Bruyninckx and J. DeSchutter, Comments on "A new method for the nonlinear transformation of means and covariances in filters and estimators," *IEEE Trans. Autom. Contr.*, **47**, 8, 1406–1409, 2002.
20. S. Julier and J. Uhlmann, "A consistent, debiased method for converting between polar and Cartesian coordinate systems," *Proc. SPIE Confr.*, 1997.
21. S. Haykin, *Kalman Filtering and Neural Networks* (New York: John Wiley, 2001).
22. A. Smith and A. Gelfand, "Bayesian statistics without tears: A sampling-resampling perspective," *Am. Statistic.*, **46**, 2, 84–88, 1992.
23. 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).
24. R. van der Merwe and E. Wan, *REBEL: Recursive Bayesian Estimation Library University of Oregon* (see **http://choosh.ece.ogi.edu/rebel** for more details), 2002.
25. J. Candy and J. Zicker, "Deconvolution of noisy transient signals:A Kalman filtering application," *LLNL Rep.*, UCID-87432, and *Proc. of CDC Confr.*, Orlando, 1982.
26. V. Aidala and S. Hammel, "Utilization of modified polar-coordinates for bearings-only tracking," *IEEE Trans. Autom. Contr.*, **AC-28**, 283–294, 1983.
27. V. Aidala, "Kalman filter behavior in bearings-only velocity and position estimation," *IEEE Trans. Aerosp. Electron. Syst.*, **AES-15**, 29–39, 1979.
28. S. Haykin and N. de Freitas, Eds., "Sequential state estimation: from Kalman filters to particle filters," *Proc. IEEE*, **92**, 3, 399–574, 2004.
29. R. van der Merwe and E. Wan, "Sigma-Point Kalman filters for probabilistic inference in dynamic state-space models," PH.D. dissertation, OGI School Sci. Eng., Beaverton, OR Online: **http://www.cse.ogi.edu/rudmerwe/pubs/, 2004**.
30. K. Ito and K. Xiong, "Gaussian filters for nonlinear filtering problems," *IEEE Trans. Autom. Control*, **45**, 5, 910–927, 2000.
31. R. Bucy and K. Senne, "Digital synthesis of nonlinear filters," *Automatica*, **7**, 3, 287–298, 1971.
32. I. Arasaratnam, S. Haykin and R. Elliott, "Discrete-time nonlinear filtering algorithms using Gauss-Hermite quadrature," *Proc. IEEE*, **95**, 5, 953–976, 2007.

33. K. Ito and K. Xiong, "Gaussian filters for nonlinear filtering problems," *IEEE Trans. Autom. Control*, **45**, 5, 910–927, 2000.
34. N. Cui, L. Hong and J. Layne, "A comparison of nonlinear filtering approaches with an application to ground target tracking," *Signal Proc.*, **85**, 1469–1492, 2005.
35. H. Sorenson and D. Alspach, "Recursive Bayesian estimation using Gaussian sums," *Automatica*, **7**, 465–479, 1971.
36. D. Alspach and H. Sorenson, "Nonlinear Bayesian estimation using Gaussian sum approximations," *IEEE Trans. Autom. Control*, **17**, 4, 439–448, 1972.
37. J. Kotecha and P. Djuric, "Gaussian sum particle filtering," *IEEE Trans. Signal Proc.*, **51**, 10, 2602–2612, 2003.
38. G. Bierman, *Factorization Methods of Discrete Sequential Estimation* (New York: Academic Press, 1977).

PROBLEMS

6.1 Let x_1 and x_2 be *i.i.d.* with distribution $\mathcal{N}(0, 1)$. Suppose $y = x_1^2 + x_2^2$, then

(*a*) What is the distribution of y, $\mathrm{p}_Y(y)$?

(*b*) Suppose $E\{y\} = 2$ and $\sigma_y^2 = 4$, using the sigma point transformation what are the sigma-points for $\mathbf{x} = [\mathbf{x_1} \quad \mathbf{x_2}]'$?

(*c*) What are the sigma-points for y?

6.2 Suppose $x \sim \mathcal{N}(0, 1)$ and $y = x^2$,

(*a*) What is the distribution of y, $\mathrm{p}_Y(y)$?

(*b*) What is the Gaussian approximation of the mean and variance of $\mathrm{p}_Y(y)$? (*Hint*: Use linearization)

(*c*) What is the sigma-point transformation and corresponding mean and variance estimates of $\mathrm{P}_Y(y)$?

6.3 From the following set of nonlinear system models, develop the *SPBP* algorithm for each:

(*a*) *Synchronous (unsteady) motor:* $\ddot{x}(t) + C\dot{x}(t) + p \sin x(t) = L(t)$

(*b*) *Duffing equation:* $\ddot{x}(t) + \alpha x(t) + \beta x^3(t) = F \cos \omega t)$

(*c*) *Van der Pol equation:* $\ddot{x}(t) + \epsilon \dot{x}(t)[1 - \dot{x}^2(t)] + x(t) = m(t)$

(*d*) *Hill equation:* $\ddot{x}(t) - \alpha x(t) + \beta p(t) x(t) = m(t)$

6.4 Suppose 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}. Develop the *SPBP* for this process. Suppose the parameters ω and α are

unknown, develop the *SPBP* such that the parameters are jointly estimated along with the states. (*Hint*: augment the states and parameters to create a new state vector).

6.5 The Mackey-Glass time delay differential equation is given by

$$\begin{aligned}\dot{x}(t) &= \frac{\alpha x(t-\tau)}{1+x(t-\tau)^N} - \beta x(t) + w(t)\\ y(t) &= x(t) + v(t)\end{aligned}$$

where α, β 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 [21, 29]).

Develop the *SPBP* for this process.

6.6 Assume that a target is able to maneuver, that is, we assume that the target velocity satisfies a first-order *AR* model given by:

$$v_\tau(t) = -\alpha v_\tau(t-1) + w_\tau(t-1) \quad \text{for } w \sim \mathcal{N}(0, R_{w_\tau w_\tau})$$

(*a*) Develop the Cartesian tracking model for this process.

(*b*) Develop the corresponding *SPBP* assuming all parameters are known *a priori*.

(*c*) Develop the corresponding *SPBP* assuming α is unknown.

6.7 Nonlinear processors can be used to develop neural networks used in many applications. A generic neural network behavior is governed by the following relations:

$$\begin{aligned}x(t) &= x(t-1) + w(t-1)\\ y(t) &= c[x(t), u(t), \alpha(t)] + v(t)\end{aligned}$$

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}.

Develop the *SPBP* for this process.

6.8 Consider the problem of estimating a random signal from an AM modulator characterized by

$$\begin{aligned}s(t) &= \sqrt{2P}a(t)\sin\omega_c t\\ r(t) &= s(t) + v(t)\end{aligned}$$

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}. The discrete-time Gauss-Markov model for this system was developed (chapter 5 problems) from the continuous-time representation using first differences, then:

(*a*) Develop the *SPBP*.

(*b*) Assume the carrier frequency, ω_c is unknown. Develop the *SPBP* for this process.

6.9 Develop the Gaussian sum (*G-S*) processor algorithm using the *XBP* instead of the *SPBP*. In the literature, this is the usual approach that is used. How does the overall algorithm differ? What are the apparent pitfalls involved in this approach compared to the *SPBP* approach?

6.10 We are given a sequence of data and know it is Gaussian with an unknown mean and variance. The distribution is characterized by $y \sim \mathcal{N}(\mu, \sigma^2)$.

(*a*) Formulate the problem in terms of a state–space representation. (*Hint*: Assume that the measurements are modeled in the usual manner (scale by standard deviation and add mean to a $\mathcal{N}(0,1)$ "known" sequence, say $\nu(t)$).

(*b*) Using this model, develop the *SPBP* technique to estimate the model parameters.

(*c*) Synthesize a set of data of 2000 samples at a sampling interval $dt = 0.01$ with process covariance, $R_{ww} = \text{diag}[1 \times 10^{-5},\ 1 \times 10^{-6}]$ and measurement noise of $R_{vv} = 1 \times 10^{-6}$ with $\bar{x}(0) = [\sqrt{20} \quad 3]'$.

(*d*) Develop the *SPBP* algorithm and apply it to this data, show the performance results (final parameter estimates, etc.). That is, the find the best estimate of the parameters defined by $\Theta := [\mu \quad \sigma]'$ using the *SPBP* approach. Show the mathematical steps in developing the technique and construct simple *SPBP* to solve.

7

PARTICLE-BASED BAYESIAN STATE–SPACE PROCESSORS

7.1 INTRODUCTION

In this chapter we develop particle-based processors using the state–space representation of signals and show how they evolve from the Bayesian perspective using their inherent Markovian structure along with importance sampling techniques as our basic construct. Particle filters offer an alternative to the Kalman model-based processors discussed in the previous chapters possessing the capability not just to characterize unimodal distributions but also to characterize multimodal distributions. We first introduce the generic state–space particle filter (*SSPF*) and investigate some of its inherent distributions and implementation requirements. We develop a generic sampling-importance-resampling (*SIR*) processor and then perhaps its most popular form—the "bootstrap" particle filter. Next we investigate the resampling problem and some of the more popular resampling techniques also incorporated into the bootstrap filter from necessity. The bootstrap and its variants are compared to the classical and modern processors of the previous chapters. Finally, we apply these processors to a variety of problems and evaluate their performance using statistical testing as part of the design methodology.

7.2 BAYESIAN STATE–SPACE PARTICLE FILTERS

Particle filtering (*PF*) is a sequential Monte Carlo method employing the sequential estimation of relevant probability distributions using the "importance sampling" techniques developed in Chapter 3 and the approximations of distributions with discrete

Bayesian Signal Processing. By James V. Candy

random measures [1–4]. The key idea is to represent the posterior distribution by a set of N_p random samples, the *particles*, with associated weights, $\{x_i(t), \mathcal{W}_i(t)\}$; $i = 1, \ldots, N_p$, and compute the required Monte Carlo estimates. Of course, as the number of particles becomes very large the *MC* representation becomes an equivalent characterization of the analytical description of the *posterior* distribution (e.g., see Ex. 3.15 which converges to the optimal Bayesian estimate).

Thus, particle filtering is a technique to implement sequential Bayesian estimators by *MC* simulation. It offers an alternative to approximate Kalman filtering for nonlinear problems [1, 5]. In *PF*, continuous distributions are approximated by "discrete" random measures composed of these weighted particles or point masses where the *particles* are actually samples of the unknown or hidden states from the state–space representation and the *weights* are the "probability masses" estimated using the Bayesian recursions as shown in Fig. 7.1. From the figure we see that associated with each particle, $x_i(t)$ is a corresponding weight or (probability) mass, $\mathcal{W}_i(t)$ (filled circle). Knowledge of this random measure, $\{x_i(t), \mathcal{W}_i(t)\}$ characterizes an estimate

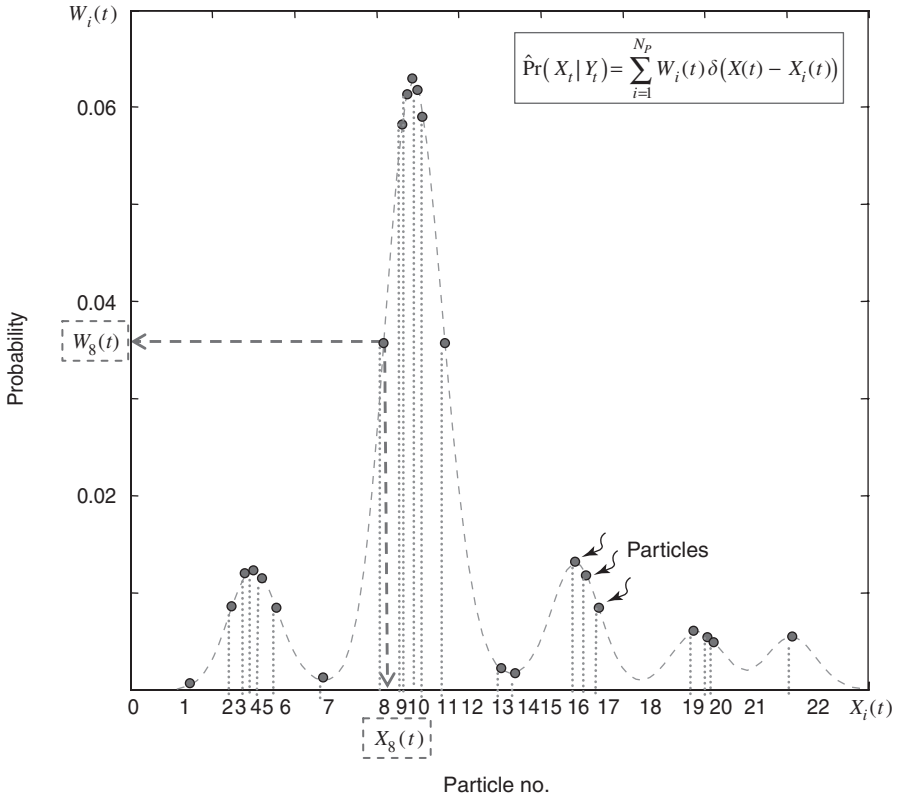


FIGURE 7.1 Particle filter representation of posterior probability distribution in terms of weights (probabilities) and particles (samples).

of the instantaneous (at time t) filtering posterior distribution (solid line),

$$\hat{\Pr}(x(t)|Y_t) \approx \sum_{i=1}^{N_p} \mathcal{W}_i(t)\delta(x(t) - x_i(t))$$

We observe from the figure that the particles need not be equally-spaced or conform to a uniform grid and that they tend to coalesce in high probability regions (*HPR*).

Importance sampling plays a crucial role in state–space particle algorithm development. The *PF* does *not* involve linearizations around current estimates, but rather approximations of the desired distributions by these discrete random measures in contrast to the Kalman filter which sequentially estimates the conditional mean and covariance used to characterize the (Gaussian) filtering posterior, $\Pr(x(t)|Y_t)$. Particle filters are a sequential *MC* methodology based on "point mass" representation of probability distributions that only require a state–space representation of the underlying process. This representation provides a set of particles that evolve at each time-step leading to an instantaneous approximation of the target posterior distribution of the state at time t given all of the data up to that time. Figure 7.2 illustrates the evolution of the posterior at each time step. Here we see the estimated posterior at times t_1, t_2

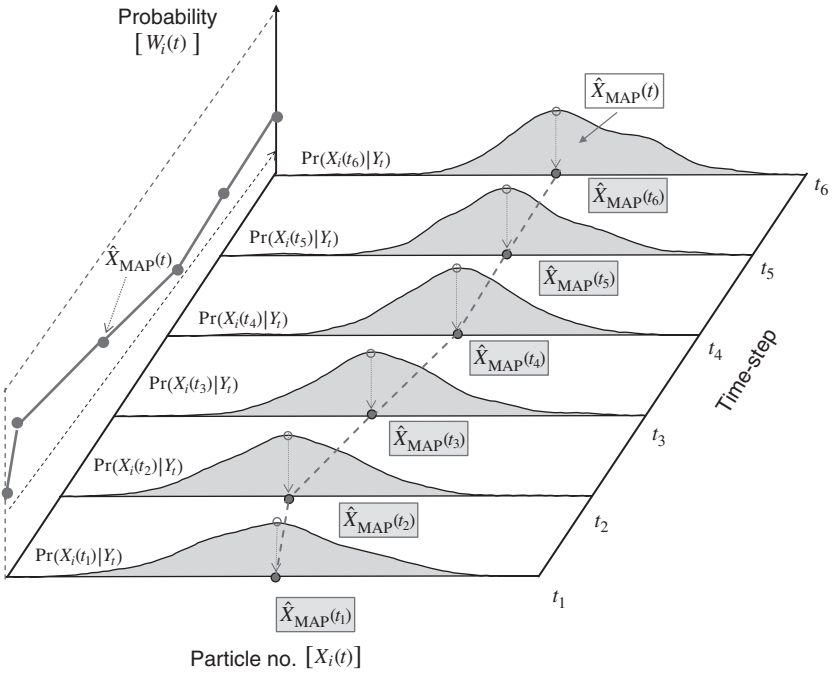


FIGURE 7.2 Particle filter surface (X_i vs. t vs. $\hat{\Pr}(X_i(t) \mid Y_t)$) representation of posterior probability distribution in terms of time (index), particles (samples) and weights or probabilities (left plot illustrates extracted *MAP* estimates vs. t).

to t_6 creating an instantaneous approximation of t vs x_i vs. $\hat{\Pr}(x(t)|Y_t)$. Statistics are calculated across the ensemble at each time-step to provide estimates of the states. For example, the *minimum mean-squared error* (*MMSE*) estimate is easily determined by averaging over x_i, since

$$\begin{aligned}\hat{x}_{\text{MMSE}}(t) &= \int x(t)\Pr(x(t)|Y_t)\,dx \approx \int x(t)\hat{\Pr}(x(t)|Y_t)\,dx \\ &= \int x(t)\left(\sum_{i=1}^{N_p}\mathcal{W}_i(t)\delta(x(t)-x_i(t))\right)dx = \sum_{i=1}^{N_p}\mathcal{W}_i(t)x_i(t)\end{aligned}$$

The maximum *a posteriori* (*MAP*) estimate is simply determined by finding the sample corresponding to the maximum weight of $x_i(t)$ across the ensemble at each time-step (as illustrated in Fig. 7.2), that is,

$$\hat{x}_{\text{MAP}}(t) = \arg\max_{x_i(t)} \hat{\Pr}(x(t)|Y_t) \tag{7.1}$$

In Bayesian processing, the idea is to sequentially estimate the posterior, $\Pr(x(t-1)|Y_{t-1}) \rightarrow \Pr(x(t)|Y_t)$. Recall that optimal algorithms "exactly" track these distributions; however, they are impossible to implement because the updates require integration that cannot usually be performed analytically. Recall that the batch joint posterior distribution follows directly from the chain rule under the Markovian assumptions on $x(t)$ and the conditional independence of $y(t)$, that is,

$$\Pr(X_t|Y_t) = \prod_{i=0}^{t}\Pr(x(i)|x(i-1)) \times \Pr(y(i)|x(i)) \quad \text{for } \Pr(x(0)) := \Pr(x(0)|x(-1)) \tag{7.2}$$

The corresponding sequential importance sampling solution to the Bayesian estimation problem was given generically in Eq. 3.59 starting with the recursive form for the importance distribution as

$$q(X_t|Y_t) = q(X_{t-1}|Y_{t-1}) \times q(x(t)|X_{t-1}, Y_t)$$

leading to the sequential expression for the importance weights as

$$W(t) \propto \frac{\Pr(X_t|Y_t)}{q(X_t|Y_t)} = \frac{\Pr(Y_t|X_t) \times \Pr(X_t)}{q(X_{t-1}|Y_{t-1}) \times q(x(t)|X_{t-1}, Y_t)}$$

$$W(t) = W(t-1) \times \frac{\overbrace{\Pr(y(t)|x(t))}^{\text{Likelihood}} \times \overbrace{\Pr(x(t)|x(t-1))}^{\text{Transition}}}{q(x(t)|X_{t-1}, Y_t)} \tag{7.3}$$

Similarly the *state–space particle filter* (*SSPF*) evolving from this sequential importance sampling construct follows directly. Recall that the generic state–space characterization representing the transition and likelihood probabilities is:

$$\begin{aligned} \Pr(x(t)|x(t-1)) &\Leftrightarrow \mathcal{A}(x(t)|x(t-1)) \\ \Pr(y(t)|x(t)) &\Leftrightarrow \mathcal{C}(y(t)|x(t)) \end{aligned}$$

which leads to an alternate representation of Eq. 7.2

$$\Pr(X_t|Y_t) = \prod_{i=0}^{t} \mathcal{A}(x(i)|x(i-1)) \times \mathcal{C}(y(i)|x(i)) \quad \text{for } \mathcal{A}(x(0)|x(-1)) := \Pr(x(0)) \tag{7.4}$$

Thus the generic state–space sequential importance sampling solution is given by

$$\begin{aligned} x_i(t) &\sim q(x(t)|X_{t-1}, Y_t) \\ W_i(t) &= W_i(t-1) \times \frac{\mathcal{C}(y(t)|x_i(t)) \times \mathcal{A}(x(t)|x_i(t-1))}{q(x_i(t)|X_{t-1}(i), Y_t)} \\ \mathcal{W}_i(t) &= \frac{W_i(t)}{\sum_{i=1}^{N_p} W_i(t)} \end{aligned} \tag{7.5}$$

where the sample at time t, $x_i(t)$ is referred to as the population or system of particles and $X_t(i)$ for the i^{th}-particle as the history (trajectory or path) of that particular particle [6]. It is important to note that a desired feature of sequential *MC* sampling is that the N_p-particles of $X_t(i)$ are *i.i.d.*

In the practical application of the *SSPF* algorithm, we can obtain samples from the posterior by *augmenting* each of the existing samples with the new state draw, that is,

$$X_t(i) = \{x_i(t),\, X_{t-1}(i)\}$$

where $x_i(t) \sim q(x(t)|X_{t-1}, Y_t)$ and $X_{t-1}(i) \sim q(X_{t-1}|Y_{t-1})$.

Now if we further assume that

$$q(x(t)|X_{t-1}, Y_t) \;\rightarrow\; q(x(t)|x(t-1), y(t)) \tag{7.6}$$

then the importance distribution is *only* dependent on $[x(t-1), y(t)]$ which is common when performing filtering, $\Pr(x(t)|Y_t)$, at each instant of time.

Assuming this is true, then the *SSPF* with $x_i(t) \rightarrow X_t(i)$ and $y(t) \rightarrow Y_t$ recursion becomes

$$\begin{aligned} x_i(t) &\sim q(x(t)|x(t-1), y(t)) \\ W_i(t) &= W_i(t-1) \times \frac{\mathcal{C}(y(t)|x_i(t)) \times \mathcal{A}(x(t)|x_i(t-1))}{q(x_i(t)|x_i(t-1), y(t))} \\ \mathcal{W}_i(t) &= \frac{W_i(t)}{\sum_{i=1}^{N_p} W_i(t)} \end{aligned} \tag{7.7}$$

and the filtering *posterior* is estimated by

$$\hat{\Pr}(x(t)|Y_t) \approx \sum_{i=1}^{N_p} \mathcal{W}_i(t) \times \delta(x(t) - x_i(t)) \tag{7.8}$$

We summarize the *generic SSPF* in Table 7.1. Note that as N_p becomes large, in the limit, we have

$$\lim_{N_p \rightarrow \infty} \hat{\Pr}(x(t)|Y_t) \rightarrow \Pr(x(t)|Y_t) \tag{7.9}$$

which implies that the Monte Carlo error *decreases* as the number of particles *increase*.

7.3 IMPORTANCE PROPOSAL DISTRIBUTIONS

Selection of the importance distribution is a critical part of the design phase in particle filtering. Besides assuring that the distribution "covers" the posterior, there are a number of properties that can also be satisfied to achieve a robust design.

7.3.1 Minimum Variance Importance Distribution

Unfortunately, the generic algorithm presented in the previous section has a serious flaw, the *variance* of the importance weights *increases* over time [4, 6]. Therefore, the algorithm degenerates to a *single* non-zero weight after a few iterations. One way to limit this degeneracy is to choose an importance distribution that minimizes the weight variance based on the available information, $[X_{t-1}, Y_t]$. That is, we would like the solution to the problem of minimizing the variance, $V_{q(x(t)|X_{t-1},Y_t)}(W_i(t))$, with respect to $q(\cdot)$ such that

$$V_q(W_i(t)) = W_i^2(t) \left[\int \frac{(\Pr(y(t)|x(t))\Pr(x(t)|X_{t-1}(i)))^2}{q(x(t)|X_{t-1}(i), Y_t)} \, dx(t) - \Pr(y(t)|X_{t-1}(i))^2 \right]$$

is minimized. It has been shown [4] that the *minimum variance importance distribution* that minimizes the variance of the set of weights, $\{W_i(t)\}$ is given by:

$$q_{MV}(x(t)|X_{t-1}, Y_t) \rightarrow \Pr(x(t)|x(t-1), y(t)) \tag{7.10}$$

TABLE 7.1 Generic State–Space Particle Filtering Algorithm

Initialize

$$x_i(0) \rightarrow \Pr(x(0)); \quad W_i(0) = \frac{1}{N_p}; \quad i = 1, \ldots, N_p \qquad \text{[sample]}$$

Importance Sampling

$$x_i(t) \sim \mathcal{A}(x(t)|x_i(t-1)) \qquad \text{[state transition]}$$

State–Space Transition

$$\mathcal{A}(x(t)|x_i(t-1)) \Leftarrow A(x(t-1), u(t-1), w_i(t-1)); \quad w_i \sim \Pr(w_i(t)) \qquad \text{[transition]}$$

Measurement Likelihood

$$\mathcal{C}(y(t)|x_i(t)) \Leftarrow C(x(t), u(t), v(t)); \quad v_i \sim \Pr(v(t)) \qquad \text{[likelihood]}$$

Weight Update

$$W_i(t) = W_i(t-1) \times \frac{\mathcal{C}(y(t)|x_i(t)) \times \mathcal{A}(x(t)|x_i(t-1))}{q(x_i(t)|x_i(t-1), y(t))} \qquad \text{[weights]}$$

Weight Normalization

$$\mathcal{W}_i(t) = \frac{W_i(t)}{\sum_{i=1}^{N_p} W_i(t)} \qquad \text{[weight normalization]}$$

Distribution

$$\hat{\Pr}(x(t)|Y_t) \approx \sum_{i=1}^{N_p} \mathcal{W}_i(t)\delta(x(t) - x_i(t)) \qquad \text{[posterior distribution]}$$

State Estimation (Inference)

$$\hat{x}(t|t) = E\{x(t)|Y_t\} \approx \sum_{i=1}^{N_p} \mathcal{W}_i(t)x_i(t) \qquad \text{[conditional mean]}$$

$$\hat{X}_{MAP}(t) = \arg\max_{x_i(t)} \hat{\Pr}(x(t)|Y_t) \qquad \text{[maximum a-posteriori]}$$

$$\hat{X}_{MED}(t) = \text{median}\left\{\hat{\Pr}(x(t)|Y_t)\right\} \qquad \text{[median]}$$

Let us investigate this minimum variance proposal distribution in more detail to determine its realizability. Using Bayes' rule we can decompose[1] this proposal distribution in steps with $\mathrm{A} = x(t)$ and $\mathrm{B} = x(t-1), y(t)$:

$$\Pr(x(t)|x(t-1), y(t)) = \frac{\Pr(x(t-1), y(t)|x(t)) \times \Pr(x(t))}{\Pr(x(t-1), y(t))} \qquad (7.11)$$

[1] We apply the following form of Bayes' rule: $\Pr(A|B) = \Pr(B|A) \times \Pr(A)/\Pr(B)$.

but the first term in the numerator can be decomposed further to obtain

$$\begin{aligned}\Pr(x(t-1), y(t)|x(t)) &= \Pr(y(t)|x(t-1), x(t)) \times \Pr(x(t-1)|x(t)) \\ &= \Pr(y(t)|x(t-1), x(t)) \times \left[\frac{\Pr(x(t)|x(t-1))\Pr(x(t-1))}{\Pr(x(t))}\right]\end{aligned} \tag{7.12}$$

Substituting this relation into Eq. 7.11 and canceling the $\Pr(x(t))$ terms, we obtain

$$\Pr(x(t)|x(t-1), y(t)) = \frac{\Pr(y(t)|x(t-1), x(t)) \times \Pr(x(t)|x(t-1)) \times \Pr(x(t-1))}{\Pr(x(t-1), y(t))} \tag{7.13}$$

Expand the denominator in Eq. 7.13 using Bayes' rule,

$$\Pr(x(t-1), y(t)) = \Pr(y(t)|x(t-1)) \times \Pr(x(t-1))$$

substitute, cancel the $\Pr(x(t-1))$ terms and apply the conditional independence assumption of the measurements on past states, that is,

$$\Pr(y(t)|x(t-1), x(t)) \;\rightarrow\; \Pr(y(t)|x(t))$$

to obtain the final expression for the *minimum variance proposal distribution* as

$$q_{MV}(x(t)|X_{t-1}, Y_t) = \Pr(x(t)|x(t-1), y(t)) = \frac{\Pr(y(t)|x(t)) \times \Pr(x(t)|x(t-1))}{\Pr(y(t)|x(t-1))} \tag{7.14}$$

If we substitute this expression for the importance distribution in the weight recursion of Eq. 3.63, then we have

$$W(t) = W(t-1) \times \frac{\Pr(y(t)|x(t)) \times \Pr(x(t)|x(t-1))}{q_{MV}(x(t)|X_{t-1}, Y_t)}$$

or

$$W(t) = W(t-1)\Pr(y(t)|x(t))\Pr(x(t)|x(t-1)) \times \left[\frac{\Pr(y(t)|x(t-1))}{\Pr(y(t)|x(t)) \times \Pr(x(t)|x(t-1))}\right]$$

Canceling like terms and applying the Chapman-Kolmogorov relation, we obtain the corresponding *minimum variance weights*

$$\begin{aligned}W_{MV}(t) &= W_{MV}(t-1) \times \Pr(y(t)|x(t-1)) = W_{MV}(t-1) \\ &\quad \times \int \mathcal{C}(y(t)|x(t)) \times \mathcal{A}(x(t)|x(t-1))\, dx(t)\end{aligned} \tag{7.15}$$

which indicates that the importance weights can be calculated *before* the particles are propagated to time t. From this expression we can also see the problem with

the minimum variance importance function approach: (1) we must sample from $\Pr(x(t)|x(t-1), y(t))$; and (2) we must evaluate the integral which generally has *no* analytic form.

7.3.2 Transition Prior Importance Distribution

Another choice for an importance distribution is the *transition prior*. This prior is defined in terms of the state–space representation by $\mathcal{A}(x(t)|x(t-1)) \Leftarrow A(x(t-1), u(t-1), w(t-1))$ which is dependent on the known excitation and process noise statistics and is given by

$$q_{prior}(x(t)|x(t-1), Y_t) \;\rightarrow\; \Pr(x(t)|x(t-1))$$

Substituting this choice into the expression for the weights of Eq. 7.3 gives

$$\begin{aligned} W_i(t) &= W_i(t-1) \times \frac{\Pr(y(t)|x_i(t)) \times \Pr(x(t)|x_i(t-1))}{q_{prior}(x(t)|x_i(t-1), Y_t)} \\ &= W_i(t-1) \times \mathcal{C}(y(t)|x_i(t)) \end{aligned} \tag{7.16}$$

since the priors cancel.

Note two properties for this choice of importance distribution. First, the weight does *not* use the most recent observation, $y(t)$ and second it does not use the past particles $x_i(t-1))$ but only the likelihood. This choice is easily implemented and updated by simply evaluating the measurement likelihood, $\mathcal{C}(y(t)|x_i(t)); i = 1, \ldots, N_p$ for the sampled particle set. In contrast to the minimum variance choice, these weights require the particles to be propagated to time t *before* the weights can be calculated.

This choice of importance distribution can lead to problems, since the transition prior is not conditioned on the measurement data, especially the most recent. Failing to incorporate the latest available information from the most recent measurement to propose new values for the states leads to only a few particles having significant weights when their likelihood is calculated. The transition prior is a much broader distribution than the likelihood indicating that only a few particles will be assigned a large weight. Thus, the algorithm will degenerate rapidly and lead to poor performance especially when data *outliers* occur or measurement noise is *small*. These conditions lead to a "mismatch" between the prior prediction and posterior distributions. Techniques such as the auxiliary particle filter [2, 7, 8] as well as local linearized particle filters [4, 6, 9] have been developed that drive the particles to regions of high likelihood by incorporating the current measurement. Thus, the *SSPF* algorithm takes the same generic form as before with the minimum variance approach; however, we note that the importance weights are much simpler to evaluate with this approach which has been termed the *bootstrap PF*, the *condensation PF*, or the survival of the fittest algorithm. We summarize the *bootstrap particle filter* algorithm in Table 7.2 to follow.

7.4 RESAMPLING

The main objective in simulation-based sampling techniques is to generate *i.i.d.* samples from the targeted posterior distribution in order to perform statistical inferences extracting the desired information. Thus, the importance weights are quite critical since they contain probabilistic information about each specific particle. In fact, they provide us with information about "how probable a sample drawn from the target posterior has been" [10, 11]. Therefore, the weights can be considered acceptance probabilities enabling us to generate independent (approximately) samples from the posterior, $\Pr(x(t)|Y_t)$. Recall that the empirical distribution, $\hat{\Pr}(x(t)|Y_t)$ is defined over a set of finite (N_p) random measures, $\{x_i(t), \mathcal{W}_i(t)\}$; $i = 1, \ldots, N_p$ approximating the posterior, that is,

$$\hat{\Pr}(x(t)|Y_t) \approx \sum_{i=1}^{N_p} \mathcal{W}_i(t)\delta(x(t) - x_i(t)) \tag{7.17}$$

One of the major problems with importance sampling algorithms is the depletion of the particles. The degeneracy of the particle weights creates a problem that must be resolved before these algorithms can be of any pragmatic use. It occurs because the variance of the importance weights increases in time [4] thereby making it impossible to avoid this weight degradation. Degeneracy implies that a large computational effort is devoted to updating particles whose contribution to the posterior is negligible. This approach is bound to fail in the long run, since the weight degeneration leads to a few particles containing most of the probability mass. Thus, there is a need to somehow resolve this problem to make the sequential simulation-based techniques viable. This requirement leads to the idea of "resampling" the particles.

Resampling involves sampling N_p-draws from the current population of particles using the normalized weights as selection probabilities. The resampling process is illustrated in Fig. 7.3. Particles of low probability (small weights) are removed and those of high probability (large weights) are retained and replicated. Resampling results in two major effects: (1) the algorithm is more complex and is not merely the simple importance sampling method; and (2) the resampled trajectories, $X_t(i)$ are no longer *i.i.d.* and the normalized weights are set to $1/N_p$.

Resampling, therefore, can be thought of as a realization of enhanced particles, $\hat{x}_k(t)$, extracted from the original samples, $x_i(t)$ based on their "acceptance probability", $\mathcal{W}_i(t)$, at time t. Statistically, we have

$$\Pr(\hat{x}_k(t) = x_i(t)) = \mathcal{W}_i(t) \quad \text{for } i = 1, \ldots, N_p \tag{7.18}$$

or we write it symbolically as

$$\hat{x}_k(t) \;\Rightarrow\; x_i(t)$$

where "$\Rightarrow$" defines the *resampling operator* generating a set of new particles, $\{\hat{x}_k(t)\}$, replacing the old set, $\{x_i(t)\}$.

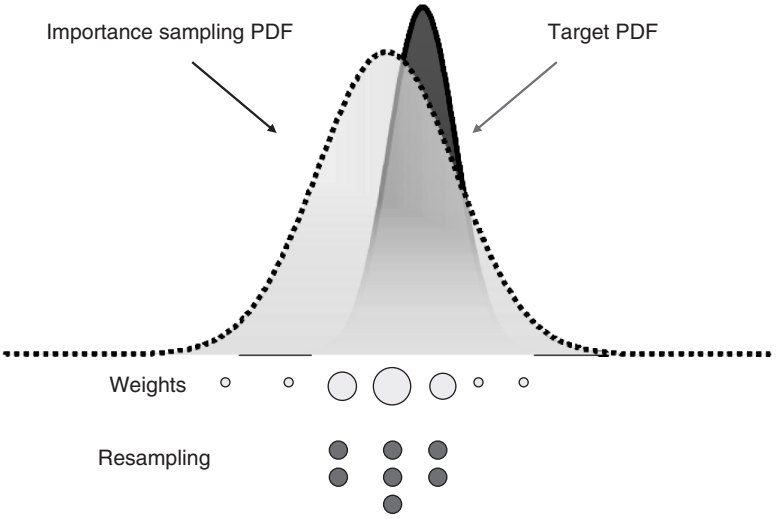


FIGURE 7.3 Resampling consists of processing the predicted particles with their associated weights (probabilities), duplicating those particles of high weight (probability) and discarding those of low weight.

The fundamental concept in resampling theory is to preserve particles with large weights (i.e., large probabilities) while discarding those with small weights. Two steps must occur to resample effectively: (1) a decision, on a weight-by-weight basis, must be made to *select* the appropriate weights and *reject* the inappropriate; and (2) resampling must be performed to minimize the degeneracy. This overall strategy when coupled with importance sampling is termed sequential *sampling-importance-resampling* (*SIR*) [4].

A reasonable measure of degeneracy is the *effective* particle sample size based on the coefficient of variation [12] defined by

$$N_{eff}(t) := \frac{N_p}{E_q\{W^2(X_t)\}} = \frac{N_p}{1 + V_q(W(X_t))} \leq N_p \tag{7.19}$$

An estimate of the effective number of particles at time t is given by

$$\hat{N}_{eff}(t) = \frac{1}{\sum_{i=1}^{N_p} W_i^2(t)} \tag{7.20}$$

and a decision based on the *rejection method* [13] is made by comparing it to a threshold, N_{thresh}. That is, when $\hat{N}_{eff}(t)$ is less than the threshold, resampling is performed.

$$\hat{N}_{eff}(t) = \begin{cases} \leq N_{thresh} & \text{Resample} \\ > N_{thresh} & \text{Accept} \end{cases}$$

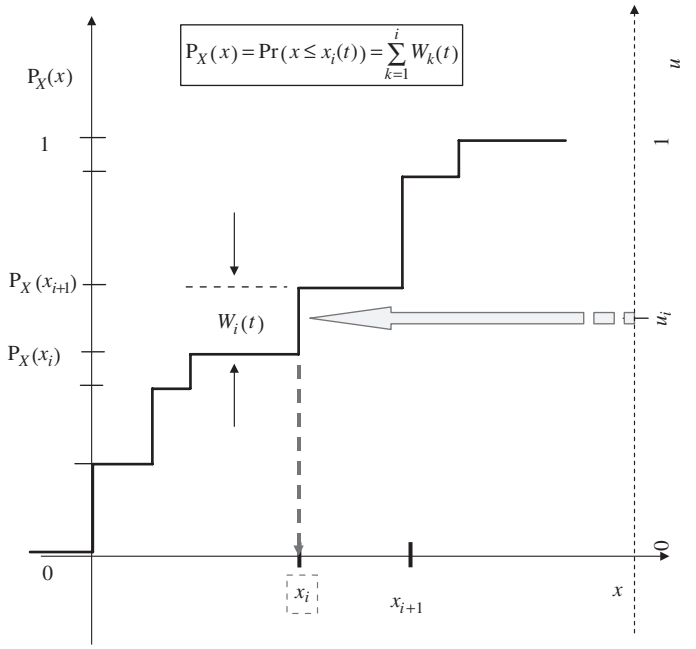


FIGURE 7.4 Resampling (with replacement) by inverse transforming a uniform sampler to generate samples from target distribution.

Once the decision is made to resample, a uniform sampling procedure [14] can be applied removing samples with low importance weights and replicating samples with high importance weights. Resampling (with replacement) generates a set of new samples with uniform weights from an approximate discrete posterior distribution,

$$\hat{\Pr}(x(t)|Y_t) \approx \sum_{i=1}^{N_p} \hat{\mathcal{W}}_i(t)\delta(x(t) - \hat{x}_i(t)) \tag{7.21}$$

so that $\Pr[\hat{x}_i(t) = x_j(t)] = \hat{W}_j(t)$. The resulting independent and identically distributed sample from the probability mass of Eq. 7.21 is uniform such that the sampling induces the mapping of $\{\hat{x}_i(t), \hat{\mathcal{W}}_i(t)\} \rightarrow \{x_i(t), \mathcal{W}_i(t)\}, \hat{\mathcal{W}}_i(t) = 1/N_p \ \forall i$. The selection of $\hat{x}_i(t) = x_j(t)$ is shown in Fig. 7.4. Here the procedure is termed systematic resampling [4]. For each resampled particle, that is, N_i-times is related to the original particle. The methodology relies on uniformly sampling the cumulative distribution function resulting in the new replicated particles or uniform weighting. The resampling algorithm is incorporated in Table 7.2.

Resampling decreases the degeneracy problem algorithmically, but introduces its own set of problems. After one resampling step, the simulated trajectories are *no*

TABLE 7.2 Bootstrap *SIR* State–Space Particle Filtering Algorithm

Initialize

$$x_i(0) \sim \Pr(x(0)) \quad W_i(0) = \frac{1}{N_p} \quad i = 1, \ldots, N_p \qquad \text{[sample]}$$

Importance Sampling

$$x_i(t) \sim \mathcal{A}(x(t)|x_i(t-1)) \Leftarrow A(x(t-1), u(t-1), w_i(t-1));$$
$$w_i \sim \Pr(w_i(t)) \qquad \text{[state transition]}$$

Weight Update

$$W_i(t) = \mathcal{C}(y(t)|x_i(t)) \Leftarrow C(x(t), u(t), v(t)); \quad v \sim \Pr(v(t)) \qquad \text{[weight/likelihood]}$$

Weight Normalization

$$\mathcal{W}_i(t) = \frac{W_i(t)}{\sum_{i=1}^{N_p} W_i(t)}$$

Resampling Decision

$$\hat{N}_{eff} = \frac{1}{\sum_{i=1}^{N_p} W_i^2(t)} \qquad \text{[effective samples]}$$

$$\hat{N}_{eff} = \begin{cases} \text{Resample} & \leq N_{thresh} \\ \text{Accept} & > N_{thresh} \end{cases}$$

Resampling

$$\hat{x}_i(t) \Rightarrow x_i(t)$$

Distribution

$$\hat{\Pr}(x(t)|Y_t) \approx \sum_{i=1}^{N_p} \mathcal{W}_i(t)\delta(x(t) - \hat{x}_i(t)) \qquad \text{[posterior distribution]}$$

longer statistically independent. Therefore, the simple theoretical convergence results under these assumptions lose their validity. Pragmatically, resampling can limit algorithm parallelization because combining particles causes an increase in computational complexity. Also there is a possible loss in diversity caused by replication of those particles with highest importance weights [4]. Thus as with any methodology there are tradeoffs that must be considered.

7.4.1 Multinomial Resampling

There are a variety of techniques available to implement the basic resampling method [1, 6, 16, 19]. The usual approach is to resample with replacement, since the probability of each particle $x_i(t)$ is given by the normalized weight $W_i(t)$. Therefore, the number

of times N_i that each particular particle in the original set $\{x_i(t)\}$ is selected follows a binomial distribution, $\text{Bin}(N_p, W_i(t))$. The corresponding vector, $[N_1, \ldots, N_{N_p}]$ is distributed according to a multinomial distribution with parameter, N_p and probability of success $[W_1(t), \ldots, W_{N_p}(t)]$. With this resampling scheme, particles in the original set with small variance weights are most likely discarded, while those of high weights are replicated in proportion to these weights. The *multinomial resampling* method is given by:

- Given a random measure at time t, that is, a set of particles and weights, $\{x_i(t), W_i(t)\}, i = 1, \ldots, N_p$;
- Sample uniformly, $u_k \rightarrow \mathcal{U}(0, 1); k = 1, \ldots, N_p$;
- Determine the index, i_k: $i_k = k$ for $\Pr(x_{i_k}(t) = x_k(t)) = u_k$;
- Select a new sample, $\hat{x}_{i_k}(t) \Rightarrow x_i(t)$ and weight, $\hat{W}_{i_k}(t) = \frac{1}{N_p}$ based on the new sample index, i_k; and
- Generate the new random (resampled) measure: $\{\hat{x}_{i_k}, \hat{W}_{i_k}(t)\}$; for $k = 1, \ldots, N_p$.

Here the index notation, i_k designates the original i^{th}-particle or parent and the new k^{th}-particle using the inverse *CDF* method of Sec. 3.3. This sampling scheme is equivalent to drawing, $i_k; k = 1, \ldots, N_p$ samples from a multinomial distribution with parameters, $\mathcal{M}(N_{i_k}, W_{i_k}(t))$ and corresponding statistics: mean, $E\{N_{i_k}\} = N_p$ and variance, $\text{Var}(N_{ik}) = N_p W_{ik}(t)(1 - W_{ik}(t))$.

The basic idea is to first construct the *CDF* from the original random measure, $\{x_i(t), W_i(t)\}$, since it is given by

$$\Pr(X(t) \leq x_i(t)) \approx \sum_{i=1}^{N_p} W_i(t)\mu(x(t) - x_i(t)) \tag{7.22}$$

where $\mu(\cdot)$ is the unit step function.

Uniform samples, u_k, are drawn from the interval $[0, 1]$ and projected onto the inverse *CDF* (see Sec. 3.3) corresponding to the associated probability and identifying the particular new particle sample index, i_k, and corresponding replacement particle, $\hat{x}_{i_k}(t)$ leading to the resampling

$$\hat{x}_{i_k}(t) \;\Rightarrow\; x_i(t) \tag{7.23}$$

Clearly those particles or samples with highest probability (or weights) will be selected more frequently, thereby, replacing particles with lower probability (weights) and therefore, the new random measure is created, that is,

$$\{\hat{x}_{i_k}(t), \hat{W}_{i_k}(t)\} \;\Rightarrow\; \{x_i(t), W_i(t)\} \quad \text{with } \hat{W}_{i_k}(t) = \frac{1}{N_p} \tag{7.24}$$

This resampling technique represents a direct implementation of random sampling by generating an *i.i.d.* sample from the empirical posterior distribution

$$\begin{aligned}\Pr(x(t)|Y_t) &\approx \sum_{i=1}^{N_p} W_i(t)\delta(x(t)-x_i(t)) \\ &\to \sum_{i=1}^{N_p} \hat{W}_{i_k}(t)\delta(x(t)-\hat{x}_{i_k}(t)) = \frac{1}{N_p}\sum_{i=1}^{N_p}\delta(x(t)-\hat{x}_{i_k}(t))\end{aligned} \tag{7.25}$$

A second more efficient way of generating this measure is the "systematic" resampling method.

7.4.2 Systematic Resampling

The *systematic resampling* method is based on an *ordered* technique in which a set of N_p-ordered uniform variates are generated [20]. It minimizes the error variance between the original selected sample and its mean. Thus, the *systematic sampling* method is given by:

- Given a random measure at time t, that is, a set of particles and weights, $\{x_i(t), W_i(t)\}, i = 1, \ldots, N_p$;
- Sample uniform N_p-ordered variates: $\hat{u}_k = u_k + \frac{k-1}{N_p}$ for $k = 1, \ldots, N_p$ and $u_k \to \mathcal{U}(0,1)$;
- Determine the index, i_k: $i_k = k$ for $\mathrm{P}_X(x_{k-1}(t)) \leq \hat{u}_k \leq \mathrm{P}_X(x_k(t))$; (see Fig. 7.4)
- Select a new sample, $\hat{x}_{i_k}(t) \Rightarrow x_i(t)$ and weight, $\hat{W}_{i_k}(t) = \frac{1}{N_p}$ based on the new sample index, i_k; and
- Generate the new random (resampled) measure: $\{\hat{x}_{i_k}, \hat{W}_{i_k}(t)\}$; for $k = 1, \ldots, N_p$.

where recall the *CDF* is given by: $\mathrm{P}_X(x_k(t)) = \sum_{k=1}^{N_p} W_k(t)\mu(x(t)-x_k(t))$ with $\mu(\cdot)$ is a unit step function.

The final sampling scheme we discuss has a low weight variance, the residual method.

7.4.3 Residual Resampling

The *residual resampling* method is based on the idea of estimating the number of times each particle should be replicated, that is, the i^{th}-particle is replicated, say

$$\overline{N}_p(i) := \mathrm{Int}(E\{N_p(i)\}) = \mathrm{Int}(N_p \times W_i(t)) \tag{7.26}$$

times where Int means the "smallest integer value of".

The remaining particles are sampled using the multinomial sampling method discussed above. Here we have

$$\overline{N}_p(t) := N_p - \sum_{i=1}^{N_p} \overline{N}_p(i) \tag{7.27}$$

with corresponding weights

$$\hat{W}_i(t) = \frac{1}{\overline{N}_p(t)}(N_p W_i(t) - \overline{N}_p(i)) \tag{7.28}$$

The overall effect is to reduce the variance by $E\{(N_p(i) - E\{\overline{N}_p(i)\})^2\}$, since the particles cannot be replicated less than $E\{N_p(i)\}$ times.

We summarize the *residual resampling* method by:

- Given a random measure at time t, that is, a set of particles and weights, $\{x_i(t), W_i(t)\}, i = 1, \ldots, N_p$;
- Calculate $\overline{N}_p(i)$: $\overline{N}_p(i) = \text{Int}(N_p \times W_i(t))$;
- Multinomial Sample: $\hat{x}_i(t) \Rightarrow x_i(t)$ for $i = 1, \ldots, \overline{N}_p(i)$; and
- Update the new random (resampled) measure: $\{\hat{x}_{i_k}, \hat{W}_{i_k}(t)\}$; for $k = 1, \ldots, N_p$.

So we see that there are a variety of resampling schemes that can be employed to solve the particle degeneracy problem. We can now update our generic particle filtering algorithm to incorporate a resampling procedure and alleviate the degeneracy problem created by the variation of the weights.

To visualize the "resampling" approach mitigating the particle degeneracy problem, the *SIR* is illustrated in Fig. 7.5. Here we show the evolution of the particles and weights starting with the uniform weighting $\left(\hat{W}_i(t-1) = \frac{1}{N_p}\right)$. Once the initial weights are updated, they are *resampled uniformly*. Next they are propagated using the state–space transition mechanism (model), then updated using the measurement likelihood producing the measure, $\{x_i(t), W_i(t)\}, i = 1, \ldots, N_p$ leading to an approximation of the posterior distribution at time t. This measure is then resampled, propagated, updated and so on. A generic flow diagram of the algorithm is shown in Fig. 7.6 where we again illustrate the basic ingredients of the *SIR* technique.

7.5 STATE–SPACE PARTICLE FILTERING TECHNIQUES

There are a number of pragmatic *PF* techniques that have been introduced in the literature. Here we discuss some of the more robust and popular techniques that have been applied to a wide variety of problems starting with the bootstrap processor [1, 2, 6].

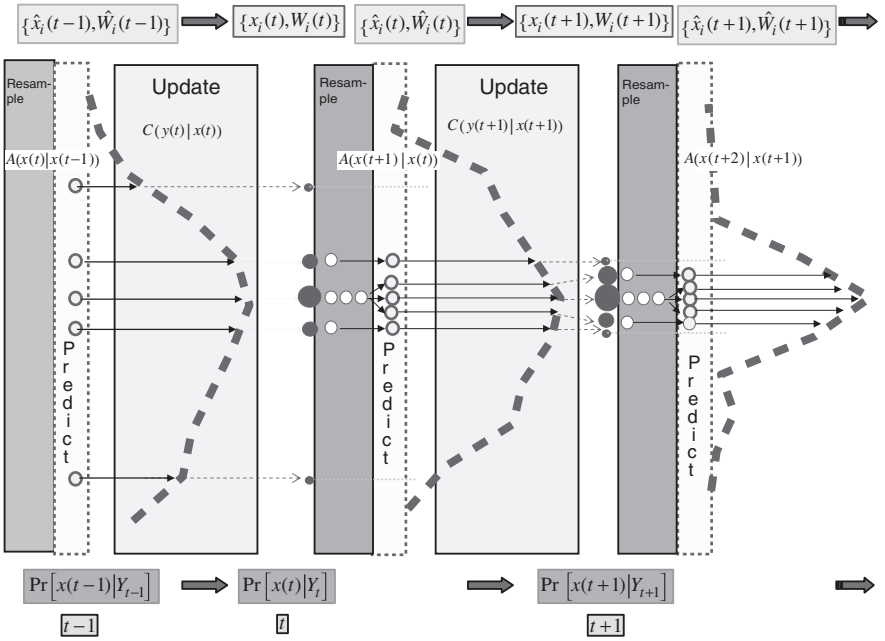


FIGURE 7.5 Evolution of particle filter weights and particles using the sequential state–space *SIR* algorithm: resampling, propagation-step (state–space transition model), update-step (state–space measurement likelihood), resampling ....

7.5.1 Bootstrap Particle Filter

The basic "bootstrap" algorithm developed by Gordon, Salmond and Smith [16] is one of the first practical implementations of the processor to the tracking problem. It is the most heavily applied of all *PF* techniques due to its simplicity. Thus, the *SSPF* algorithm takes the same generic form as before with the minimum variance approach; however, we note that the importance weights are much simpler to evaluate with this approach which has been termed the *bootstrap PF*, the *condensation PF*, or the survival of the fittest algorithm [2, 16, 22].

As mentioned previously, it is based on sequential sampling-importance-resampling (*SIR*) ideas and uses the *transition prior* as its underlying proposal distribution,

$$q_{prior}(x(t)|x(t-1), Y_t) = \Pr(x(t)|x(t-1))$$

The corresponding weight becomes quite simple and only depends on the likelihood; therefore, it is not even necessary to perform a sequential updating because

$$W(t) = \left(\frac{\Pr(y(t)|x(t)) \times \Pr(x(t)|x(t-1))}{\Pr(x(t)|x(t-1))}\right) \times W(t-1) = \Pr(y(t)|x(t)) \times W(t-1)$$

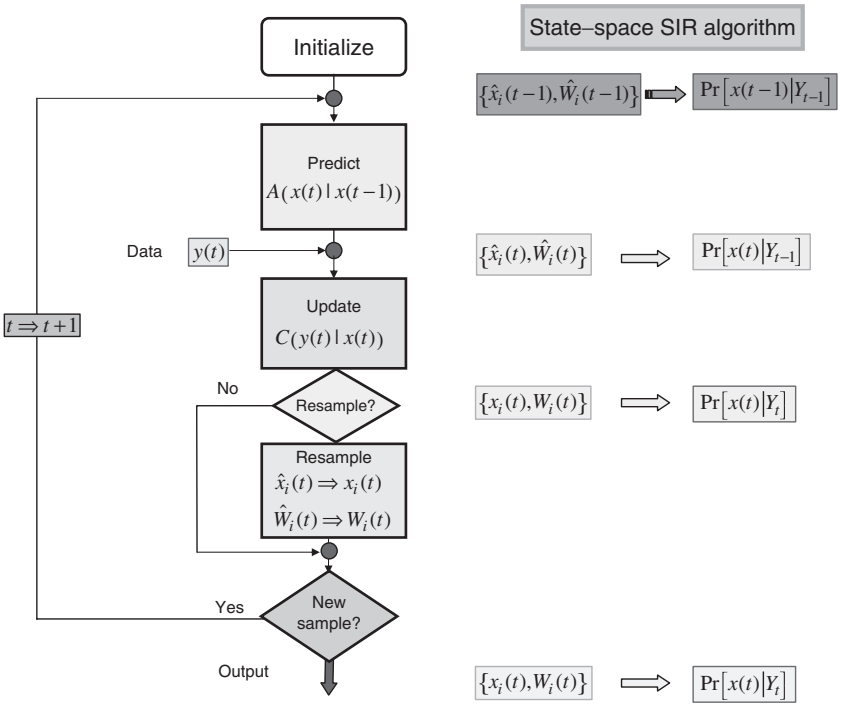


FIGURE 7.6 State–space *SIR* particle filtering algorithm structure: initialization, propagation (state transition), updating (measurement likelihood), resampling.

since the filter requires resampling to mitigate variance (weight) increases at each time step [16]. *After* resampling, the new weights become

$$\mathcal{W}(t) \;\rightarrow\; \frac{1}{N_p} \;\Rightarrow\; W(t) = \Pr(y(t)|x(t)) = \mathcal{C}(y(t)|x(t))$$

revealing that there is *no need* to save the likelihood (weight) from the previous step! With this in mind we summarize the simple *bootstrap particle filter* algorithm in Table 7.2. One of the primary features as well as shortcomings of this technique is that it does not use the latest available measurement in the importance proposal, but only the previous particles, $x_i(t-1)$, which differs from the minimum variance approach. Also in order to achieve *convergence*, it is necessary to resample at every time step. In practice, however, many applications make the decision to resample based on the effective sample-size metric discussed in the previous section.

In order to construct the bootstrap *PF*, we assume that: (1) $x_i(0) \sim \Pr(x(0))$ is known; (2) $\mathcal{A}(x(t)|x(t-1))$, $\mathcal{C}(y(t)|x(t))$ are known; (3) samples can be generated using the process noise input and the state-space model, $A(x(t-1), u(t-1), w(t-1))$; (4) the likelihood is available for point-wise evaluation, $\mathcal{C}(y(t)|x(t))$ based on the measurement model, $C(x(t), v(t))$; and (5) resampling is performed at every time-step. To implement the algorithm we

- Generate the initial state, $x_i(0)$
- Generate the process noise, $w_i(t)$

- Generate the particles, $x_i(t) = A(x_i(t-1), u(t-1), w_i(t-1))$—the *prediction-step*
- Generate the likelihood, $\mathcal{C}(y(t)|x_i(t))$ using the current particle and measurement—the *update step*
- Resample the set of particles retaining and replicating those of highest weight (probability), $\hat{x}_i(t) \Rightarrow x_i(t)$
- Generate the new set, $\{\hat{x}_i(t), \hat{\mathcal{W}}_i(t)\}$ with $\hat{\mathcal{W}}_i(t) = \frac{1}{N_p}$

Next we revisit the model of Jazwinski [21] in Chapter 5 and apply the simple bootstrap algorithm to demonstrate the *PF* solution using the state–space *SIR* particle filtering algorithm.

Example 7.1

Recall the discrete state–space representation of the basic problem given by the Markovian model:

$$\begin{aligned} x(t) &= (1 - 0.05\triangle T)x(t-1) + 0.04\triangle T x^2(t-1) + w(t-1) \\ y(t) &= x^2(t) + x^3(t) + v(t) \end{aligned}$$

where $\triangle t = 0.01$, $w \sim \mathcal{N}(0, 10^{-6})$ and $v \sim \mathcal{N}(0, 0.09)$. The initial state is Gaussian distributed with $\bar{x}_i(0) \sim \mathcal{N}(\bar{x}(0), \overline{P}(0))$ and $\bar{x}(0) = 2.0$, $\overline{P}(0) = 10^{-2}$.

We selected the following *simulation* run parameters:

Number of Particles:	250
Number of Samples:	150
Number of States:	1
Sampling Interval:	0.01 sec
Number of Measurements:	1
Process Noise Covariance:	1×10^{-6}
Measurement Noise Covariance:	9×10^{-2}
Initial State:	2
Initial State Covariance:	10^{-20}

Thus, the bootstrap *SIR* algorithm of Table 7.2 for this problem becomes:

1. Draw samples (particles) from the state transition distribution: $x_i(t) \rightarrow \mathcal{N}(x(t) : \mathbf{a}[x(t-1)], \mathbf{R}_{ww})$, that is, generate

$$w_i(t) \;\rightarrow\; \Pr(w(t)) \sim \mathcal{N}(0, R_{ww})$$

and calculate $\{x_i(t)\}$ using the process model and $w_i(t)$

$$x_i(t) = (1 - 0.05\triangle T)x_i(t-1) + 0.04\triangle T x_i^2(t-1) + w_i(t-1)$$

2. Estimate the weight/likelihood, $W_i(t) = \mathcal{C}(\mathbf{y}(t)|x_i(t)) \rightarrow \mathcal{N}(\mathbf{y}(t) : \mathbf{c}[x_i(t)], \mathbf{R}_{vv}(t))$

$$c[x_i(t)] = x_i^2(t) + x_i^3(t)$$

$$\ln \mathcal{C}(\mathbf{y}(t)|x_i(t)) = -\frac{1}{2} \ln 2\pi R_{vv} - \frac{(y(t) - x_i^2(t) - x_i^3(t))^2}{2R_{vv}}$$

3. Update the weight: $W_i(t) = \mathcal{C}(\mathbf{y}(t)|x_i(t))$
4. Normalize the weight: $\mathcal{W}_i(t) = W_i(t) / \sum_{i=1}^{N_p} W_i(t)$
5. Decide to resample if $N_{eff} \leq N_{thresh}$
6. If resample: $\hat{x}_i(t) \Rightarrow x_i(t)$
7. Estimate the instantaneous posterior:

$$\hat{\Pr}(x(t)|Y_t) \approx \sum_{i=1}^{N_p} \mathcal{W}_i(t) \delta(x(t) - \hat{x}_i(t))$$

8. Estimate (inference) the corresponding statistics:

$$\begin{aligned} \hat{X}_{\text{MAP}}(t) &= \arg\max_{x_i(t)} \hat{\Pr}(x(t)|Y_t) \\ \hat{X}_{\text{MMSE}}(t) &= E\{x(t)|Y_t\} = \sum_{i=1}^{N_p} \hat{x}_i(t) \hat{\Pr}(x(t)|Y_t) \\ \hat{X}_{\text{MEDIAN}}(t) &= \text{median}(\hat{\Pr}(x(t)|Y_t)) \end{aligned}$$

Note that compared to the previous examples of Chapter 5 for the extended Bayesian processors, we have added more process noise to demonstrate the effectiveness of the bootstrap processor. The results of the bootstrap *PF* are shown in Fig. 7.7. The usual classical performance metrics are shown: zero-mean ($0.03 < 0.17$), whiteness (0.78% out) and *WSSR* (below threshold) all indicating (approximately) a tuned processor. The *PF* tracks the state and measurement after the initial transient error has diminished.

In Fig. 7.8 we show the bootstrap state and measurement estimates (inferences), that is, the *MAP* and *MMSE* compared to the modern sigma-point processor *SPBP* (*UKF*). The plots demonstrate that the *PF* can outperform the sigma-point design, that assumes a unimodal Gaussian distribution. The estimated state and predicted measurement posterior distributions are shown in Fig. 7.9 along with a time-slice in Fig. 7.10 at time 1.04 sec demonstrating the capability of the bootstrap *PF* to characterize the multimodal nature of this problem. △△△

This completes the development of the most popular and simple *PF* technique. We mention in passing that a simple pragmatic method of preventing the sample impoverishment problem is to employ a method suggested by Gordon [16] and refined in [17, 18] termed particle "roughening" which is similar to adding process noise to constant parameters when constructing a random walk *GM* model.

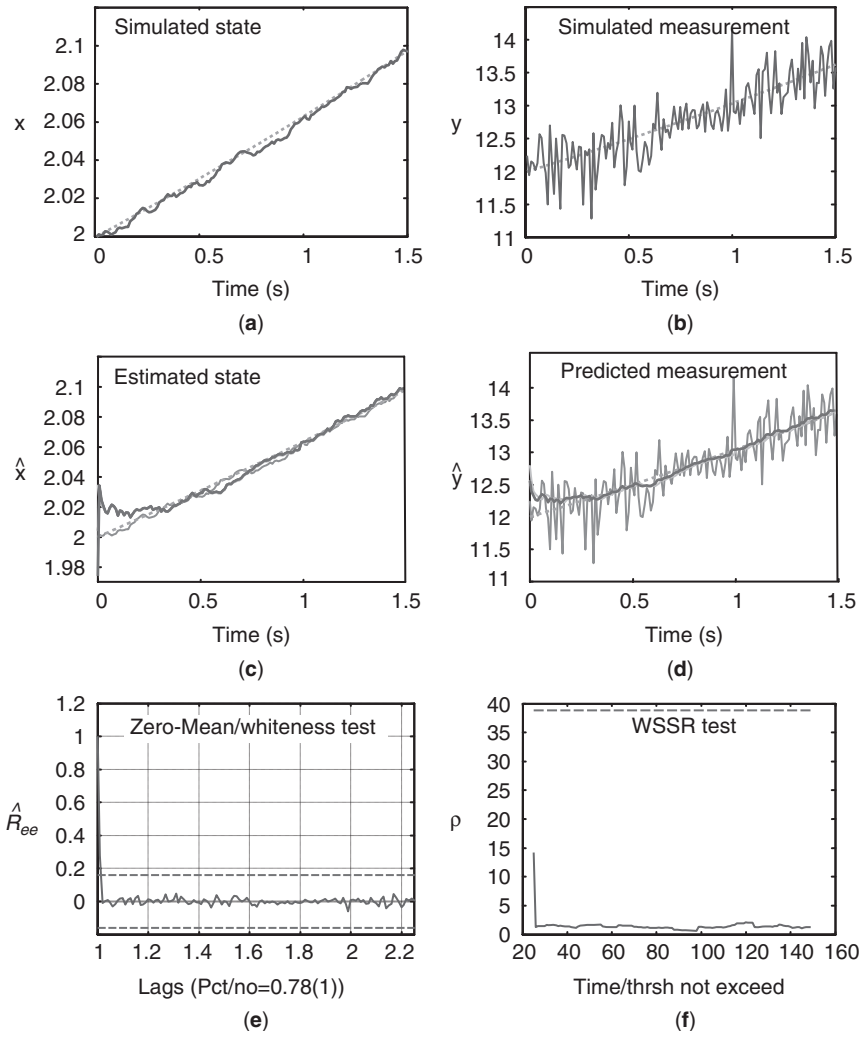


FIGURE 7.7 Nonlinear trajectory simulation/estimation for low process noise case: (a) Simulated state and mean. (b) Simulated measurement and mean. (c) Bootstrap state estimate. (d) Bootstrap measurement estimate. (e) Zero-Mean/Whiteness test ($0.03 < 0.17/0.78\%$ out). (f) WSSR test (below threshold).

Roughening[2] consists of adding random noise to each particle *after* resampling is accomplished, that is, the *a posteriori* particles are modified as

$$\tilde{x}_i(t) = \hat{x}_i(t) + \epsilon_i(t) \tag{7.29}$$

[2] Roughening is useful in estimating embedded state–space model parameters and is applied to the joint state/parameter estimation problem in Sec. 8.4 to follow.

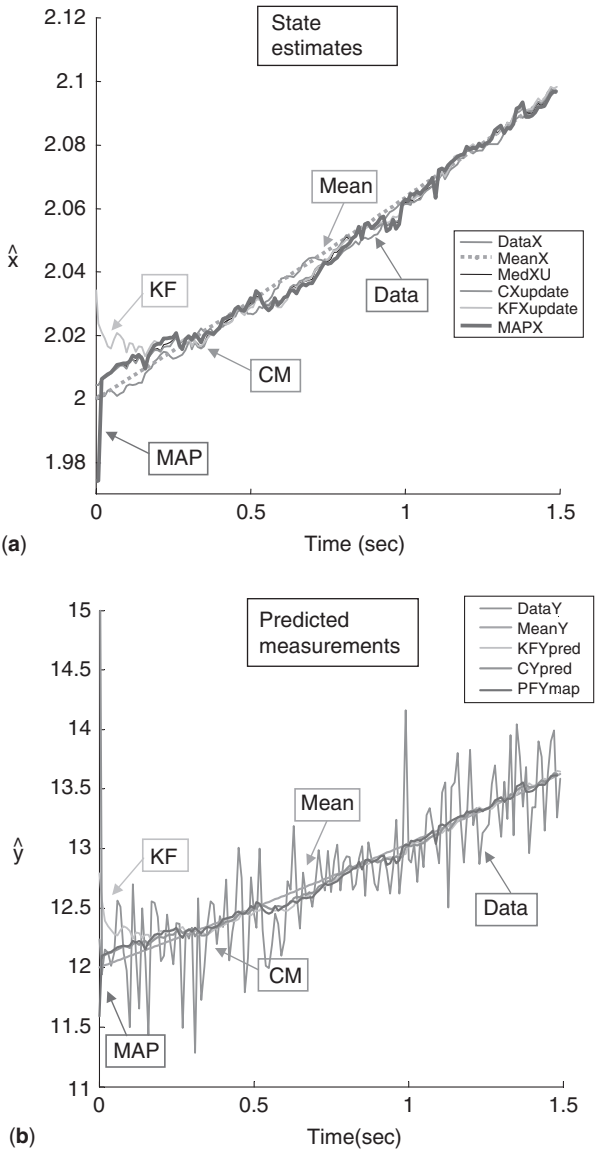


FIGURE 7.8 Nonlinear trajectory Bayesian estimation comparison for low process noise case problem: (a) State estimates: *MAP, MMSE, UKF,* median. (b) Predicted measurement estimates: *MAP, MMSE, UKF,* median.

where $\epsilon_i \sim \mathcal{N}(0, \text{diag}[\kappa\, \mathcal{M}_n\, N_p^{-1/N_x}])$ and κ is a constant "tuning" parameter (e.g. $\sim$0.2), $\mathcal{M}_n$ is a vector of the maximum difference between particle components *before* roughening, the n^{th}-element of $\mathcal{M}$ given by (see Sec. 8.4 for application):

$$\mathcal{M}_n = \max_{ij} |x_i^{(n)}(t) - x_j^{(n)}(t)| \quad \text{for } n = 1, \ldots, N_x \tag{7.30}$$

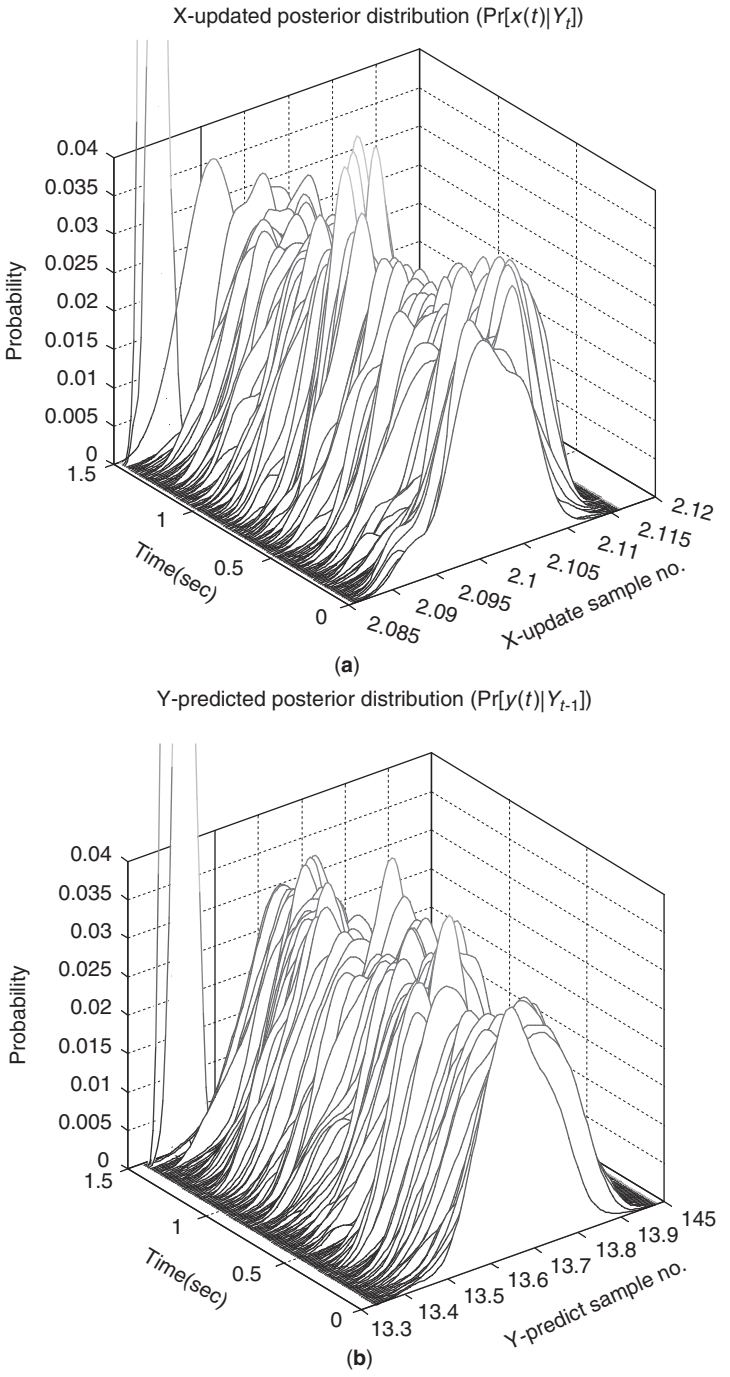


FIGURE 7.9 Nonlinear trajectory Bayesian estimation instantaneous posterior distributions: (a) Updated state distribution. (b) Predicted measurement distribution.

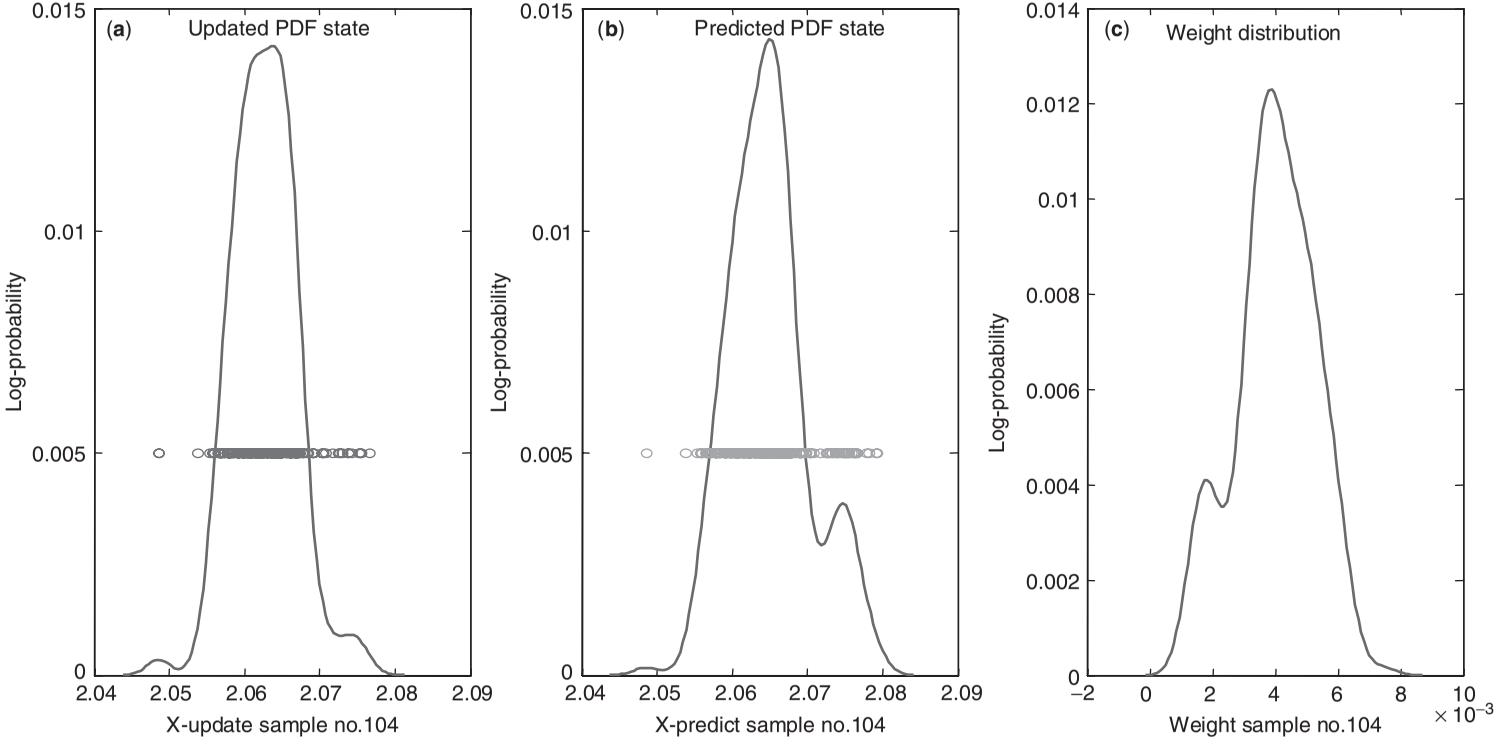


FIGURE 7.10 Nonlinear trajectory Bayesian estimation instantaneous posterior distribution at time-slice 1.04 sec: (a) Updated state distribution. (b) Predicted state distribution. (c) Weight (likelihood for bootstrap) distribution.

Next we consider some alternative approaches that attempt to approximate the minimum variance importance distribution more closely for improved performance.

7.5.2 Auxiliary Particle Filter

The auxiliary particle filter employing sampling-importance-resampling (*ASIR*) is a variant of the standard *SIR* [8]. It is based on attempting to mitigate two basic weaknesses in particle filtering: (1) poor outlier performance; and (2) poor posterior tail performance. These problems evolve from the empirical approximation of the filtering posterior which can be considered a *mixture* distribution.

The basic concept of *ASIR* is to mimic the operation of the minimum variance (optimal) importance distribution, $q_{MV}(x(t)|x(t-1), y(t))$, by introducing an *auxiliary* variable, $\mathcal{K}$, representing the weight of the mixture used in the empirical prediction distribution estimate. The idea is to perform resampling at time $(t-1)$ using the available measurement at time t *before* the particles $\{x_i(t)\}$ are propagated to time t through the transition and likelihood distributions. The key step is to favor particles at time $(t-1)$ that are likely to "survive" (largest weights) at the *next* time-step, t. The problem is that these schemes tend to introduce a *bias* into the estimated posterior that must then be corrected by modifying the weight of the remaining particles. Thus, the *ASIR* is a two-stage process such that: (1) particles with large predictive likelihoods at time-step $(t-1)$ are propagated; and (2) the resulting particles are then re-weighted and drawn from the resulting posterior.

Following the development in Cappe [6], we start with a proposal over the entire path $\{X_t\}$ up to time t under the assumption that the joint posterior at time $(t-1)$ is well approximated by a particle representation, $\{\mathcal{W}_i(t-1), X_{t-1}(i)\}$. Thus, the joint importance proposal for the "new" particles $\{X_t(i)\}$ is

$$q(X_t) = \overbrace{q(X_{t-1}|Y_t)}^{PAST} \times \overbrace{q(x(t)|x(t-1), y(t))}^{NEW} \tag{7.31}$$

Note that the "past" trajectories depend on the data *up to* time-step t to enable the adaption to the new data, $y(t)$, while the "new" conditional importance distribution $(q \rightarrow q_{MV})$ incorporates the new state, $x(t)$. We substitute an empirical distribution for $\Pr(X_{t-1}|Y_t)$ centered on the previous particle paths $\{X_{t-1}(i)\}$

$$q(X_{t-1}|Y_t) \approx \sum_{i=1}^{N_p} \mathcal{K}_i(t-1)\delta(X_{t-1} - X_{t-1}(i)) \tag{7.32}$$

where $\sum_{i=1}^{N_p} \mathcal{K}_i(t-1) = 1$ and $\mathcal{K}_i(t-1) > 0$. The i^{th} weight (probability mass) for each particle in this proposal is based on the pre-selected particles that are a "good fit" to the new data point $y(t)$. One choice [8] for these weights is to choose a *point estimate* of the state such as its mean,

$$\hat{m}_i(t) = \int x(t) \times \Pr(x(t)|x_i(t-1))\, dx(t) \tag{7.33}$$

and then compute the weighting function as the likelihood evaluated at this point

$$\mathcal{K}_i(t-1) = \mathcal{C}(y(t)|\hat{m}_i(t))$$

as in the bootstrap technique [16] or if the particles $\{x_i(t-1)\}$ are weighted [6], then

$$\mathcal{K}_i(t-1) = \mathcal{W}_i(t-1) \times \mathcal{C}(y(t)|\hat{m}_i(t))$$

which follows from the marginal $\Pr(X_{t-1}|Y_t)$—a smoothing distribution. That is, using the particle approximation from time $(t-1)$ and expanding, we obtain

$$\Pr(X_{t-1}|Y_t) \propto \int \Pr(X_{t-1}|Y_{t-1}) \times \mathcal{A}(x(t)|x(t-1)) \times \mathcal{C}(y(t)|x(t))\, dx(t) \tag{7.34}$$

and using the empirical distribution approximation for the first term gives

$$\Pr(X_{t-1}|Y_t) \approx \sum_{i=1}^{N_p} \mathcal{W}_i(t-1)\delta(X_{t-1} - X_{t-1}(i)) \times \int \mathcal{C}(y(t)|x(t))\mathcal{A}(x(t)|x_i(t-1))\, dx(t) \tag{7.35}$$

One approximation [8] $\mathcal{A}(x(t)|x_i(t-1)) \to \delta(x(t) - \hat{m}_i(t))$ leads to the estimator

$$\hat{\Pr}(X_{t-1}|Y_t) \approx \sum_{i=1}^{N_p} \underbrace{\mathcal{C}(y(t)|\hat{m}_i(t)) \times \mathcal{W}_i(t-1)}_{\text{combined weight}} \times \delta(X_{t-1} - X_{t-1}(i)) \tag{7.36}$$

giving the desired result above

$$\mathcal{K}_i(t-1) := \mathcal{W}_i(t-1) \times \mathcal{C}(y(t)|\hat{m}_i(t)) \tag{7.37}$$

Using this proposal, the generalized weight

$$\mathcal{W}_{\text{aux}}(i,t) := \frac{\Pr(X_t|Y_t)}{q(X_t)}$$

is determined from the ratio of the posterior

$$\Pr(X_t|Y_t) \propto \int \mathcal{C}(y(t)|x(t)) \times \mathcal{A}(x(t)|x(t-1)) \times \Pr(X_{t-1}|Y_{t-1})\, dx(t)$$

to the joint proposal giving

$$\mathcal{W}_{\text{aux}}(i,t) = \frac{\Pr(X_t|Y_t)}{q(X_t)} = \frac{\mathcal{W}_i(t-1)}{\mathcal{K}_i(t-1)} \times \frac{\mathcal{C}(y(t)|x_i(t)) \times \mathcal{A}(x_i(t)|x_i(t-1))}{q(x_i(t)|x_i(t-1), y(t))} \tag{7.38}$$

TABLE 7.3 Auxiliary *SIR* State–Space Particle Filtering Algorithm

Initialize

$$x_i(0) \sim \Pr(x(0)) \quad W_i(0) = 1/N_p \quad i = 1, \ldots, N_p \qquad \text{[sample]}$$

Auxiliary Weights

Bootstrap Processor: $\{\mathcal{W}_i(t), x_i(t)\}$

$$\hat{m}_i(t) \approx E\{x(t)\} \qquad \text{[bootstrap mean]}$$

Weight calculation

$$K_i(t-1) = \mathcal{W}_i(t-1) \times \mathcal{C}(y(t)|\hat{m}_i(t)) \qquad \text{[bootstrap likelihood]}$$

Weight Normalization

$$\mathcal{K}_i(t-1) = K_i(t-1) \Big/ \sum_{i=1}^{N_p} K_i(t-1) \qquad \text{[auxiliary weight]}$$

Resampling

Select indices $\{j(i)\}$ using $\{\mathcal{K}_{j(i)}(t-1)\}$: $\hat{x}_i(t) \Rightarrow x_{j(i)}(t)$ [resample]

$$\alpha_i(t-1) := \mathcal{W}_{j(i)}(t-1)/\mathcal{K}_{j(i)}(t-1) \qquad \text{[first stage weights]}$$

Importance Sampling Proposal (Optimal)

$$\tilde{x}_i(t) \sim q(\hat{x}_i(t)|x_i(t-1), y(t)) \qquad \text{[sample]}$$

Weight Update

$$\tilde{W}_i(t) = \alpha_i(t-1) \times \frac{\mathcal{C}(y(t)|\tilde{x}_i(t)) \times \mathcal{A}(\tilde{x}_i(t)|x_i(t-1))}{q(\hat{x}_i(t)|x_i(t-1), y(t))} \qquad \text{[weight-update]}$$

Weight Normalization

$$\mathcal{W}_{\text{aux}}(i,t) = \tilde{W}_i(t) \Big/ \sum_{i=1}^{N_p} \tilde{W}_i(t) \qquad \text{[normalize]}$$

Distribution

$$\hat{\Pr}(x(t)|Y_t) \approx \sum_{i=1}^{N_p} \mathcal{W}_{\text{aux}}(i,t)\delta(x(t) - \tilde{x}_i(t)) \qquad \text{[posterior distribution]}$$

which follows directly from substitution of the empirical distributions [6]. Here the *bias* correction, $1/\mathcal{K}_i$, is introduced into the sampler to correct the auxiliary weight (first stage).

We summarize the auxiliary particle filter in Table 7.3. First, the bootstrap technique is executed to generate a set of particles and auxiliary weights at time-step $(t-1)$, that is, $\{\mathcal{W}_i(t), x_i(t)\}$ which are used to estimate the set of means (or modes) $\{\hat{m}_i(t)\}$ as in Eq. 7.33 for the "smoothing" weight calculation of Eq. 7.37 generating the probability masses, $\{\mathcal{K}_i(t-1)\}$. These weights are then used in resampling to generate the set of "most likely" particles and weights under the new sampling indices $\{j(i)\}$, $\{\mathcal{K}_{j(i)}(t-1), x_{j(i)}(t-1)\}$, that is, $\hat{x}_i(t-1) \Rightarrow x_{j(i)}(t-1); i = 1, \ldots, N_p$.

Next, samples are drawn from the optimal proposal and used to update the auxiliary weights using the resampled particles and first stage weights defined by $\alpha_i(t-1) := \mathcal{W}_i(t-1)/\mathcal{K}_i(t-1)$. The posterior distribution is estimated using the auxiliary weights to complete the process.

If the process is governed by severe nonlinearities or contaminated by high process noise, then a single point estimate such as $\hat{m}_i(t)$ does not sufficiently represent the transition probability $\Pr(x(t)|x_i(t-1))$ very well. Therefore, we can expect the *ASIR* performance to be poor even yielding weaker results than that of the bootstrap processor. However, if the process noise is small, implying that a point estimate can characterize the transition probability reasonably well, then the *ASIR* is less sensitive to "outliers" and the weights will be more uniformly balanced resulting in excellent performance compared to the bootstrap. These concepts can be used as an aid to help decide when such a technique is applicable to a given problem.

7.5.3 Regularized Particle Filter

In order to reduce the degeneracy of the weights in the *SIR* processor, resampling was introduced as a potential solution; however, it was mentioned that even though the particles are "steered" to high probability regions, they tend to lose their diversity among other problems introduced by such a procedure [1, 44]. This problem occurs because samples are drawn from a discrete rather than continuous distribution (see Sec. 1.5). Without any attempt to correct this problem, the particles can collapse to a single location giving a poor characterization of the posterior distribution and therefore result in poor processor performance.

One solution to the diversity problem is to develop a continuous rather than discrete approximation to the empirical posterior distribution using the kernel density estimator of Sec. 3.2 and then perform resampling directly from it. This is termed a regularization-step resulting in diversification by a form of "jittering" the particles; thus, the processor is called the *regularized* particle filter (*RPF*). The key idea of the *RPF* is the transformation of the discrete empirical posterior distribution, $\hat{\Pr}(x(t)|Y_t) \rightarrow \Pr(x_t|Y_t)$ in order to resample from an absolutely continuous distribution producing a "new" set of N_p-particles with different locations.

To be more precise let us define the properties of a kernel that can be applied to this problem [7]. A *regularization kernel* $\mathcal{K}(\mathbf{x})$ is a symmetric probability density function such that: (1) $\mathcal{K}(\mathbf{x}) \geq 0$; (2) $\int \mathcal{K}(\mathbf{x})\, d\mathbf{x} = 1$; (3) $\int \mathbf{x}\mathcal{K}(\mathbf{x})\, d\mathbf{x} = 0$; and (4) $\int \|\mathbf{x}^2\|\mathcal{K}(\mathbf{x})\, d\mathbf{x} < \infty$ and for any positive bandwidth $\triangle_x$ the corresponding *rescaled* kernel is defined by

$$\mathcal{K}_{\triangle_x}(\mathbf{x}) = \left(\frac{1}{\triangle_x}\right)^{N_x} \mathcal{K}\left(\frac{\mathbf{x}}{\triangle_x}\right) \quad \text{for } \mathbf{x} \in \mathcal{R}^{N_x \times 1} \tag{7.44}$$

The most important property of the kernel density follows from its regularization property, that is, for any distribution $\mathcal{P}(\mathbf{x}) \in \mathcal{R}^{N_x \times 1}$, the regularization results in an absolutely continuous probability distribution, $\mathcal{K}_{\triangle_x}(\mathbf{x}) * \mathcal{P}(\mathbf{x})$, with $*$ the convolution operator, such that

$$\frac{d}{d\mathbf{x}}[\mathcal{K}_{\triangle_x}(\mathbf{x}) * \mathcal{P}(\mathbf{x})] = \int \mathcal{K}_{\triangle_x}(\mathbf{x} - \alpha)\mathcal{P}(\alpha)\, d\alpha \tag{7.45}$$

If $\mathcal{P}$ is an empirical distribution, then

$$\hat{\mathcal{P}}(\mathbf{x}) \approx \sum_{i=1}^{N_p} \delta(\mathbf{x} - \mathbf{x}_i)$$

for $\mathbf{x}_i \rightarrow \{\mathbf{x}_1, \ldots, \mathbf{x}_{N_p}\}$ a sample from the posterior and therefore

$$\frac{d}{d\mathbf{x}}[\mathcal{K}_{\triangle_x}(\mathbf{x}) * \hat{\mathcal{P}}(\mathbf{x})] = \left(\frac{1}{\triangle_x}\right)^{N_x} \sum_{i=1}^{N_p} \mathcal{W}_i \mathcal{K}\left(\frac{\mathbf{x} - \mathbf{x}_i}{\triangle_x}\right) = \sum_{i=1}^{N_p} \mathcal{W}_i \mathcal{K}_{\triangle_x}(\mathbf{x} - \mathbf{x}_i) \quad (7.46)$$

Both the bandwidth and the kernel are selected in practice to minimize the mean integrated error between the posterior and regularized distribution. Classical kernels result under specialized assumptions such as the Epanechnikov, Box, Triangle, Gaussian etc. (see [7], Chapter 12 for more details).

One of the underlying assumptions of this transformation is that the true posterior $\Pr(x_t|Y_t)$ has a unity covariance which is not the case when implementing the *RPF* technique. Therefore, at each time-step we must estimate the ensemble mean and covariance by the usual sample approach given by

$$\begin{aligned} m(t) &= \frac{1}{N_p} \sum_{i=1}^{N_p} x_i(t) \\ R_{xx}(t) &= \frac{1}{N_p} \sum_{i=1}^{N_p} (x_i(t) - m(t))(x_i(t) - m(t))' \end{aligned} \quad (7.47)$$

With this calculation, we factor the covariance using the Cholesky decomposition to yield the matrix square roots used in a whitening transformation (unity covariance), that is,

$$R_{xx}(t) = L^{1/2}(t) L^{T/2}(t)$$

which leads to the new scaled kernel

$$\mathcal{K}_{\triangle_x}(\mathbf{x}) = \frac{1}{|L^{1/2}|(\triangle_x)^{N_x}} \mathcal{K}\left(\frac{L^{-1/2}\mathbf{x}}{\triangle_x}\right) \quad (7.48)$$

The old particles are then "jittered" by using the step

$$\tilde{x}_i(t) = x_i(t) + \triangle_x L^{\frac{1}{2}}(t) \epsilon_i(t) \quad (7.49)$$

where the $\{\epsilon_i(t)\}$ are drawn from the new scaled kernel above. A typical example of kernel density estimation of a discrete probability mass function is shown in Fig. 7.1 where the circles represent the discrete point masses (impulses) at the particular location and the continuous approximation to the probability density is shown by the

smooth curve provided by the kernel density estimate using a Gaussian window. This completes the *RPF* technique which is summarized in Table 7.4. Next we discuss another popular approach to produce particle diversity.

7.5.4 MCMC Particle Filter

Another approach to increase the diversity in the particle set $\{x_i(t), W_i(t)\}; i = 1, \ldots, N_p$ is to take an *MCMC* step(s) with the underlying invariant distribution targeted as the posterior $\Pr(X_t|Y_t)$ on each particle [45]. The *MCMC* particle filter is available in two varieties: (1) *MCMC*-step(s) with the usual *BSP*; and (2) full *MCMC* iterative filter. The sequential processors use the *MCMC*-steps as part of the resampling process for especially insensitive (weight divergence) problems, while the full *MCMC* iterative processor is available as a separate algorithm typically executed using the Metropolis, Metropolis-Hastings or Gibbs samplers of Chapter 3. We confine our discussion to the *MCMC*-step approach, since we are primarily interested in sequential techniques and refer the interested reader to [1, 23, 24] for the iterative approach.

The main idea is that the particles are distributed as $\Pr(X_t(i)|Y_t)$, then applying a Markov chain transition kernel defined by

$$\mathcal{T}(X_t|X_t(i)) := \Pr(X_t|X_t(i)) \tag{7.50}$$

with posterior invariant distribution such that

$$\Pr(X_t|Y_t) = \int \mathcal{T}(X_t|X_t(i)) \times \Pr(X_t(i)|Y_t)\, dX_t(i) \tag{7.51}$$

continues to result in a particle set with the desired posterior as its invariant distribution. However, the new particle locations after the *move* result in high probability regions of the state–space. It has been shown that by applying the *MCMC* transition kernel that the total variance of the current distribution can only decrease [46]. Any of the *MCMC* methods (*M-H*, *G-S*, *S-S*, etc.) can be incorporated into the *SMC* framework to achieve the desired move occuring after the resampling operation.

Following [46, 47] the objective is to move the set of particles using a combination of importance sampling, resampling and *MCMC* sampling, that is, the approach is to:

- *Initialize* the set of particles yielding: $\{x_i(t)\}; i = 1, \ldots, N_p$
- *Resample* this set to obtain: $\{\hat{x}_i(t)\}; i = 1, \ldots, N_p$
- *Move* using an *MCMC* step(s) to generate the "new" set of particles $\{\tilde{x}_i(t)\}; i = 1, \ldots, N_p$ with $\tilde{x}_i(t) \sim \Pr(X_t|Y_t)$ and transition kernel, $\mathcal{T}(\tilde{x}_i(t)|X_t(i))$

The *move*-step performs one or more iterations of an *MCMC* technique on each selected particle *after* the resampling step with invariant distribution $\Pr(X_t|Y_t)$. Note that *before* the move, the resampled particles are distributed $\hat{x}_i(t) \sim \Pr(X_t|Y_t)$; therefore, the "moved" particles, $\tilde{x}_i(t)$ are approximately distributed by this posterior as well. The move-step improves particle diversity by enriching the particle locations to those highest probability regions.

TABLE 7.4 Regularized Particle Filtering Algorithm

Initialize

Draw: $x_i(0) \sim \Pr(x(0)) \quad W_i(0) = \frac{1}{N_p} \quad i = 1, \ldots, N_p$ [sample]

Importance Sampling

Draw: $x_i(t) \sim \mathcal{A}(x(t)|x_i(t-1))$ [state transition]

Weight Update

$W_i(t) = \mathcal{C}(y(t)|x_i(t)) \Leftarrow C(x(t), u(t), v(t)); \quad v \sim \Pr(v(t))$ [weight/likelihood]

Weight Normalization

$$\mathcal{W}_i(t) = W_i(t) \Big/ \sum_{i=1}^{N_p} W_i(t)$$

Resampling Decision

$$\hat{N}_{eff} = 1 \Big/ \sum_{i=1}^{N_p} W_i^2(t) \quad \text{[effective samples]}$$

$$\hat{N}_{eff} = \begin{cases} \text{Resample} & \leq N_{thresh} \\ \text{Accept} & > N_{thresh} \end{cases} \quad \text{[decision]}$$

Regularization Sample Statistics

$$m(t) = 1/N_p \sum_{i=1}^{N_p} x_i(t) \quad \text{[sample mean]}$$

$$R_{xx}(t) = 1/N_p \sum_{i=1}^{N_p} (x_i(t) - m(t))(x_i(t) - m(t))' \quad \text{[sample covariance]}$$

Factorization

$R_{xx}(t) = L^{1/2}(t) L^{T/2}(t)$ [Cholesky decomposition]

Resampling

$\hat{x}_i(t) \Rightarrow x_i(t)$

Diversification

Draw: $\epsilon_i(t) \sim \mathcal{K}_{\triangle_x}(x(t) - \hat{x}_i(t))$ [sample]

Diversify

$\tilde{x}_i(t) = \hat{x}_i(t) + \triangle_x L^{1/2}(t) \epsilon_i(t)$ [generate sample]

Distribution

$$\hat{\Pr}(x(t)|Y_t) \approx \sum_{i=1}^{N_p} \mathcal{W}_i(t) \delta(x(t) - \tilde{x}_i(t)) \quad \text{[posterior distribution]}$$

For instance, let us "track" the i^{th} particle with corresponding high valued weight. Based on its associated weight (probability), $\hat{x}_i(t) \Rightarrow x_i(t)$ is selected according to

$$\Pr(\hat{x}_i(t) = x_i(t)) = \mathcal{W}_i(t)$$

Resampling results in replication of the i^{th}-particle, N_i-times, producing the resampled set, $\{\hat{x}_{i1}(t), \hat{x}_{i2}(t), \ldots, \hat{x}_{iN_i}(t)\}$. Next the *move-step* moves each replicant $\hat{x}_{ij}(t) \rightarrow \tilde{x}_{ij}(t)$ to a distinct (unique) location in the region of strongest support dictated by $\Pr(X_t|Y_t)$. This provides the "move-step" for the *SMC* technique and mitigates the divergence problem.

To be more specific, let us illustrate the "move" by choosing the Metropolis-Hastings technique of Table 3.1 to perform our *MCMC*-step using the random walk *M-H* approach of Sec. 3.4.

We start with the basic bootstrap *PF* of Sec. 7.5 to obtain the set of resampled particles $\{\hat{x}_i(t)\}$. Using the random walk model, we perform the "move" step to obtain the new set of particles as

$$\tilde{x}_i(t) = \hat{x}_i(t) + \epsilon_i(t) \quad \text{for } \epsilon_i \sim \mathrm{p}_E(\epsilon) \tag{7.52}$$

One choice is $\epsilon_i \sim \mathrm{p}_E(\epsilon) = \mathcal{N}(0, R_{\epsilon\epsilon})$, since it is symmetric, easy to generate and will simplify the calculations even further. The corresponding *acceptance* probability for the *M-H* approach is specified by

$$A(\tilde{x}_i(t), \hat{x}_i(t)) = \min\left\{\frac{\Pr(\tilde{X}_t(i)|Y_t))}{\Pr(\hat{X}_t(i)|Y_t))} \times \frac{q(\hat{x}_i(t)|\tilde{x}_i(t))}{q(\tilde{x}_i(t)|\hat{x}_i(t))}, 1\right\} \tag{7.53}$$

where $\tilde{X}_i(t) := \{\tilde{x}_i(t), X_{t-1}(i)\}$ and $\hat{X}_i(t) := \{\hat{x}_i(t), X_{t-1}(i)\}$ are the augmented sets of joint random particles. Drawing samples from the random walk with (symmetric) Gaussian distribution enables us to simplify the acceptance probability, since $q(\hat{x}_i(t)|\tilde{x}_i(t)) = q(\tilde{x}_i(t)|\hat{x}_i(t))$ canceling these terms in Eq. 7.53 to produce

$$A(\tilde{x}_i(t), \hat{x}_i(t)) = \min\left\{\frac{\Pr(\tilde{X}_t(i)|Y_t))}{\Pr(\hat{X}_t(i)|Y_t))}, 1\right\} \tag{7.54}$$

But from the sequential Bayesian recursion,

$$\Pr(X_t|Y_t) \propto \Pr(y(t)|x(t))\Pr(x(t)|x(t-1)) \times \Pr(X_{t-1}|Y_{t-1})$$

we obtain

$$A(\tilde{x}_i(t), \hat{x}_i(t)) = \min\left\{\frac{\Pr(y(t)|\tilde{x}_i(t)) \times \Pr(\tilde{x}_i(t)|x_i(t-1))}{\Pr(y(t)|\hat{x}_i(t)) \times \Pr(\hat{x}_i(t)|x_i(t-1))}, 1\right\} \tag{7.55}$$

With this information in mind, the implementation of the bootstrap *PF* with (random walk) *MCMC*-step is given in Table 7.5. Another approach to improve particle diversity is using local linearization techniques which can be implemented with any of the classical/modern algorithms of the previous two chapters.

TABLE 7.5 MCMC Particle Filtering Algorithm

Initialize

Draw: $x_i(0) \sim \Pr(x(0)) \quad W_i(0) = \dfrac{1}{N_p} \quad i = 1, \ldots, N_p$ [sample]

Importance Sampling

Draw: $x_i(t) \sim \mathcal{A}(x(t)|x_i(t-1))$ [state transition]

Weight Update

$W_i(t) = \mathcal{C}(y(t)|x_i(t))$ [weight/likelihood]

Weight Normalization

$$\mathcal{W}_i(t) = W_i(t) \Big/ \sum_{i=1}^{N_p} W_i(t)$$

Resampling Decision

$$\hat{N}_{eff} = 1 \Big/ \sum_{i=1}^{N_p} W_i^2(t) \quad \text{[effective samples]}$$

$$\hat{N}_{eff} = \begin{cases} \text{Resample} & \leq N_{thresh} \\ \text{Accept} & > N_{thresh} \end{cases} \quad \text{[decision]}$$

Resampling

$\hat{x}_i(t) \Rightarrow x_i(t)$

Diversification Acceptance Probability

$$A(\tilde{x}_i(t), \hat{x}_i(t)) = \min \left\{ \frac{\mathcal{C}(y(t)|\tilde{x}_i(t)) \times \mathcal{A}(\tilde{x}_i(t)|x_i(t-1))}{\mathcal{C}(y(t)|\hat{x}_i(t)) \times \mathcal{A}(\hat{x}_i(t)|x_i(t-1))}, 1 \right\}$$

Diversify

Draw: $\epsilon_i(t) \sim \mathcal{N}(0, R_{\epsilon\epsilon})$ [sample]

$\tilde{x}_i(t) = \hat{x}_i(t) + \epsilon_i(t)$ [generate sample]

Draw: $u_k \to \mathcal{U}(0, 1)$ [uniform sample]

Decision

$$x_i(t) = \begin{cases} \tilde{x}_i(t) & \text{if } u_k < A(x_i, \hat{x}_i) \\ \hat{x}_i(t) & \text{otherwise} \end{cases}$$

Distribution

$$\hat{\Pr}(x(t)|Y_t) \approx \sum_{i=1}^{N_p} \mathcal{W}_i(t) \delta(x(t) - x_i(t)) \quad \text{[posterior distribution]}$$

7.5.5 Linearized Particle Filter

Attempts to approximate the minimum variance importance proposal distribution of Sec. 7.2 continue to evolve [24]. The motivation for this is based on inadequacies created by selecting the transition prior of Sec. 7.5.1 as the proposal leading to the popular bootstrap algorithm [16] of Table 7.2. As mentioned previously, the bootstrap approach requires resampling to mitigate the particle depletion problem and lack of incorporating the most recent measurement in the weight update. These reasons have led to the development of the *PF* that incorporates the latest measurement sample. One way to do so is to generate an approximately Gaussian importance proposal based on linearization methods [4, 7, 9] with the idea of selecting

$$\Pr(x(t)|X_{t-1}, Y_t) \approx q_{\mathcal{N}}(x(t)|Y_t) \tag{7.56}$$

as a Gaussian proposal. This approach is used to provide coverage of the actual posterior due to its long-tailed distribution while incorporating the latest available measurement. This Gaussian proposal is the result of marginalizing the prior state, $x(t-1)$, of the minimum variance proposal. That is,

$$q_{\mathcal{N}}(x(t)|Y_t) \rightarrow \Pr(x(t)|Y_t) = \int \Pr(x(t)|x(t-1), y(t)) \times \Pr(x(t-1)|Y_{t-1})\, dx(t-1) \tag{7.57}$$

In a sense this implicit marginalization effectively averages the proposal with respect to the previous posterior, $\Pr(x(t-1)|Y_{t-1})$, which incorporates all of the "information" about $x(t-1)$. Thus, we see that by choosing the Gaussian importance distribution as our proposal enables us to implicitly incorporate all of the knowledge available about the previous state as well as incorporate the current measurement, $y(t)$. These features make $q_{\mathcal{N}}(x(t)|Y_t)$ a reasonable choice as long as it provides the overlapping support or coverage of the desired posterior [9].

One of the linearization approaches to implementing the minimum variance importance function, $q_{MV}(x(t)|x(t-1), y(t))$ is to estimate it as a Gaussian prior, that is,

$$q_{MV}(x(t)|x(t-1), y(t)) \sim \mathcal{N}(\hat{x}(t|t), \tilde{P}(t|t)) \tag{7.58}$$

where the associated (filtered) conditional mean, $\hat{x}(t|t) = E\{x(t)|Y_t\}$ and error covariance, $\tilde{P}(t|t) = E\{\tilde{x}(t|t)\tilde{x}'(t|t)\}$ for $\tilde{x}(t|t) = x(t) - \hat{x}(t|t)$ are obtained from an additional estimation scheme.

There are a variety of choices available, each evolving from either the classical (linearized, extended or iterated Kalman filters) or the modern unscented (sigma-point) Kalman filter. Further alternatives are also proposed to these approaches [4], but we will confine our discussion to these popular and readily available approaches [9]. In each of these implementations a linearization, either of the nonlinear dynamics and measurement models in the classical case or in the statistical linearization (unscented transformation) as in the unscented or sigma-point case occurs. All of

these "linearized-based" processors provide the updated or filtered conditional mean estimate

$$\hat{x}(t|t) = \hat{x}(t|t-1) + K(x^*(t))e(t) \tag{7.59}$$

where

$\hat{x}(t|t-1)$ is the predicted conditional mean, $E\{x(t)|Y_{t-1}\}$;
$K(x^*(t))$ is the gain or weight based on the particular linearization technique; and
$e(t)$ is the innovation sequence.

Here the choices are:

$x^*(t) \rightarrow x_o(t)$ is a reference trajectory in the *linearized* case; or
$x^*(t) \rightarrow \hat{x}(t|\alpha)$ is a *extended* or *iterated cases*; and
$x^*(t) \rightarrow \chi_i(t|t)$ is a sigma-point in the *unscented case*.

The error covariance is much different in each case, that is,

$$\tilde{P}(t|t) = (I - K(x^*(t))C[x^*(t)])\tilde{P}(t|t-1) \tag{7.60}$$

in the linearized and extended cases with measurement Jacobian

$$C[x^*(t)] = \left.\frac{\partial c_i[x]}{\partial x_j}\right|_{x=x^*(t)} \tag{7.61}$$

and, of course,

$$\tilde{P}(t|t-1) = A[x^*(t)]\tilde{P}(t-1|t-1)A'[x^*(t)] + R_{ww}(t-1) \tag{7.62}$$

for $A[x^*(t)] = \left.\frac{\partial a_i[x]}{\partial x_j}\right|_{x=x^*(t)}$.

In the sigma-point (unscented) case, we have

$$\tilde{P}(t|t) = \tilde{P}(t|t-1) - K(x^*(t))R_{ee}(t)K'(x^*(t)) \tag{7.63}$$

So we see that depending on the linearization method or the particular classical or unscented processor we select, we will generate the Gaussian prior at each time-step which is used to obtain the minimum variance importance distribution. Thus, we see why they are called the class of "local" linearization-based particle filters. The linearized particle filter algorithm is shown in Table 7.6 where we see that the conditional mean and covariance are estimated in each time-step of the algorithm and particles are drawn from

$$x_i(t) \rightarrow \hat{\Pr}(x(t)|Y_t) \sim \mathcal{N}(\hat{x}(t|t), \tilde{P}(t|t)) \tag{7.64}$$

the updated "Gaussian" estimates and the weights also follow the importance distribution of Eq. 7.58 given by

$$W_i(t) = \frac{\mathcal{C}(y(t)|x_i(t)) \times \mathcal{A}(x(t)|x_i(t-1))}{q_{MV}(x(t)|x_i(t-1), y(t))} \tag{7.65}$$

TABLE 7.6 Linearized Particle Filtering Algorithm

Initialize

Draw: $x_i(0) \sim \Pr(x(0)) \quad W_i(0) = \frac{1}{N_p} \quad i = 1, \ldots, N_p$ [sample]

Linearization

LZKF/EKF/IEKF/UKF Processor

$\{x_i(t), \tilde{P}_i(t)\} = \mathcal{N}(\hat{x}(t|t), \tilde{P}(t|t))$

Importance Sampling

$x_i(t) \sim \mathcal{N}(\hat{x}(t|t), \tilde{P}(t|t))$ [state draw]

Weight Update

$$W_i(t) = \frac{\mathcal{C}(y(t)|x_i(t)) \times \mathcal{A}(x(t)|x_i(t-1))}{q_{MV}(x(t)|x_i(t-1), y(t))}$$ [MV weight]

for

$q_{MV}(x(t)|x_i(t-1), y(t)) = \mathcal{N}(x_i(t), \tilde{P}_i(t))$

Weight Normalization

$$\mathcal{W}_i(t) = W_i(t) \Big/ \sum_{i=1}^{N_p} W_i(t)$$

Resampling Decision

$$\hat{N}_{eff} = 1 \Big/ \sum_{i=1}^{N_p} W_i^2(t)$$ [effective samples]

$$\hat{N}_{eff} = \begin{cases} \leq N_{thresh} & \text{Resample} \\ > N_{thresh} & \text{Accept} \end{cases}$$ [decision]

Resampling

$\hat{x}_i(t) \Rightarrow x_i(t)$

Distribution

$$\hat{\Pr}(x(t)|Y_t) \approx \sum_{i=1}^{N_p} \mathcal{W}_i(t) \delta(x(t) - \hat{x}_i(t))$$ [posterior distribution]

This completes the linearized particle filters, next we consider some of the practical considerations for design.

7.6 PRACTICAL ASPECTS OF PARTICLE FILTER DESIGN

Monte Carlo methods, even those that are model-based, are specifically aimed at providing a reasonable estimate of the underlying posterior distribution; therefore,

performance testing typically involves estimating just "how close" the estimated posterior is to the "true" posterior. However, within a Bayesian framework, this comparison of posteriors only provides a measure of relative performance but does not indicate just how well the underlying model embedded in the processor "fits" the measured data. Nevertheless, these closeness methods are usually based on the Kullback-Leibler (*KL*) divergence measure [58] providing such an answer. However in many cases we do not know the true posterior and therefore we must resort to other means of assessing performance such as evaluating mean-squared error (*MSE*) or more generally checking the validity of the model by evaluating samples generated by the prediction or the likelihood cumulative distribution to determine whether or not the resulting sequences have evolved from a uniform distribution and are *i.i.d.* (see Sec. 3.3)—analogous to a whiteness test for Gaussian sequences.

Thus, *PF* are essentially sequential estimators of the posterior distribution employing a variety of embedded models to achieve meaningful estimates. In contrast to the *BP* designs which are typically based on Gaussian assumptions, the *PF* have no such constraints per se. In the linear case, a necessary and sufficient condition for optimality of the linear *BP* is that the corresponding innovations or residual sequence must be zero-mean and white (see Sec. 5.6 for details). In lieu of this constraint, a variety of statistical tests (whiteness, uncorrelated inputs, etc.) were developed in Sec. 5.6 evolving from this known property. When the linear Bayesian processors were "extended" to the nonlinear case, the same tests were performed based on approximate Gaussian assumptions. Clearly, when noise is additive Gaussian these arguments can still be applied. These same statistical tests can also be performed based on the on the innovations or residual sequences resulting from the *PF* estimates (*MAP*, *ML*, *MMSE*) inferred from the estimated posterior distribution. However, some other more meaningful performance tests for the *PF* can also be applied for improved design and performance evaluation.

7.6.1 Posterior Probability Validation

Much effort has been devoted to the validation problem with the most significant results evolving from the information theoretical point of view [59]. Following this approach, we start with the basic ideas and quickly converge to a reasonable solution to the *distribution validation* problem [59, 60].

Let us first define some concepts about probabilistic information necessary for the development. 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 ω_i such that $X(\omega_i) = x_i$, is

$$\mathcal{I}(x_i) = -\log_b \Pr(X(\omega_i) = x_i) = -\log_b \Pr(x_i) \tag{7.66}$$

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). The *entropy* or *average*

information is defined by

$$\mathcal{H}(x_i) := E_X\{\mathcal{I}(x_i)\} = \sum_{i=1}^{N} \mathcal{I}(x_i)\Pr(x_i) = -\sum_{i=1}^{N} \Pr(x_i)\log_b \Pr(x_i) \tag{7.67}$$

Mutual information is defined in terms of the information available in the occurence of the event $Y(\omega_j) = y_j$ about the event $X(\omega_i) = x_i$ or

$$\mathcal{I}(x_i; y_j) = \log_b \frac{\Pr(x_i|y_j)}{\Pr(x_i)} = \log_b \Pr(x_i|y_j) - \log_b \Pr(x_i) \tag{7.68}$$

Now using these concepts, we take the information theoretic approach to distribution estimation following [59, 60]. Since many processors are expressed in terms of their "estimated" probability distributions, quality or "goodness" can be evaluated by its similarity to the true underlying probability distribution generating the measured data.

Suppose $\Pr(x_i)$ is the *true* discrete posterior probability distribution and $\hat{\Pr}(\hat{x}_i)$ is the estimated distribution. Then the *Kullback-Leibler Information* (*KL*) quantity of the true distribution relative to the estimated is defined by using

$$\begin{aligned} \mathcal{I}_{KL}(\Pr(x_i); \hat{\Pr}(\hat{x}_i)) &:= E_X\left\{\ln \frac{\Pr(x_i)}{\hat{\Pr}(\hat{x}_i)}\right\} = \sum_{i=1}^{N} \Pr(x_i) \ln \frac{\Pr(x_i)}{\hat{\Pr}(\hat{x}_i)} \\ &= \sum_{i=1}^{N} \Pr(x_i) \ln \Pr(x_i) - \sum_{i=1}^{N} \Pr(x_i) \ln \hat{\Pr}(\hat{x}_i) \end{aligned} \tag{7.69}$$

where we chose $\log_b = \ln$. The *KL* possesses some very interesting properties which we state without proof (see [60] for details) such as

1. $\mathcal{I}_{KL}(\Pr(x_i); \hat{\Pr}(\hat{x}_i)) \geq 0$
2. $\mathcal{I}_{KL}(\Pr(x_i); \hat{\Pr}(\hat{x}_i)) = 0 \Leftrightarrow \Pr(x_i) = \hat{\Pr}(\hat{x}_i)\, \forall i$
3. The negative of the *KL* is the *entropy*, $\mathcal{H}_{KL}(\Pr(x_i); \hat{\Pr}(\hat{x}_i))$

The second property implies that as the *estimated* posterior distribution approaches the *true* distribution, then the value of the *KL* approaches *zero* (minimum). Thus, investigating Eq. 7.69, 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 estimated posterior relative to the true distribution, that is,

$$\mathcal{L}(\hat{x}_i) := E_X\{\ln \hat{\Pr}(\hat{x}_i)\} = \sum_{i=1}^{N} \Pr(x_i) \ln \hat{\Pr}(\hat{x}_i) \tag{7.70}$$

where $\mathcal{L}(\hat{x}_i)$ is defined as the *average log-likelihood* of the random variable of value $\ln \hat{\Pr}(\hat{x}_i)$. Clearly, the *larger* the average log-likelihood, the *smaller* the *KL* implying a *better* model.

The third property, *entropy*, is approximately equal to $\frac{1}{N}$ times the probability that the relative frequency distribution of N measurements obtained from the estimated posterior equals the true distribution.

The $\mathcal{I}_{KL}$ is applied frequently to parameter estimation/system identification problems to estimate the intrinsic order of the unknown system [5]. Two popular information metrics have evolved from this theory: the Akaike Information Criterion (*AIC*) and the minimum data length (*MDL*) description [59, 60, 62, 63]. Both are used to perform *order estimation* and are closely related as shown below

$$\begin{aligned} AIC(\eta) &= -\ln R_{\epsilon\epsilon} + 2\frac{\eta}{N} \\ MDL(\eta) &= -\ln R_{\epsilon\epsilon} + \frac{\eta}{2}\ln N \end{aligned}$$

where η is the system order, ϵ is the one-step prediction error with corresponding covariance $R_{\epsilon\epsilon}$ and N is the number of samples (data) values.

However, our interest lies in comparing two probability distributions to determine "how close" they are to one another. Even though $\mathcal{I}_{KL}$ does quantify the difference between the true and estimated distributions, unfortunately it is not a distance measure due to its lack of symmetry. However, the *Kullback divergence* (*KD*) defined by a combination of $\mathcal{I}_{KL}$

$$\mathcal{J}_{KD}(\Pr(x_i); \hat{\Pr}(\hat{x}_i)) = \mathcal{I}_{KL}(\Pr(x_i); \hat{\Pr}(\hat{x}_i)) + \mathcal{I}_{KL}(\hat{\Pr}(\hat{x}_i); \Pr(x_i)) \tag{7.71}$$

is a distance measure between distributions indicating "how far" one is from the other. Consider the following example of this calculation.

Example 7.2

We would like to calculate the *KD* for two Gaussian distributions, $\mathrm{p}_i(x) \sim \mathcal{N}(m_i, V_i)$; $i = 1, 2$ to establish a closeness measure. First, we calculate the *KL* information,

$$\mathcal{I}_{KL}(\mathrm{p}_1(x); \mathrm{p}_2(x)) = E_{\mathrm{p}_1}\left\{\ln \frac{\mathrm{p}_1(x)}{\mathrm{p}_2(x)}\right\} = E_{\mathrm{p}_1}\left\{\frac{1}{2}\ln \frac{V_2}{V_1} + \frac{(x-m_2)^2}{2V_2} - \frac{(x-m_1)^2}{2V_1}\right\}$$

Now performing the expectation term-by-term gives

$$\mathcal{I}_{KL}(\mathrm{p}_1(x); \mathrm{p}_2(x)) = \frac{1}{2}\ln \frac{V_2}{V_1} + \frac{1}{2V_2}\int (x-m_2)^2 \mathrm{p}_1(x)\,dx - \frac{1}{2V_1}\int (x-m_1)^2 \mathrm{p}_1(x)\,dx$$

Since the last term (under the integral) is the variance, V_1, it is simply $-\frac{1}{2}$ and therefore all we need do is expand and perform the integration of the second integral

term-by-term to give

$$\mathcal{I}_{KL}(\mathrm{p}_1(x);\mathrm{p}_2(x)) = \frac{1}{2}\ln\frac{V_2}{V_1} + \frac{1}{2V_2}\left[\int x^2\mathrm{p}_1(x)\,dx - 2m_2\int x\mathrm{p}_1(x)\,dx + m_2^2\int \mathrm{p}_1(x)\,dx\right] - \frac{1}{2}$$

or identifying terms from the properties of moments, we obtain

$$\mathcal{I}_{KL}(\mathrm{p}_1(x);\mathrm{p}_2(x)) = \frac{1}{2}\ln\frac{V_2}{V_1} + \frac{1}{2V_2}[(V_1 + m_1^2) - 2m_2m_1 + m_2^2] - \frac{1}{2}$$

Finally, we have

$$\mathcal{I}_{KL}(\mathrm{p}_1(x);\mathrm{p}_2(x)) = \frac{1}{2}\ln\frac{V_2}{V_1} + \frac{V_1 + (m_1 - m_2)^2}{2V_2} - \frac{1}{2}$$

Performing the same calculation, we get

$$\mathcal{I}_{KL}(\mathrm{p}_2(x);\mathrm{p}_1(x)) = \frac{1}{2}\ln\frac{V_1}{V_2} + \frac{V_2 + (m_1 - m_2)^2}{2V_1} - \frac{1}{2}$$

and therefore the *KD* is

$$\begin{aligned}\mathcal{J}_{KD}(\mathrm{p}_1(x);\mathrm{p}_2(x)) &= \mathcal{I}_{KL}(\mathrm{p}_1(x);\mathrm{p}_2(x)) + \mathcal{I}_{KL}(\mathrm{p}_2(x);\mathrm{p}_1(x)) \\ &= \frac{V_1^2 + (m_1 - m_2)^2(V_1 + V_2) + V_2^2}{2V_1V_2} - 1\end{aligned}$$

This completes the example. △△△

The *KL* and therefore the *KD* can also be determined by probability distribution estimation using *MC* sampling techniques [64, 65]. As another example of this approach consider how to apply the *KL* information to distinguish between a unimodal Gaussian and a Gaussian mixture.

Example 7.3

Suppose we would like to test whether a given data set is from a Gaussian distribution specified by $\mathcal{N}(m, V)$ or from a Gaussian mixture distribution specified by $p\mathcal{N}(m_1, V_1) + (1-p)\mathcal{N}(m_2, V_2)$ where p is the mixing coefficient (probability). The

Kullback divergence can easily be calculated and compared to a bound α to determine "how close" the data is to a mixture or a Gaussian, that is,

$$\mathcal{J}_{KD}(\Pr(x_i); \hat{\Pr}(\hat{x}_i)) = \mathcal{J}_{KD}(p\mathcal{N}(m_1, V_1) + (1-p)\mathcal{N}(m_2, V_2); \mathcal{N}(m, V)) < \alpha$$

In order to perform the test we choose m and $\sqrt{V}$ to minimize the KD above. Solving we obtain

$$m = pm_1 + (1-p)m_2; \quad \text{and} \quad V = pV_1 + (1-p)V_2 + p(1-p)(m_1 - m_2)^2$$

A typical bound of $\alpha = 0.1$ appears to perform well in distinguishing a single Gaussian from a mixture. $\triangle\triangle\triangle$

This completes the section, next we discuss an entirely different approach to this distribution problem by investigating "goodness of fit" testing.

7.6.2 Model Validation Testing

Of the major practical concerns with all model-based processors is whether or not the model embedded in the processor "matches" the underlying phenomena and can be used to extract meaningful information from the noisy measurements. As mentioned, in the classical Gaussian-based techniques, the zero-mean/whiteness testing of the innovations is a critical measure of this match. These properties are also used extensively for linearized models evolving from nonlinear dynamic systems as well [5]. In all of these cases the distributions are considered unimodal and typically Gaussian.

In the non-unimodal (non-Gaussian) case the diagnostics are more complicated. The roots of *MC* model diagnostics lie in the basic Uniform Transformation Theorem of Sec. 3.3 and the works of Rosenblatt [48] and Smith [49] under the general area of "goodness-of-fit" statistical tests [39, 40, 50, 51, 57, 58].

The fundamental idea is based on analyzing the *predicted measurement cumulative distribution*, $\mathrm{P}_Y(y(t)|Y_{t-1})$. A processor is considered consistent *only* if the measurement $y(t)$ is governed by the statistics of its predicted cumulative distributions. Therefore, *validation* consists of statistically testing that the measurements "match" the predictions using the underlying model embedded in the processor. By defining the *residual sequence* as

$$\epsilon(t) := \mathrm{P}_Y(y(t)|Y_{t-1}) = \Pr(Y(t) \leq y(t)|Y_{t-1}) = \int_{Y \leq y} \Pr(y'(t)|Y_{t-1}) dy'(t) \quad (7.72)$$

we must show that $\{\epsilon(t)\}$ is a valid realization of an independent, identically distributed, process uniformally distributed on the interval $[0, 1]$ given the measurements Y_{t-1}. Thus, the statistical test validates whether or not the sequence is $\epsilon(t) \sim \mathcal{U}(0, 1)$ (component-wise for the vector case).

More formally, we use Rosenblatt's theorem[3] that states, if $y(t)$ is a continuous random vector and the underlying model is "valid", then the corresponding sequence $\{\epsilon(t)\}$ is *i.i.d.* on $[0,1]$. Under the assumption that the residual sequence is standard uniform, then we can transform to obtain an equivalent Gaussian sequence [48] such that

$$\nu(t) = \Phi^{-1}(\epsilon(t)) \qquad \text{for } \nu \sim \mathcal{N}(0,1) \text{ with } \epsilon \sim \mathcal{U}(0,1) \tag{7.73}$$

where Φ^{-1} is the inverse standard Gaussian cumulative distribution. Once the residual sequence is transformed, then all of the classical Gaussian statistical tests can be performed to ensure validity of the underlying model.

With this in mind we are still required to solve two basic problems: (1) the estimation of the residual sequence $\epsilon(t)$ or equivalently the estimation of the predictive measurement cumulative distribution, $\mathrm{P}_Y(y(t)|Y_{t-1})$; and (2) the diagnostic statistical testing of $\epsilon(t)$ or equivalently $\nu(t)$, that is, demonstrating that $\epsilon \sim \mathcal{U}(0,1)$ or $\nu \sim \mathcal{N}(0,1)$.

The *key* to estimating the residual sequence is based on representing the predictive cumulative distribution as an infinite mixture [39, 40, 51]

$$\epsilon(t) = \mathrm{P}_Y(y(t)|Y_{t-1}) = \int \mathrm{P}_Y(y(t)|x(t)) \times \Pr(x(t)|Y_{t-1})\, dx(t) \tag{7.74}$$

An *MC* approach to this estimation problem is to "particulate" the required underlying distribution and estimate the empirical prediction cumulative distribution or equivalently the residuals. Perhaps the simplest *MC* technique is to sample from the predicted state distribution directly "when possible", that is, if we could replace the prediction distribution with its empirical representation, then we could obtain an estimated residual [57], that is,

$$\hat{\epsilon}(t) = \int \mathrm{P}_Y(y(t)|x(t)) \times \left[\frac{1}{N_p} \sum_{i=1}^{N_p} \delta\left(x(t) - x_i(t)\right) \right] dx(t) \tag{7.75}$$

or simply

$$\hat{\epsilon}(t) = \frac{1}{N_p} \sum_{i=1}^{N_p} \mathrm{P}_Y(y(t)|x_i(t)) \tag{7.76}$$

However, if direct sampling of the predicted state distribution is *not possible*, then the *predictive decomposition* into the transition prior and posterior can be

[3] The theorem states that for a given random vector $y \in \mathcal{R}^{N_y \times 1}$ with corresponding distribution $P_Y(y)$, the transformed vector, $\epsilon = Ty$, is uniformly distributed on the N_y-hypercube for $\Pr(\epsilon) = \prod_{i=1}^{N_y} \epsilon_i$ when $0 \le \epsilon_i \le 1$. The transformation, T is given by $\epsilon_i = \mathrm{P}_Y(y_i|Y_{i-1}); i = 1, \ldots, N_y$ [48].

accomplished, that is,

$$\Pr(x(t)|Y_{t-1}) = \int \Pr(x(t)|x(t-1)) \times \Pr(x(t-1)|Y_{t-1})\, dx(t-1) \tag{7.77}$$

or using the state–space representation of the transition prior

$$\Pr(x(t)|Y_{t-1}) = \int \mathcal{A}(x(t)|x(t-1)) \times \Pr(x(t-1)|Y_{t-1})\, dx(t-1) \tag{7.78}$$

and performing a particle approximation, we obtain the empirical estimate of the $(t-1)$-step posterior as

$$\hat{\Pr}(x(t-1)|Y_{t-1}) = \sum_{k=1}^{N_p} \mathcal{W}_k(t-1)\delta\left(x(t-1) - x_k(t-1)\right) \tag{7.79}$$

Substituting this expression into Eq. 7.78 gives the prediction distribution

$$\begin{aligned}\hat{\Pr}(x(t)|Y_{t-1}) &= \int \mathcal{A}(x(t)|x(t-1)) \\ &\quad \times \left[\sum_{k=1}^{N_p} \mathcal{W}_k(t-1)\delta\left(x(t-1) - x_k(t-1)\right)\right] dx(t-1) \\ &= \sum_{k=1}^{N_p} \mathcal{W}_k(t-1)\mathcal{A}(x(t)|x_k(t-1))\end{aligned} \tag{7.80}$$

and the residual of Eq. 7.74 becomes

$$\hat{\epsilon}(t) = \int \mathrm{P}_Y(y(t)|x(t)) \left[\sum_{k=1}^{N_p} \mathcal{W}_k(t-1)\mathcal{A}(x(t)|x_k(t-1))\right] dx(t) \tag{7.81}$$

If we draw a sample, $x_i(t)$, from the transition prior, $x_i \sim \mathcal{A}(x(t)|x_k(t-1))$ and use the perfect sample approximation,

$$\mathcal{A}(x(t)|x_k(t-1)) \approx \delta(x(t) - x_i(t)) \tag{7.82}$$

then substituting into Eq. 7.81 for the transition distribution yields

$$\hat{\epsilon}(t) = \sum_{k=1}^{N_p} \mathcal{W}_k(t-1) \int \mathrm{P}_Y(y(t)|x(t)) \times \delta\left(x(t) - x_i(t)\right)\, dx(t) \tag{7.83}$$

giving the final expression for the estimated residual as [39, 40, 51]

$$\hat{\epsilon}(t) = \sum_{k=1}^{N_p} \mathcal{W}_k(t-1) \mathrm{P}_Y(y(t)|x_i(t)) \tag{7.84}$$

for

$$\mathrm{P}_Y(y(t)|x_i(t)) = \int_{Y \le y(t)} \mathcal{C}(y(t)|x_i(t))\, dy(t)$$

which can be estimated through direct integration or using an empirical *CDF* [14], if necessary.

Once we have obtained the estimate of the residual $\epsilon(t)$ (or equivalently the transformed residual $\nu(t)$), we can perform diagnostic tests to evaluate the "goodness-of-fit" and therefore evaluate the validity of the embedded dynamic model.

There are a variety of diagnostic tests that can be performed which include: χ^2-testing (*C-sq*), Kolmogorov-Smirnov (*K-S*) tests, normality testing (graphically), zero-mean/whiteness testing, etc. Here we concentrate on the *C-sq* and *K-S* as well as moment testing of the transformed residuals. Zero-mean/whiteness testing was already discussed in Sec. 5.6.

Any of the Kalman techniques can also be used to generate an approximation to the sequence of residuals or prediction cumulative distribution, using the empirical *PDF*, *EKF*, *UKF*, Gauss-Hermite (*G-H*) grid-based integration as well as the Gaussian mixture approach, that is, Gaussian sums (*G-S*) (see [51, 66–69] for more details).

7.6.2.1 Chi-Square Model Validation Test Chi-square (*C-Sq*) tests are hypothesis tests with the null hypothesis, $\mathcal{H}_o$, that an N-length sequence of data is a random sample from a specified distribution against the alternative that it is not [14, 15]. It is usually applied to Gaussian distributions. *C-Sq* tests are based on the fact that the exponent of a Gaussian distribution, that is, the square of the ratio of the random variable minus its mean divided by its standard deviation is chi-square distributed with N degrees of freedom, that is, $(x - \mu/\sigma)^2 \sim \chi^2(N)$. However, this test can be applied to any distribution.

For instance, if the random variable is binomially distributed with $\mathrm{B}(N, \mathrm{p})$ for p the *success probability*, then it takes the same form as the exponent above—it is distributed (in the limit) as $\chi^2(1)$. Extending this to a multinomial distribution with parameters N and $\mathrm{p}(i)$ for $i = 1, \ldots, k-1$; $\mathcal{M}(N, \{\mathrm{p}(i)\})$, then in the limit as $N \to \infty$ the test statistic, say $\mathcal{C}_{k-1}$, has an approximate $\chi^2(k-1)$ distribution.[4]

Hypothesis testing that $\mathcal{H}_o$ is true using the test statistic is specified by the value κ (probability bound) of the $\chi^2(k-1)$ using $\Pr(\mathcal{C}_{k-1} \ge \kappa) = \alpha$ for α the significance level of the test. Thus, if the test statistic is *less than* κ the null hypothesis is accepted and the selected distribution is correct.

[4] To be more specific, if the *i.i.d.* random variables, $y_1, \ldots, y_{k-1}$ are multinomially distributed with $y_k = N - \sum_{i=1}^{k-1} y_i$ and $\mathrm{p}(k) = 1 - \sum_{i=1}^{k-1} \mathrm{p}(i)$, then the statistic, $\mathcal{C}_{k-1} = \sum_{i=1}^{k-1} (y_i - N_p(i))^2 / N_p(i)$ is $\mathcal{C}_{k-1} \sim \chi^2(k-1)$ [15].

For our problem, a goodness-of-fit test of the residuals follows directly from their uniformity property of an adequate model. The most common statistical tests for uniformity follows from the χ^2-test based on segmenting the estimated residual sequence on [0,1] into subintervals and testing. The chi-square statistical test can be used to decide whether or not the residual sequence $\epsilon(t)$ is $\mathcal{U}(0,1)$ or equivalently the transformed residual, $\nu(t)$ is $\mathcal{N}(0,1)$.

The *C-Sq* test statistic for our problem is given by

$$\mathcal{C}_{N_\epsilon-1} = \sum_{i=1}^{N_\epsilon} \frac{(n_\epsilon(i) - \overline{\epsilon})^2}{\overline{\epsilon}} \tag{7.85}$$

where N is the total number of residual samples; N_ϵ is the number of bins (equally spaced subintervals); $n_\epsilon(i)$ is the number of residual counts in the i^{th}-bin (subinterval); and $\overline{\epsilon}$ is the expected number of counts per bin given by $\overline{\epsilon} = \frac{N}{N_\epsilon}$.

If the residual sequence is uniform, then $\mathcal{C}_{N_\epsilon-1} \sim \chi^2(N_\epsilon - 1)$ and κ is compatible with a $\chi^2(N_\epsilon - 1)$ distribution at significance level, α. Therefore, if $\mathcal{C}_{N_\epsilon-1} \leq \kappa$ the null hypothesis that the residual sequence is uniformly distributed is accepted and the model is adequate (validated) otherwise it is rejected. Thus, the χ^2-model validation test is:

- *Partition* the N-sample residual sequence into N_ϵ bins (equally spaced subintervals)
- *Count* the number of residual samples in each bin, $n_\epsilon(i)$; $i = 1, \ldots, N_\epsilon$
- *Calculate* the expected bin count, $\overline{\epsilon} = N/N_\epsilon$
- *Calculate* the test statistic $\mathcal{C}_{N_\epsilon-1}$ of Eq. 7.85
- *Test* that $\mathcal{C}_{N_\epsilon-1} \leq \kappa$ [Accept $\mathcal{H}_o$]

Example 7.4

Suppose we have a residual sequence, $\epsilon(t)$, scaled on [0, 1] of $N = 1000$ samples and we would like to test that it is uniformly distributed using the χ^2-model validation test. We partition the sequence into $N_\epsilon = 10$ bins; therefore, the expected counts per bin is $\overline{\epsilon} = 100$. At the $\alpha = 5\%$ significance level, the test statistic,

$$\mathcal{C}_{N_\epsilon-1} = 3.22$$

is less than κ (probability bound) accepting the null hypothesis that the sequence is uniform and therefore the model is validated. △△△

Next we consider another more robust method for goodness-of-fit testing.

7.6.2.2 Kolmogorov-Smirnov Model Validation Test The chi-square goodness-of-fit test suffers from the limitations of arbitrary interval widths and the requirement of large data sets. An alternative or complementary approach is the

Kolmogorov-Smirnov (K-S) goodness-of-fit test that is based on deciding whether or not a hypothesized (e.g., uniform) or estimated cumulative distribution characterizes a given data sequence (e.g., residual). The hypothesis test is given by:

$$\begin{aligned} \mathcal{H}_o&: \quad \hat{\mathrm{P}}_E(\epsilon) = \mathrm{P}_o(\epsilon) \\ \mathcal{H}_1&: \quad \hat{\mathrm{P}}_E(\epsilon) \neq \mathrm{P}_o(\epsilon) \end{aligned} \tag{7.86}$$

where $\hat{\mathrm{P}}_E$ is the underlying (estimated) population CDF and P_o is the hypothesized CDF. The test statistic used in making the decision is:

$$\mathcal{K} = \max_{\epsilon} |\mathrm{P}_E(\epsilon) - \mathrm{P}_o(\epsilon)| \tag{7.87}$$

where $\hat{\mathrm{P}}_E$ is given by the empirical distribution function estimate

$$\hat{\mathrm{P}}_N(\epsilon) = \frac{N_\epsilon}{N} \rightarrow \Pr(E \leq \epsilon) = \mathrm{P}_E(\epsilon) \quad \text{as } N \rightarrow \infty \tag{7.88}$$

For large N, $\mathcal{K} \approx 0$ with $\mathcal{H}_o$ *true* while for $\mathcal{H}_1$ true, $\mathcal{K}$ is close to the maximum difference. Therefore, we reject $\mathcal{H}_o$ if $\mathcal{K} > \kappa$ with κ a constant determined by the level-of-significance of the hypothesis test, α. That is,

$$\alpha = \Pr(\mathcal{K} > \kappa \,|\, \mathcal{H}_o) \approx 2e^{-2N\kappa^2} \tag{7.89}$$

Thus the K-S-test is:

- *Estimate* the empirical CDF, $\hat{\mathrm{P}}_E(\epsilon)$
- *Calculate* K-S test statistic $\mathcal{K}$ from Eq. 7.87
- *Test* that

$$\mathcal{K} < \sqrt{-\frac{1}{2N} \ln \frac{\alpha}{2}} \quad [\text{Accept } \mathcal{H}_o]$$

Example 7.5

We have a residual sequence, $\epsilon(t)$, scaled on [0, 1] of $N = 25$ samples and we would like to perform the K-S test that the samples are uniformly distributed at the $\alpha = 5\%$ significance level. The test statistic,

$$\mathcal{K} = 0.17$$

is less than $\kappa = 0.26$ accepting the null hypothesis that the sequence is uniform and therefore the model is valid. The test is shown in Fig. 7.11 as the location of the maximum deviation between the hypothesized and empirical distributions. △△△

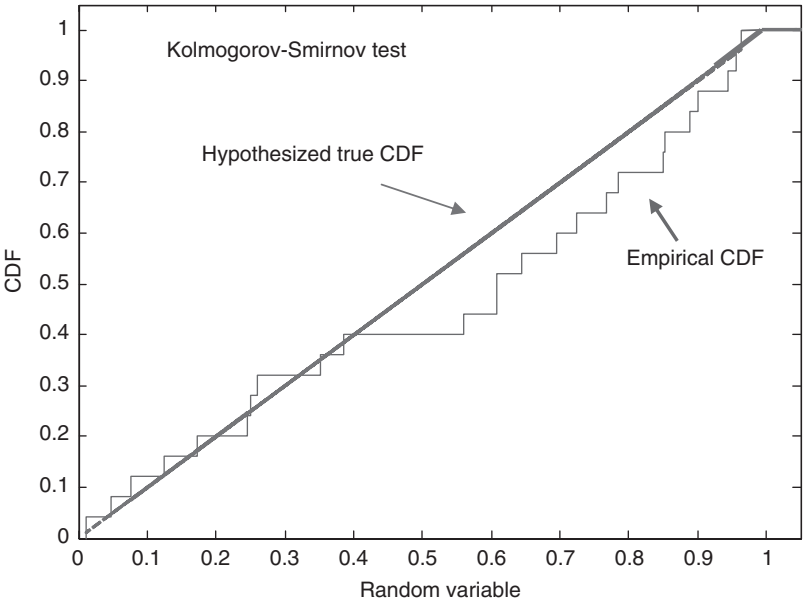


FIGURE 7.11 Kolmogorov-Smirnov model validation test of residual sequence: hypothesized and empirical CDF.

When the transformed residuals are used, then the standard zero-mean/whiteness testing can be accomplished as well as estimating the moments of the Gaussian distribution which we discuss in the next section.

7.6.2.3 Moment-Based Model Validation Test When the residuals are transformed from the hypothesized uniformly distributed sequence to a standard Gaussian, $\nu \sim \mathcal{N}(0, 1)$, then a wealth of normality diagnostics can be applied to validate the adequacy of the embedded model. Besides the zero-mean/whiteness and *WSSR* tests of Sec. 5.6 for Gaussian processes, the usual suite of diagnostics can be applied to estimate the underlying moments to check for consistency of the Gaussian assumption and therefore model validation. The brute force approach is simply to calculate the mean-squared (estimation) error over an ensemble of runs of the processor and any truth model available for comparison,

$$\xi = \sqrt{E\{(\Theta_{\text{true}} - \hat{\Theta})^2\}} \tag{7.90}$$

where Θ can be any parameter, state or measurement estimated by $\hat{\Theta}$ and the expectation can be calculated by integrating or solving over an ensemble generated by executing the processor a multitude of times and averaging. This can be very costly and sometimes impossible because of the lack of the "truth".

Another approach is to calculate a set of statistical indexes that can be used to "qualitatively" assess performance and model validity using the transformed residual

sequence, $\nu(t)$ [51–53]. Here the first four central moments of an N-sample sequence of transformed residuals are estimated based on

$$m_\nu(k) = \frac{1}{N}\sum_{t=1}^{N}(\nu(t) - m_\nu(1))^k \quad \text{for } k \geq 2 \tag{7.91}$$

with the first moment the sample mean

$$m_\nu(1) = \frac{1}{N}\sum_{t=1}^{N}\nu(t) \tag{7.92}$$

These moments can then be used to construct the following diagnostic indices which are asymptotically distributed $\mathcal{N}(0,1)$ (see [51] for details).

- *Bias Index:* $\mathcal{B}_N = \sqrt{N}m_\nu(1)$
- *Dispersion Index:* $\mathcal{D}_N = \frac{Nm_\nu(2) - N + 1}{\sqrt{2(N-1)}}$
- *Skewness Index:* $\mathcal{S}_N = \sqrt{\frac{(N+1)(N+3)}{6(N-2)}} \times \frac{m_\nu(3)}{\sqrt{m_\nu^3(2)}}$
- *Tail Index:* $\mathcal{T}_N = (N+1)\frac{\sqrt{(N+3)(N+5)}}{\sqrt{24N(N-2)(N-3)}} \times \left(\frac{m_\nu(3)}{m_\nu(4)} - \frac{3(N-1)}{(N+1)}\right)$
- *Joint Index:* $\mathcal{J}_N = \mathcal{S}_N^2 + \mathcal{T}_N^2$

From a pragmatic perspective these indices are used in a more qualitative manner even though they are quantitative. They are used to expose "surprising values", that is, for N not too small, the indices can be bound by some constant, β and compared with upper and lower quantile estimates of their exact distribution [51]. For instance, consider the quantile, $\mathcal{D}_{N,\alpha}$, of the exact dispersion index distribution under the assumption of a valid model. Then it can be shown [51] that

$$\mathcal{D}_{N,\alpha} = (\chi^2_{N-1,\alpha} - (N-1)/\sqrt{2(N-1)}) \tag{7.93}$$

where $\chi^2_{N-1,\alpha}$ is $\chi^2(N-1)$ distributed. Other measures such as the skewness and tail indices, S_N and $\mathcal{T}_N$ can be obtained from *MC* simulations [51].

If $\mathcal{B}_N$ is surprisingly high or low, the measurements tend to be larger or smaller then predicted ($\hat{y}(t)$), while a surprisingly high or low dispersion index, $\mathcal{D}_N$ indicates that the measurements are under or over dispersed. The S_N and $\mathcal{T}_N$ are useful in analyzing the measurement distribution. A higher or lower S_N indicates a skew to either right or left while a higher or lower $\mathcal{T}_N$ indicates longer or shorter tails respectively. The $\mathcal{J}_N$ is an asymptotically equivalent to normality tests [50, 54]. A suite of other statistics exist for testing correlations [55, 56] as well.

This completes the section on practical aspects of *PF* design and analysis. We will couple these statistical tests to the classical whiteness testing techniques to evaluate the performance of the processors. Next let us consider the design of a "bootstrap" processor on a canonical problem: a case study for population growth.

7.7 CASE STUDY: POPULATION GROWTH PROBLEM

In this section we discuss the development of state–space particle filters (*SSPF*) for the population growth problem. We consider this well-known problem that has become a *benchmark* for many of the *PF* algorithms. It is highly nonlinear and nonstationary. Thus, consider the problem of [25] and [20, 26–29].

The state transition and corresponding measurement model are given by

$$x(t) = \frac{1}{2}x(t-1) + \frac{25x(t-1)}{1+x^2(t-1)} + 8\cos(1.2(t-1)) + w(t-1)$$
$$y(t) = \frac{x^2(t)}{20} + v(t)$$

where $\triangle t = 1.0$, $w \sim \mathcal{N}(0, 10)$ and $v \sim \mathcal{N}(0, 1)$. The initial state is Gaussian distributed with $x(0) \sim \mathcal{N}(0.1, 5)$.

In terms of the nonlinear state–space representation, we have

$$a[x(t-1)] = \frac{1}{2}x(t-1) + \frac{25x(t-1)}{1+x^2(t-1)}$$
$$b[u(t-1)] = 8\cos(1.2(t-1))$$
$$c[x(t)] = \frac{x^2(t)}{20}$$

In the Bayesian framework, we would like to estimate the instantaneous posterior filtering distribution,

$$\hat{\Pr}(x(t)|Y_t) \approx \sum_{i=1}^{N_p} \mathcal{W}_i \delta(x(t) - x_i(t)) \tag{7.94}$$

where the unnormalized importance weight is given by

$$W_i(t) = \frac{\mathcal{C}\left(y(t)|x_i(t)\right) \times \mathcal{A}\left(x(t)|x_i(t-1)\right)}{q(x_i(t)|X_{t-1}, Y_t)} \tag{7.95}$$

The *weight recursion* for the bootstrap case is $W_i(t) = W_i(t-1) \times \mathcal{C}(y(t)|x(t))$. Therefore, for the Bayesian processor, we have that the state transition probability is given by

$$\mathcal{A}(x(t)|x(t-1)) \sim \mathcal{N}(x(t) : \mathbf{a}[x(t-1)], \mathbf{R}_{ww}) \tag{7.96}$$

Thus, the *SIR* algorithm becomes:

1. Draw samples (particles) from the state transition distribution: $x_i(t) \rightarrow \mathcal{N}(x(t) : \mathbf{a}[x(t-1)], \mathbf{R}_{ww})$

$$w_i(t) \;\rightarrow\; \Pr(w(t)) \sim \mathcal{N}(0, R_{ww})$$

$$x_i(t) = \frac{1}{2}x_i(t-1) + \frac{25x_i(t-1)}{1+x_i^2(t-1)} + 8\cos(1.2(t-1)) + w_i(t-1)$$

2. Estimate the weight/likelihood,

$$W_i(t) \;=\; \mathcal{C}(\mathbf{y}(t)|x(t)) \;\rightarrow\; \mathcal{N}(\mathbf{y}(t) : \mathbf{c}[x(t)], \mathbf{R}_{vv}(t))$$

$$c[x_i(t)] \;=\; \frac{x_i^2(t)}{20}$$

3. Normalize the weight: $\mathcal{W}_i(t) = W_i(t) / \sum_{i=1}^{N_p} W_i(t)$
4. Resample: $\hat{x}_i \Rightarrow x_i$
5. Estimate the instantaneous posterior:

$$\hat{\Pr}(x(t)|Y_t) \approx \sum_{i=1}^{N_p} \mathcal{W}_i \delta(x(t) - x_i(t))$$

6. Estimate (inference) the corresponding statistics:

$$\hat{X}_{\text{MAP}}(t) \;=\; \arg\max_{x(t)} \hat{\Pr}(x(t)|Y_t)$$

$$\hat{X}_{\text{MMSE}}(t) \;=\; E\{x(t)|Y_t\} = \sum_{i=1}^{N_p} x_i(t)\hat{\Pr}(x(t)|Y_t)$$

$$\hat{X}_{\text{MEDIAN}}(t) \;=\; \text{median}(\hat{\Pr}(x(t)|Y_t))$$

We show the simulated data in Fig. 7.12. In *a* we see the hidden state and *b* the noisy measurement. The estimated instantaneous posterior distribution surface for the state is shown in Fig. 7.13a while slices at selected instants of time are shown in b with the circles annotating particle locations normalized to a constant weight. Here we see that the posterior is clearly not unimodal and in fact we can see its evolution in time as suggested by Fig. 7.1 previously. The final state and measurement estimates are shown in Fig. 7.12 demonstrating the effectiveness of the *PF* bootstrap processor for this problem. Various ensemble estimates are shown (e.g., median, *MMSE*, *MAP*). It is clear from the figure that the *EKF* gives a very poor *MMSE* estimate since the posterior is *not* Gaussian (unimodal).

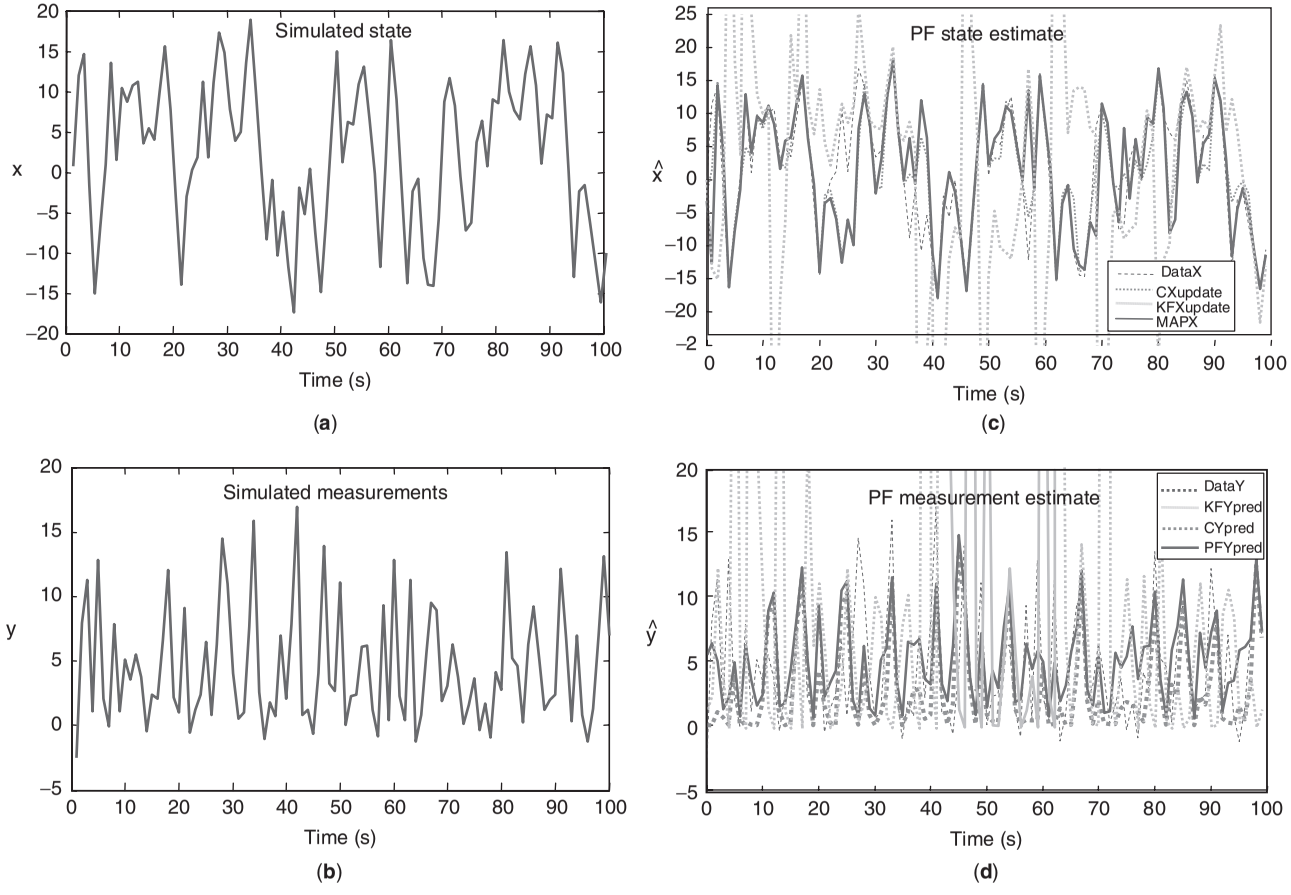


FIGURE 7.12 Nonlinear, nonstationary, non-Gaussian problem: (a) Simulated state with mean. (b) Simulated measurement with mean. (c) Ensemble of state estimates: median, *EKF, MMSE, MAP*. (d) Ensemble of measurement estimates: median, *EKF, MMSE, MAP*.

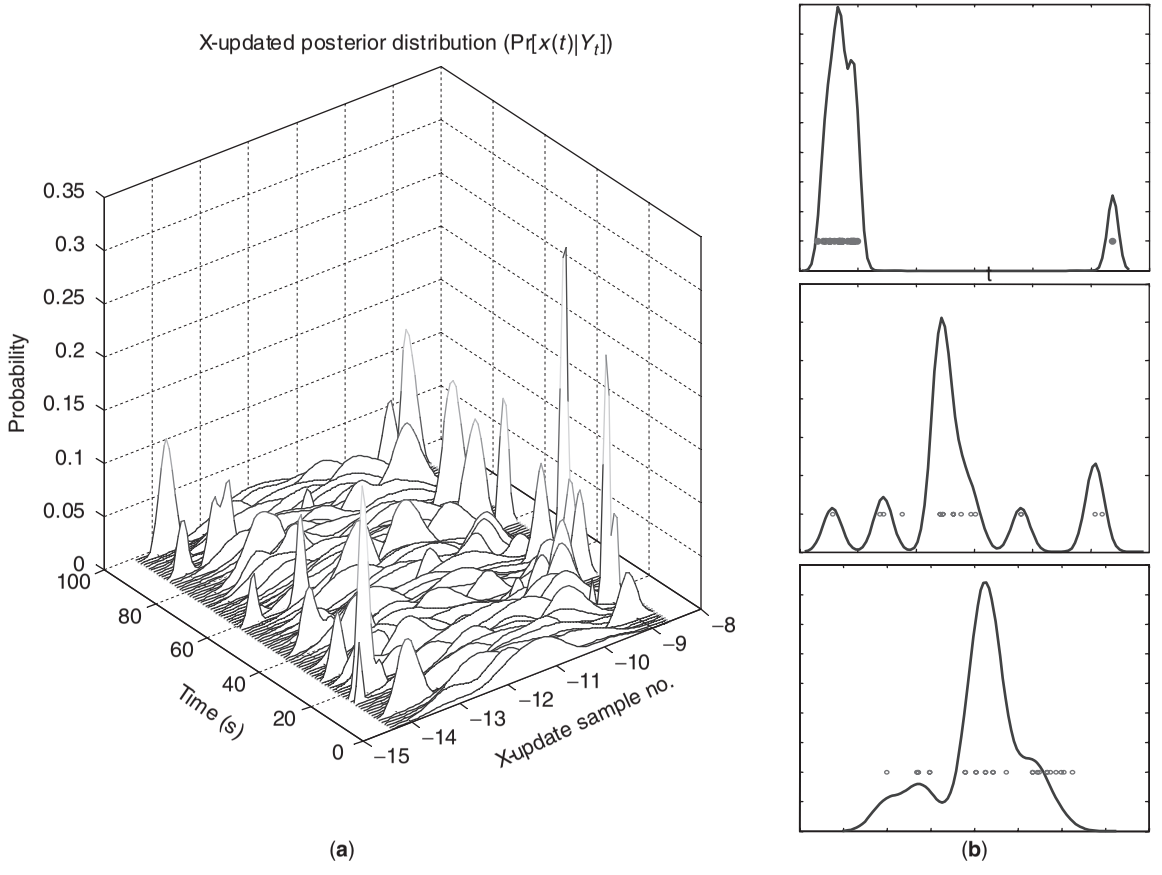


FIGURE 7.13 Nonlinear, nonstationary, non-Gaussian problem: (a) Instantaneous posterior surface. (b) Time slices of the posterior (cross-section) at selected time-steps with particle locations annotated by circles with constant amplitudes.

7.8 SUMMARY

In this chapter we have discussed the development of state–space particle filters (*SSPF*). After introducing the idea of Bayesian particle filters, we showed how the state–space models could easily be interpreted in terms of this framework. We developed a generic state–space particle filtering algorithm based on the importance (sampling) proposals selected either using the minimum variance or transition prior approach. However we emphasized that in practice these techniques suffer from particle depletion and lack of diversity because of ever-increasing weight variances causing divergence of the processors. We introduced the concept of resampling as a solution to the divergence problem and discussed a number of techniques to mitigate the divergence problem. With that in hand, we discussed the popular bootstrap particle filter and showed some examples to demonstrate its performance. We then proceeded to discuss improvements to the bootstrap approach attempting to approximate the minimum variance proposal. These methods included the auxiliary, regularized, *MCMC* and linearized algorithms. Next we investigated some of the practical aspects of particle filter design and developed a number of statistical tests to determine performance including both information theoretic approaches to validate the posterior distribution as well as diagnostic testing for model validation. We concluded the chapter with a case study on population growth—a nonlinear/non-Gaussian model presenting a very challenging problem for any particle filter design. Besides the references in the chapter there has been a wealth of particle filtering papers appearing in both the statistics and signal processing literature [30–43].

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. *MATLAB* has a *Statistics Toolbox* that incorporates a large suite of *PDF*s and *CDF*s as well as "inverse" *CDF* functions ideal for simulation-based algorithms. The **mhsample** command incorporate the Metropolis, Metropolis-Hastings and Metropolis independence samplers in a single command while the Gibbs sampling approach is adequately represented by the more efficient slice sampler (**slicesample**). There are even specific "tools" for sampling as well as the inverse *CDF* method captured in the **randsample** command. *PDF* estimators include the usual histogram (**hist**) as well as the sophisticated kernel density estimator (**ksdensity**) offering a variety of kernel (window) functions (Gaussian, etc.) and ICDF methods including the empirical cumulative distribution (**ecdf**) estimator. As yet no sequential algorithms are available.

In terms of statistical testing for particle filtering diagnostics *MATLAB* offers the chi-square "goodness-of-fit" test **chi2gof** as well as the Kolmogorov-Smirnov distribution test **kstest**. Residuals can be tested for whiteness using the Durbin-Watson test statistic **dwtest** while "normality" is easily checked using the **normplot** command indicating the closeness of the test distribution to a Gaussian. Other statistics are also evaluated using the **mean**, **moment**, **skewness**, **std**, **var** and **kurtosis** commands. Type *help stats* in *MATLAB* to get more details or go to the MathWorks website.

REFERENCES

1. B. Ristic, S. Arulampalam and N. Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Applications* (Boston: Artech House, 2004).
2. M. Arulampalam, S. Maskell, N. Gordon and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Trans. Signal Proc.*, **50**, 2, 174–188, 2002.
3. 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.
4. A. Doucet and X. Wang, "Monte Carlo methods for signal processing," *IEEE Signal Proc. Mag.* **24**, 5, 152–170, 2005.
5. J. Candy, *Model-Based Signal Processing* (Hoboken, NJ: John Wiley/IEEE Press, 2006).
6. O. Cappe, S. Godsill and E. Moulines, "An overview of existing methods and recent advances in sequential Monte Carlo," *Proc. IEEE*, **95**, 5, 899–924, 2007.
7. A. Doucet, N. de Freitas and N. Gordon, *Sequential Monte Carlo Methods in Practice* (New York: Springer-Verlag, 2001).
8. M. Pitt and N. Shephard, "Filtering via simulation: Auxiliary particle filters," *J. Am. Statistical Assoc.*, **94**, 446, 590–599, 1999.
9. R. van der Merwe, A. Doucet, N. de Freitas and E. Wan, "The unscented particle filter," in *Advances in Neural Information Processing Systems* **16** (Cambridge, MA: MIT Press, 2000).
10. R. van der Merwe, *Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State–Space Models*, OGI School of Science & Engr., Oregon Health & Science Univ., PhD Dissertation, 2004.
11. T. Schoen, *Estimation of Nonlinear Dynamic Systems: Theory and Applications* **Linkopings Univ.**, Linkoping, Sweden, Ph.D. Dissertation, 2006.
12. J. Liu, *Monte Carlo Strategies in Scientific Computing* (New York: Springer-Verlag, 2001).
13. J. Ruanaidh and W. Fitzgerald, *Numerical Bayesian Methods Applied to Signal Processing* (New York: Springer-Verlag, 1997).
14. A. Papoulis and S. Pillai, *Probability, Random Variables and Stochastic Processes, 4th Ed*. (New York: McGraw-Hill, 2002).
15. R. Hogg, J. McKlean and A. Craig, *Introduction to Mathematical Statistics, 6th Ed*. (Englewood Cliffs, NJ: Prentice-Hall, 2005).
16. N. Gordon, D. Salmond and A. Smith, "A novel approach to nonlinear non-Gaussian Bayesian state estimation," *IEE Proc. F.*, **140**, 107–113, 1993.
17. D. Simon, *Optimal State Estimation: Kalman H_∞ and Nonlinear Approaches* (Hoboken, NJ: John Wiley/IEEE Press, 2006).
18. P. Stavropoulos and D. Titterington, "Improved particle filters and smoothing," in *Sequential Monte Carlo Methods in Practice* (Editors A. Doucet, N. de Freitas and N. Gordon) (New York: Springer, 2001), pp. 295–317.
19. J. Liu, R. Chen and W. Wong, "Rejection control and sequential importance sampling," *J. Am. Statistical Assoc.*, **96**, 446, 1022–1061, 1998.
20. G. Kitagawa, "Non-gaussian modeling of nonstationary time series," *J. Am. Statistical Assoc.*, **82**, 400, 1032–1073, 1987.
21. A. Jazwinski, *Stochastic Processes and Filtering Theory* (New York: Academic Press, 1970).

22. M. Isard and A. Blake, "Condensation–conditional density propagation for visual tracking," *Int. J. Comput. Vis.*, **29**, 1, 5–28, 1998.

23. M. West and J. Harrison, *Bayesian Forcasting and Dynamic Models*, 2^{nd} *Ed*. (New York: Springer-Verlag, 1997).

24. O. Cappe, E. Moulines and T. Ryden, *Inference in Hidden Markov Models* (New York: Springer-Verlag, 2005).

25. M. Andrade Netto, L. Gimeno and J. Mendes, "On the optimal and suboptimal nonlinear filtering problem for discrete-time systems," *IEEE Trans. Auto. Control*, AC-23, **6**, 1063–1067, 1978.

26. G. Kitagawa, "A nonlinear smoothing method for time series analysis," *Statistica Sinica*, **1**, 2, 371–388, 1991.

27. G. Kitagawa and W. Gersch, *Smoothness Priors Analysis of Time Series* (New York: Springer-Verlag, 1997).

28. G. Kitagawa, "Monte Carlo filter and smoother for non-Gaussian nonlinear state–space models," *J. Comput. Graphical Stat.*, **5**, 1, 1–25, 1997.

29. G. Kitagawa, "Self-organizing state space model," *J. Am. Statistical Assoc.*, **97**, 447, 1207–1215, 1998.

30. H. Tanizaki, *Nonlinear Filters* (New York: Springer-Verlag, 1993).

31. M. Tanner, *Tools for Statistical Inference: Methods for the Exploration of Posterior Distributions and Likelihood Functions*, 2^{nd} *Ed*. (New York: Springer-Verlag, 1993).

32. A. Kong, J. Liu and W. Wong, "Sequential imputations and Bayesian missing data problems," *J. Am. Statistical Assoc.*, **89**, 425, 278–288, 1994.

33. J. Liu and R. Chen, "Blind deconvolution via sequential imputations," *J. Am. Statistical Assoc.*, **90**, 460, 567–576, 1995.

34. J. Liu and R. Chen, "Sequential Monte Carlo methods for dynamic systems," *J. Am. Statistical Assoc.*, **96**, 446, 1062–1044, 1998.

35. A. Smith and A. Gelfand, "Bayesian statistics without tears: a sampling-resampling perspective," *Am. Statistician*, **44**, 4, 84–88, 1992.

36. S. Haykin, *Kalman Filtering and Neural Networks* (Hoboken, NJ: Wiley, 2001).

37. S. Godsill and P. Djuric, "Special Issue: Monte Carlo methods for statistical signal processing." *IEEE Trans. Signal Proc.*, **50**, 173–499, 2002.

38. S. Haykin and N. de Freitas, "Special Issue: Sequential state estimation: from Kalman filters to particle filters." *Proc. IEEE*, **92**, 3, 399–574, 2004.

39. C. Andrieu, A. Doucet, S. Singh and V. Tadic, "Particle methods for change detection, system identification and control," *Proc. IEEE*, **92**, 6, 423–468, 2004.

40. J. Vermaak, C. Andrieu, A. Doucet and S. Godsill, "Particle methods for Bayesian modeling and enhancement of speech signals," *IEEE Trans. Speech Audio Proc.*, **10**, 3, 173–185, 2002.

41. A. Doucet, S. Godsill and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Statist. Comput.*, **10**, 6, 197–208, 2000.

42. A. Harvey, S. Koopman and N. Shephard, *State Space and Unobserved Component Models: Theory and Applications* (Cambridge, UK: Cambridge University Press, 2004).

43. J. Candy, "Bootstrap Particle Filtering," *IEEE Signal Proc. Magz.*, **24**, 4, 73–85, 2007.

44. C. Musso, N. Oudjane and F. LeGland, "Improving regularized particle filters," in *Sequential Monte Carlo Methods in Practice* (Editors A. Doucet, N. de Freitas and N. Gordon) (New York: Springer, 2001), pp. 247–271.
45. N. de Freitas, C. Andrieu, P. Hojen-Sorensen, M. Niranjan and A. Gee, "Sequential Monte Carlo methods for neural networks," in *Sequential Monte Carlo Methods in Practice* (A. Doucet, N. de Freitas and N. Gordon) (New York: Springer, 2001), pp. 359–379.
46. W. Gilks and C. Berzuini, "Following a moving target—Monte Carlo inference for dynamic Bayesian models," *J. Royal Statistical Society B.*, **63**, 127–146, 2001.
47. C. Berzuini and W. Gilks, "Resample-move filtering with cross-model jumps," in *Sequential Monte Carlo Methods in Practice* (A. Doucet, N. de Freitas and N. Gordon) (New York: Springer, pp. 117–138, 2001).
48. M. Rosenblatt, "Remarks on a multivariate transformation," *Ann. Math. Stat.*, **23**, 470–472, 1952.
49. J. Smith, "Diagnostic checks of non-standard time series models," *J. Forcasting*, **4**, 283–291, 1985.
50. A. Harvey, *Forcasting, Structural Time Series Models and the Kalman Filter* (Cambridge, UK: Cambridge Univ. Press, 1989).
51. S. Fruhwirth-Schnatter, "Recursive residuals and model diagnostics for normal and non-normal state–space models," *Environ. and Ecological Stats.*, **3**, 291–309, 1996.
52. S. Fruhwirth-Schnatter, "Bayesian model discrimination and Bayes factors for linear Gaussian state space models," *J. Royal Stat. Soc.* ***B***, 57, 237–246, 1995.
53. S. Fruhwirth-Schnatter, "Applied state space modelling of non-Gaussian time series using integration-based Kalman filtering," *Stats. and Computing*, **4**, 259–269, 1994.
54. K. Bowman and L. Shenton, "Omnibus test countours for departures from normality based on $\sqrt{b_1}$ and b_2," *Biometrika*, **62**, 2, 243–250, 1975.
55. G. Ljung and G. Box, "On a measure of lack of fit in time series models," *Biometrika*, **65**, 2, 297–303, 1978.
56. S. Shapiro and M. Wilk, "An analysis of variance test for normality," *Biometrika*, **52**, 4, 591–611, 1965.
57. R. Gerlach, C. Carter and R. Kohn, "Diagnostics for time series analysis," *J. Time Series Anal.*, **20**, 3, 309–330, 1999.
58. F. Gustafsson, *Adaptive Filtering and Change Detection* (New York: Wiley, 2000).
59. H. Akaike, "A New Look at the Statistical Model Identification," *IEEE Trans. Autom. Control*, **19**, 716–723, 1974.
60. Y. Sakamoto, M. Ishiguro and G. Kitagawa, *Akaike Information Criterion Statistics* (Boston: D. Reidel/Kluwer Academic, 1986).
61. S. Kay, *Modern Spectral Estimation* (Englewood Cliffs, NJ: Prentice-Hall, 1988).
62. S. Haykin, *Adaptive Filtering Theory* (Englewood Cliffs, NJ: Prentice-Hall, 1996).
63. M. Hayes, *Statistical Digital Signal Processing and Modeling* (New York: Wiley, 1996).
64. G. Hendeby, R. Karlsson and F. Gustafsson, "Performance issues in non-Gaussian filtering problems," *Proc. IEEE NSSPW*, Cat. No. 06EX1506, Cambridge, UK, 2006.
65. N. Cui, L. Hong and J. Layne, "A comparison of nonlinear filtering approaches with an application to ground target tracking," *Signal Proc.*, **85**, 1469–1492, 2005.

66. K. Ito and K. Xiong, "Gaussian filters for nonlinear filtering problems," *IEEE Trans. Autom. Control*, **45**, 5, 910–927, 2000.
67. H. Sorenson and D. Alspach, "Recursive Bayesian estimation using Gaussian sums," *Automatica*, **7**, 465–479, 1971.
68. D. Alspach and H. W. Sorenson, "Nonlinear Bayesian estimation using Gaussian sum approximations," *IEEE Trans. Autom. Control*, **17**, 4, 439–448, 1972.
69. I. Arasaratnam, S. Haykin, R. Elliott, "Discrete-time nonlinear filtering algorithms using Gauss-Hermite quadrature," *Proc. IEEE*, **95**, 5, 953–976, 2007.

PROBLEMS

7.1 Given a sequence of Gaussian data (measurements) characterized by $y \sim \mathcal{N}(\mu, \sigma^2)$, find the best estimate of the parameters defined by $\Theta := [\mu \quad \sigma]'$ using a "sequential" *MC* approach. Show the mathematical steps in developing the technique and construct a simple *PF* to solve the problem.

7.2 Consider the following simple model [7]

$$\begin{aligned} x(t) &= ax(t-1) + w(t-1) \quad \text{for } w \sim \mathcal{N}(0, R_{ww}(i)) \text{ with } \mathcal{I}(t) = i \\ y(t) &= x(t) + v(t) \qquad\qquad\quad \text{for } v \sim \mathcal{N}(0, R_{vv}) \end{aligned}$$

with $\Pr(\mathcal{I}(t) = i | \mathcal{I}(t-1), x(t-1)) = \Pr(\mathcal{I}(t) = i) = p_i$

(*a*) Suppose $i = \{1, 2\}$, what is the distribution, $\Pr(\mathcal{I}(t) = (i_1, i_2), x_1 | y_1, x_0)$?

(*b*) How would the marginal be estimated using a Kalman filter?

(*c*) Develop a computational approach to bootstrap *PF* algorithm for this problem.

7.3 Suppose we have two multivariate Gaussian distributions for the parameter vector, $\Theta \sim \mathcal{N}(\mu_i, \Sigma_i); i = 1, 2$

(*a*) Calculate the Kullback-Leibler (*KL*) distance metric, J.

(*b*) Suppose $\Sigma = \Sigma_1 = \Sigma_2$, recalculate the *KL* for this case.

7.4 An aircraft flying over a region can use the terrain and an archival digital map to navigate. Measurements of the terrain elevation are collected in real time while the aircraft altitude over mean sea level is measured by a pressure meter with ground clearance measured by a radar altimeter [7]. The measurement differences are used to estimate the terrain elevations and compared to a digital elevation map to estimate aircraft position. The discrete navigation model is given by

$$\begin{aligned} x(t) &= x(t-1) + u(t-1) + w(t-1) \quad \text{for } w \sim \mathcal{N}(0, R_{ww}) \\ y(t) &= c[x(t)] + v(t) \qquad\qquad\qquad\quad \text{for } v \sim \mathcal{N}(0, R_{vv}) \end{aligned}$$

where x is the $2D$-position, y is the terrain elevation measurement, the navigation systems output u and w are the respective distance traveled and error drift during one time interval. The nonlinear function $c[\cdot]$ denotes the terrain database yielding terrain elevation outputs with v the associated database errors and measurements. Both noises are assumed zero-mean, Gaussian with known statistics, while the initial state is also Gaussian, $x(0) \sim \mathcal{N}(\overline{x}(0), \overline{P}(0))$.

(*a*) Based on this generic description construct the bootstrap *PF* algorithm for this problem.

(*b*) Suppose: $\overline{P}(0) = diag[10^4\ 10^4]'$, $R_{ww} = diag[25\ 25]'$, $R_{vv} = 16$, $N = 150$ samples, $u(t) = [25\ 25]'$. Simulate the aircraft measurements and apply the bootstrap algorithm to estimate the aircraft position.

7.5 In *financial systems*, a stochastic *volatility* model is used in the analysis of financial time series such as daily fluctuations in the stock market prices, exchange rates and option pricing. The volatility is usually expressed in terms of a time-varying variance with the model given by:

$$\begin{aligned} y(t) &= \sigma(t) \times \epsilon(t) \quad \epsilon \sim \mathcal{N}(0,1) \\ \ln \sigma^2(t) &= \alpha + \beta \ln \sigma^2(t-1) + \ln \epsilon^2(t) \end{aligned}$$

or equivalently

$$\begin{aligned} y(t) &= e^{\alpha(t)/2} \times \sigma(t) \times \epsilon(t) \qquad \epsilon \sim \mathcal{N}(0,1) \\ \ln \sigma^2(t) &= \beta(t) \ln \sigma^2(t-1) + v(t) \quad v \sim \mathcal{N}(0, \tau^2) \end{aligned}$$

where $\sigma(t)$ corresponds to the time-varying volatility (amplitude) and the second relation represents the change in volatility. The parameters α and β are regression coefficients, and the remaining parameters are the $\ln \tau^2(t)$ variance term.

(*a*) Suppose we would like to estimate the unknown parameters augmenting the original state ($\ln \sigma^2(t)$) with the unknowns, $\alpha, \beta, \ln \tau^2$. Assume the parameters can be represented by a random walk driven by zero-mean, Gaussian noise processes. What is the overall model for this financial system?

(*b*) Construct a bootstrap *PF* for this problem.

(*c*) Simulate the data for $N = 1000$ samples and estimate the volatility and parameters. The simulation parameters are: $\alpha = 1.8, \beta = 0.95, \tau^2 = 0.1$, and $\epsilon \sim \mathcal{N}(0,1)$.

7.6 Develop a suite of particle filters for the *RC*-circuit problem of Ex. 5.1 where the output voltage was given by:

$$e(t) = \left(1 - \frac{\triangle T}{RC}\right) e(t-1) + \frac{\triangle T}{C} I_{\text{in}}(t-1)$$

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

$$e_{out}(t) = K_e e(t)$$

Recall 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 $x = e$, $y = e_{out}$, and $u = I_{in}$, we obtain

$$\begin{aligned} x(t) &= 0.97x(t-1) + 100u(t-1) + w(t-1) \\ y(t) &= 2x(t) + v(t) \end{aligned}$$

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 $\triangle V = 2\left(\frac{1}{2}\,4\,V\right)$. 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 $u(t) = 300\,\mu A$.

With this in mind, we know that the optimal processor to estimate the state is the linear *BP* (Kalman filter).

(*a*) After performing the simulation using the parameters above, construct a bootstrap *PF* and compare its performance to the optimal. How does it compare? Whiteness? Zero-mean? State estimation error?

(*b*) Let us assume that the circuit is malfunctioning and we do not precisely know the current values of *RC*. Construct a parameter estimator for $A = 1/RC$ using the *EKF* or *UKF* and compare its performance to the bootstrap and linearized *PF*.

(*c*) Try "roughening" the bootstrap *PF*, does its performance improve? Compare results.

7.7 Consider the storage of plutonium nitrate in a tank (see [5] for details), we would like to dynamically estimate the amount (mass) of Pu present at any given time. Losses occur due to radiolysis and evaporation. The underlying state–space model for this system is given by:

Summarizing the process and measurement models in state–space form, we have

$$\frac{d}{dt}\begin{bmatrix} m(t) \\ \rho(t) \end{bmatrix} = \begin{bmatrix} -K_r P & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} m(t) \\ \rho(t) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t) + \begin{bmatrix} w_1(t) \\ w_2(t) \end{bmatrix}$$

where u is a step function of amplitude $-K_H$. The corresponding measurement model based on pressure measurements is

$$\begin{bmatrix} \triangle P_A \\ \triangle P_B \end{bmatrix} = \begin{bmatrix} g/b & -(a/b)g \\ 0 & gH \end{bmatrix}\begin{bmatrix} m(t) \\ \rho(t) \end{bmatrix} + \begin{bmatrix} v_1(t) \\ v_2(t) \end{bmatrix}$$

discretizing the dynamics and incorporating the model parameters, we obtain the Gauss-Markov model with sampling interval of 0.1 *day* as

$$x(t) = \begin{bmatrix} 0.999 & 0 \\ 0 & 1 \end{bmatrix} x(t-1) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t-1) + w(t-1)$$

$$y(t) = \begin{bmatrix} 29.8 & -0.623 \\ 0 & 24.9 \end{bmatrix} x(t) + v(t)$$

$R_{ww} = \text{diag}[10\ 10]$, $R_{vv} = \text{diag}[5.06 \times 10^4\ \ 1.4 \times 10^5]$ with initial conditions $\hat{x}(0|0) = [988\ 1455]'$ and $\tilde{P}(0|0) = \text{diag}[0.01\ 0.01]$.

(*a*) Develop the optimal *BP* for this problem and compare its performance to the bootstrap *PF*.

(*b*) How well does the bootstrap *PF* perform? How about the linearized *PF*?

7.8 We are asked to investigate the possibility of creating a *synthetic aperture* using a single hydrophone to be towed by an AUV in search of targets. We know that a single hydrophone offers no improvement in *SNR* in the sense of array gain, but also wonder about its capability to localize, especially more than one target.

(*a*) Using the synthetic aperture model developed in the case study of Sec. 8.5, develop the bootstrap *PF* and *UKF* processors for this problem. Assume we would like to track two targets.

(*b*) Perform the same simulation outlined in the case study for two targets at $-45°$, $-10°$.

(*c*) Apply the bootstrap algorithm with and without "roughening" along with the *UKF*. Discuss processor performances and compare. Zero-mean? Whiteness?

(*d*) Implement the "optimal" *PF* processor using the *EKF* or *UKF* linearization. How does its performance compare?

7.9 We are asked to investigate the possibility of finding the range of a target using a hydrophone sensor array towed by an AUV assuming a near-field target characterized by it spherical wavefront instead of the far-field target of the previous problem using a plane wave model. The near-field processor can be captured by a wavefront curvature scheme (see [5] for more details) with process and measurement models (assuming that the parameters as well as the measurements are contaminated by additive white Gaussian noise). The following set of dynamic relations can be written succinctly as the *Gauss-Markov wavefront curvature model* as

$$\Theta(t_k) = \Theta(t_{k-1}) + w(t_k) \qquad \text{for } \Theta(t_k) = [\alpha \quad f_o \quad \theta_o \quad r_o]'$$

$$p_\ell(t_k) = \theta_1(t_k) e^{j2\pi\theta_2(t_k)(t_k - \tau_\ell(\Theta;t_k))} + v_\ell(t_k);\ \ell = 1, \ldots, L$$

where α, f_o, θ_o and r_o are the respective amplitude, target frequency, bearing and range. The time delay at the ℓ^{th}-sensor and time t_k is given in terms of the unknown parameters of

$$\tau_\ell(\Theta; t_k) := \frac{1}{c}\left(\theta_4(t_k) - \sqrt{\theta_4^2(t_k) + d_\ell^2(t) - 2d_\ell(t)\theta_4(t_k)\sin\theta_3(t_k)}\right)$$

for $d_\ell(t)$ the distance between the ℓ^{th} sensor and reference range r_o given by

$$d_\ell(t) = x_\ell + v \times t_k \quad \text{for } x_\ell \text{ the position of sensor } \ell$$

(*a*) Using this model develop the bootstrap *PF* and *UKF* processors for this problem. Assume we would like to estimate the target bearing, frequency and range ($\alpha = 1$).

(*b*) Perform a simulation with initial parameters $r_o = 3$ Km, $f_o = 51.1$ Hz and $\theta_o = 27°$ and true parameters at $r = 2$ Km, $f = 51$ Hz and $\theta = 25°$, $L = 4$.

(*c*) Apply the bootstrap algorithm with and without "roughening" along with the *UKF*. Discuss processor performances and compare. Zero-mean? Whiteness?

(*d*) Implement the "optimal" *PF* processor using the *EKF* or *UKF* linearization. How does its performance compare?

7.10 Consider the bearings-only tracking problem of Ex. 5.4 given by the state–space model. The entire system can be represented as an approximate Gauss-Markov model with the noise sources representing uncertainties in the states and measurements. 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 sensor model given by

$$y(t) = \arctan\frac{x_1(t)}{x_2(t)} + v(t)$$

for $w \sim \mathcal{N}(0, R_{ww})$ and $v \sim \mathcal{N}(0, R_{vv})$.

(*a*) Using this model develop the bootstrap *PF* and *UKF* processors for this problem. Assume we would like to estimate the target bearing, frequency and range ($\alpha = 1$).

(*b*) Perform a simulation with the following parameters: an impulse-incremental step change ($\triangle v_{ox} = -24$ knots and $\triangle v_{oy} = +10$ knots) was initiated at 0.5 h, resulting in a change of observer position and velocity depicted in the figure. The simulated bearing measurements are shown in

Fig. 5.6*d*, The initial conditions for the run were $x'(0) := [0\ 15\,\text{nm}\ 20\,\text{k}\ -10\,\text{k}]$ 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 $\triangle T = 0.33\,\text{h}$.

(*c*) Apply the bootstrap algorithm along with the *UKF*. Discuss processor performances and compare. Zero-mean? Whiteness?

(*d*) Implement the "optimal" *PF* processor using the *EKF* or *UKF* linearization. How does its performance compare?

8

JOINT BAYESIAN STATE/PARAMETRIC PROCESSORS

8.1 INTRODUCTION

In this chapter we develop the Bayesian approach to the parameter estimation/system identification problem [1–4] which is based on the decomposition of the joint posterior distributions that incorporates *both* dynamic state and parameter variables. From this formulation the following problems evolve: (1) joint state/parameter estimation; (2) state estimation; and (3) parameter (fixed and/or dynamic) estimation. The state estimation problem is thoroughly discussed in the previous chapters. However, the most common problem found in the current literature is the parameter estimation problem which can be solved "off line" using batch approaches (maximum entropy, maximum likelihood, minimum variance, least squares, etc.) or "on-line" using the expectation-maximization (*EM*) technique (see Chapter 2), the stochastic Monte Carlo approach and for that matter almost any (deterministic) optimization technique [5, 6]. These on-line approaches follow the classical (*EKF*), modern (*UKF*) and the sequential Monte Carlo or particle filter (*PF*). However, it still appears that there is *no* universally accepted approach to solving this problem especially for fixed parameters [7–9]. From the pragmatic perspective, the most useful problem is the *joint* state/parameter estimation problem, since it evolves quite naturally from the fact that a model is developed to solve the basic state estimation problem and it is found that its inherent parameters are either poorly specified, just bounded or even unknown, inhibiting the development of the processor. We call this problem the "joint" state/parameter estimation, since *both* states and parameters are estimated simultaneously on-line and the resulting processor is termed *parametrically adaptive* [18]. This terminology evolves because the inherent model parameters are adjusted sequentially as the measurement data becomes available.

Bayesian Signal Processing. By James V. Candy

In this chapter, we concentrate primarily on the joint Bayesian state/parameter estimation problem and refer the interested reader to the wealth of literature available on this subject [7–19]. First, we precisely define the three basic problems from the Bayesian perspective and then investigate the classical, modern and particle approaches to its solution. We incorporate the nonlinear re-entry problem of Jazwinski [20] used throughout as an example of parametrically adaptive design and then discuss a case study to demonstrate the approach.

8.2 BAYESIAN APPROACH TO JOINT STATE/PARAMETER ESTIMATION

To be more precise, we start by defining the joint state/parametric estimation problem. We begin by formulating the Bayesian recursions in terms of the posterior distribution using Bayes' rule for decomposition, that is,

$$\Pr(x(t), \theta(t)|Y_t) = \Pr(x(t)|\theta(t), Y_t) \times \Pr(\theta(t)|Y_t) = \Pr(\theta(t)|x(t), Y_t) \times \Pr(x(t)|Y_t) \tag{8.1}$$

From this relation, we begin to "see" just how the variety of state and parameter estimation related problems evolve, that is,

- Optimize the joint state/parameter posterior:

$$\Pr(x(t), \theta(t)|Y_t) \qquad \text{[state/parameter estimation]}$$

- Optimize the state posterior:

$$\Pr(x(t)|Y_t) \qquad \text{[state estimation]}$$

- Optimize the parametric posterior:

$$\Pr(\theta(t)|Y_t) \qquad \text{[parameter estimation]}$$

Now if we proceed with the usual factorizations, we obtain the Bayesian decomposition for the *state estimation* problem as

$$\begin{aligned} \Pr(x(t)|Y_t) &= \frac{\Pr(y(t)|x(t)) \times \Pr(x(t)|Y_{t-1})}{\Pr(y(t)|Y_{t-1})} \\ \Pr(x(t)|Y_{t-1}) &= \int \Pr(x(t)|x(t-1)) \times \Pr(x(t-1)|Y_{t-1}) dx(t-1) \end{aligned} \tag{8.2}$$

Equivalently for the *parameter estimation* problem, we have

$$\Pr(\theta(t)|Y_t) = \frac{\Pr(y(t)|\theta(t)) \times \Pr(\theta(t)|Y_{t-1})}{\Pr(y(t)|Y_{t-1})}$$
$$\Pr(\theta(t)|Y_{t-1}) = \int \Pr(\theta(t)|\theta(t-1)) \times \Pr(\theta(t-1)|Y_{t-1}) d\theta(t-1) \quad (8.3)$$

Now for the *joint state/parameter estimation* problem of Eq. 8.1, we can substitute the above equations above to obtain the posterior decomposition of interest, that is,

$$\Pr(x(t), \theta(t)|Y_t) = \frac{\Pr(x(t)|\theta(t), Y_t) \times [\Pr(y(t)|\theta(t)) \times \Pr(\theta(t)|Y_{t-1})]}{\Pr(y(t)|Y_{t-1})} \quad (8.4)$$

or

$$\Pr(x(t), \theta(t)|Y_t) = \Pr(x(t)|\theta(t), Y_t) \times \Pr(y(t)|\theta(t)) \times \frac{\int \Pr(\theta(t)|\theta(t-1)) \times \Pr(\theta(t-1)|Y_{t-1}) d\theta(t-1)}{\Pr(y(t)|Y_{t-1})} \quad (8.5)$$

This is the most common decomposition found in the literature [7–19] and leads to the maximization of the first term with respect to x and the second with respect to θ [22, 23].

Alternatively, using the state/parameter estimation form which is rarely applied, we have

$$\Pr(\theta(t), x(t)|Y_t) = \frac{\Pr(\theta(t)|x(t), Y_t) \times [\Pr(y(t)|x(t) \times \Pr(x(t)|Y_{t-1})]}{\Pr(y(t)|Y_{t-1})} \quad (8.6)$$

or

$$\Pr(\theta(t), x(t)|Y_t) = \Pr(\theta(t)|x(t), Y_t) \times \Pr(y(t)|x(t)) \times \frac{\int \Pr(x(t)|x(t-1)) \times \Pr(x(t-1)|Y_{t-1}) dx(t-1)}{\Pr(y(t)|Y_{t-1})} \quad (8.7)$$

Here the first term is maximized with respect to θ and the second with respect to x compared to the previous decomposition.

So we see that Bayes' rule can be applied in a number of ways to develop the sequential Bayesian processors for the state, parameter and joint state/parameter estimation problems.

8.3 CLASSICAL/MODERN JOINT BAYESIAN STATE/PARAMETRIC PROCESSORS

In this section, we develop the "joint" state–space Bayesian sequential processor or equivalently the parametrically adaptive Bayesian signal processor (*ABSP*) for nonlinear state–space systems. The *ABSP* is a *joint state/parametric 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. The simplified structure of the classical (*EKF*) parameter estimator is shown in Fig. 8.1. We see the basic structure of the *ABSP* which consists of two distinct, yet coupled processors: a parameter estimator and a state estimator. 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 parameter estimates are calculated, a new measurement is processed and the procedure continues. 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 pre-defined (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 [2–4]. Here we are primarily concerned with joint estimation in which the models and parameters are usually nonlinear. Thus,we will concentrate on developing parameter estimators capable of on-line operations with nonlinear dynamics.

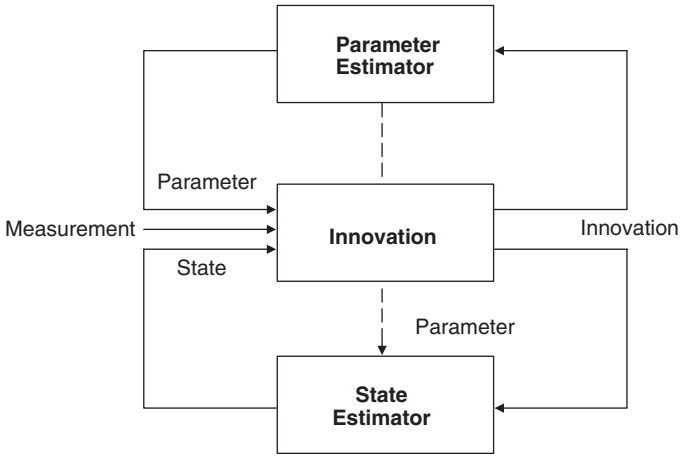


FIGURE 8.1 Nonlinear parametrically adaptive (*ABSP*): simplified processor structure illustrating the coupling between parameter and state estimators through the innovation and measurement sequences.

8.3.1 Classical Joint Bayesian Processor

From our previous discussion in Chapter 5, it is clear that the extended Bayesian processor *XBP* (extended Kalman filter) can satisfy these constraints nicely, so we begin our analysis of the *XBP* as a state/parametric estimator closely following the approach of Ljung [21] for the linear problem discussed in Sec. 8.2. The general nonlinear parameter estimator structure can be derived directly from the *XBP* algorithm in Table 5.3.

Recall the Bayesian solution to the classical problem was based on solving for the posterior distribution (see Eq. 8.2) such that each of the required distributions were represented in terms of the *XBP* estimates. In the joint state/parameter estimation case these distributions map simply by defining an augmented state vector $X(t) := [x(t)|\theta(t)]'$ to give:

$$\begin{aligned}
\Pr\left(y(t)|x(t)\right) &\sim \mathcal{N}(\mathbf{c}[x(t)], R_{vv}(t)) \\
&\Leftrightarrow \\
\Pr\left(y(t)|x(t),\theta(t)\right) &\sim \mathcal{N}(\mathbf{c}[x(t),\theta(t)], R_{vv}(t)) \\
\Pr\left(x(t)|Y_{t-1}\right) &\sim \mathcal{N}(\hat{x}(t|t-1), \widetilde{P}(t|t-1)) \\
&\Leftrightarrow \\
\Pr\left(X(t)|Y_{t-1}\right) &\sim \mathcal{N}(\hat{X}(t|t-1), \tilde{\mathcal{P}}(t|t-1)) \\
\Pr\left(y(t)|Y_{t-1}\right) &\sim \mathcal{N}(\hat{y}(t|t-1), R_{ee}(t)) \\
&\Leftrightarrow \\
\Pr\left(y(t)|Y_{t-1}\right) &\sim \mathcal{N}(\hat{y}_\theta(t|t-1), \tilde{R}_{e_\theta e_\theta}(t))
\end{aligned} \tag{8.8}$$

where

$$\begin{aligned}
\hat{X}(t|t-1) &:= [\hat{x}(t|t-1) \mid \hat{\theta}(t|t-1)]' \\
\hat{y}_\theta(t|t-1) &:= \mathbf{c}[\hat{x}(t|t-1), \hat{\theta}(t|t-1)] \\
\tilde{\mathcal{P}}(t|t-1) &:= \begin{bmatrix} \tilde{P}_{xx}(t|t-1) & | & \tilde{P}_{x\theta}(t|t-1) \\ - & - & - \\ \tilde{P}_{\theta x}(t|t-1) & | & \tilde{P}_{\theta\theta}(t|t-1) \end{bmatrix} \\
\mathcal{R}_{e_\theta e_\theta}(t) &:= C[\hat{X}(t|t-1)]\tilde{\mathcal{P}}(t|t-1)C[\hat{X}(t|t-1)]' + R_{vv}(t)
\end{aligned}$$

To develop the actual internal structure of the *ABSP*, we start with the *XBP* equations, augment them with the unknown parameters and then investigate the resulting algorithm. We first define the composite state-vector (as before) consisting of the original states, $x(t)$, and the unknown "augmented" parameters represented by $\theta(t)$, that is,

$$X(t) := \begin{bmatrix} x(t) \\ - - - \\ \theta(t) \end{bmatrix} \tag{8.9}$$

where t is the time index, $X \in R^{(N_x+N_\theta)\times 1}$ and $x \in R^{N_x\times 1}$, $\theta \in R^{N_\theta\times 1}$. Substituting this augmented state vector into the *XBP* relations of Table 5.3, the following matrix partitions evolve as

$$\tilde{\mathcal{P}}(t|t-1) := \begin{bmatrix} \tilde{P}_{xx}(t|t-1) & | & \tilde{P}_{x\theta}(t|t-1) \\ - & - & - \\ \tilde{P}_{\theta x}(t|t-1) & | & \tilde{P}_{\theta\theta}(t|t-1) \end{bmatrix}, \quad \tilde{P}_{\theta x}(t|t-1) = \tilde{P}'_{x\theta}(t|t-1) \tag{8.10}$$

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 corresponding gain

$$\mathcal{K}(t) := \begin{bmatrix} K_x(t) \\ - - - \\ K_\theta(t) \end{bmatrix} \tag{8.11}$$

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,[1]

$$\begin{aligned} \hat{X}(t|t-1) &= \begin{bmatrix} \hat{x}(t|t-1) \\ - - - \\ \hat{\theta}(t|t-1) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{a}[\hat{x}(t-1|t-1), \hat{\theta}(t-1|t-1)] \\ \quad + \mathbf{b}[\hat{x}(t-1|t-1), \hat{\theta}(t-1|t-1), u(t-1)] \\ - - - \\ \hat{\theta}(t-1|t-1) \end{bmatrix} \end{aligned} \tag{8.12}$$

where the corresponding predicted measurement equation becomes

$$\hat{y}(t|t-1) = \mathbf{c}[\hat{x}(t|t-1), \hat{\theta}(t|t-1)] \tag{8.13}$$

Next we consider the predicted error covariance

$$\tilde{P}(t|t-1) = A[\hat{x}(t|t-1), \hat{\theta}(t|t-1)]\tilde{P}(t|t-1)A'[\hat{x}(t|t-1), \hat{\theta}(t|t-1)] + R_{ww}(t-1) \tag{8.14}$$

[1] 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 Jacobian etc. will change from this representation.

which can be written in partitioned form using Eq. 8.10 and the following Jacobian process matrix[2]

$$A[\hat{x},\hat{\theta}] = \begin{bmatrix} A_x[\hat{x}(t|t-1),\hat{\theta}(t|t-1)] & | & A_\theta[\hat{x}(t|t-1),\hat{\theta}(t|t-1)] \\ - & - & - \\ O & | & I_{N_\theta} \end{bmatrix} \tag{8.15}$$

where

$$\begin{aligned} A_x[\hat{x},\hat{\theta}] &:= \frac{\partial a[\hat{x},\hat{\theta}]}{\partial x} + \frac{\partial b[\hat{x},\hat{\theta}]}{\partial x} \\ A_\theta[\hat{x},\hat{\theta}] &:= \frac{\partial a[\hat{x},\hat{\theta}]}{\partial \theta} + \frac{\partial b[\hat{x},\hat{\theta}]}{\partial \theta} \end{aligned} \tag{8.16}$$

with $A \in R^{(N_x+N_\theta)\times(N_x+N_\theta)}, A_x \in R^{N_x\times N_x}, A_\theta \in R^{N_x\times N_\theta}$.

Using these partitions, we can develop the *ABSP* directly from the *XBP* processor in this joint state and "parameter estimation" form. Substituting Eq. 8.15 into the *XBP* Prediction Covariance relation of Table 5.3 and using the partition defined in Eq. 8.10, we obtain (suppressing the $\hat{x}$, $\hat{\theta}$, time index t notation for simplicity)

$$\begin{aligned} &\tilde{\mathcal{P}}(t|t-1) \\ &= \begin{bmatrix} A_x & | & A_\theta \\ - & - & - \\ 0 & | & I_{N_\theta} \end{bmatrix} \tilde{\mathcal{P}}(t-1|t-1) \begin{bmatrix} A_x & | & A_\theta \\ - & - & - \\ 0 & | & I_{N_\theta} \end{bmatrix}' + \begin{bmatrix} R_{w_x w_x} & | & 0 \\ - & - & - \\ 0 & | & R_{w_\theta w_\theta} \end{bmatrix} \end{aligned} \tag{8.17}$$

Expanding these equations, we obtain the following set of predicted covariance relations

$$\begin{aligned} &\tilde{\mathcal{P}}(t|t-1) \\ &= \begin{bmatrix} A_x\tilde{P}_{xx}A_x' + A_\theta\tilde{P}_{\theta x}A_x' + A_x\tilde{P}_{x\theta}A_\theta' + A_\theta\tilde{P}_{\theta\theta}A_\theta' + R_{w_x w_x} & | & A_x\tilde{P}_{x\theta} + A_\theta\tilde{P}_{\theta\theta} \\ - - - & - & - - - \\ \tilde{P}_{\theta x}A_x' + \tilde{P}_{\theta\theta}A_\theta' & | & \tilde{P}_{\theta\theta} + R_{w_\theta w_\theta} \end{bmatrix} \end{aligned} \tag{8.18}$$

The innovations covariance follows from the *XBP* as

$$R_{ee}(t) = C[\hat{x},\hat{\theta}]\tilde{\mathcal{P}}(t|t-1)C'[\hat{x},\hat{\theta}] + R_{vv}(t) \tag{8.19}$$

Now we must use the partitions of $\tilde{\mathcal{P}}$ above along with the measurement Jacobian

$$C[\hat{x},\hat{\theta}] = [C_x[\hat{x}(t|t-1),\hat{\theta}(t|t-1)]|C_\theta[\hat{x}(t|t-1),\hat{\theta}(t|t-1)]] \tag{8.20}$$

[2] Here is where the underlying random walk model enters the structure. The lower block rows of A_x could be replaced by $[A_{\theta x}[\hat{x},\hat{\theta}]|A_{\theta\theta}[\hat{x},\hat{\theta}]]$ which enables a linear or nonlinear dynamic model to be embedded directly.

where

$$C_x[\hat{x},\hat{\theta}] := \frac{\partial c[\hat{x},\hat{\theta}]}{\partial x}$$
$$C_\theta[\hat{x},\hat{\theta}] := \frac{\partial c[\hat{x},\hat{\theta}]}{\partial \theta} \tag{8.21}$$

with $C \in R^{N_y \times (N_x+N_\theta)}$, $C_x \in R^{N_y \times N_x}$, $C_\theta \in R^{N_y \times N_\theta}$.

The corresponding innovations covariance follows from Eqs. 8.19 and 8.20 as

$$R_{ee}(t) = [C_x|C_\theta] \begin{bmatrix} \tilde{P}_{xx} & | & \tilde{P}_{x\theta} \\ - & - & - \\ \tilde{P}_{\theta x} & | & \tilde{P}_{\theta\theta} \end{bmatrix} \begin{bmatrix} C_x' \\ - - - \\ C_\theta' \end{bmatrix} + R_{vv}(t) \tag{8.22}$$

or expanding

$$R_{ee}(t) = C_x\tilde{P}_{xx}C_x' + C_\theta\tilde{P}_{\theta x}C_x' + C_x\tilde{P}_{x\theta}C_\theta' + C_\theta\tilde{P}_{\theta\theta}C_\theta' + R_{vv} \tag{8.23}$$

$R_{ee} \in R^{N_y \times N_y}$. The gain of the *XBP* in Table 5.3 is calculated from these partitioned expressions as

$$\mathcal{K}(t) = \begin{bmatrix} K_x(t) \\ - - - \\ K_\theta(t) \end{bmatrix} = \begin{bmatrix} \tilde{P}_{xx} & | & \tilde{P}_{x\theta} \\ - & - & - \\ \tilde{P}_{\theta x} & | & \tilde{P}_{\theta\theta} \end{bmatrix} \begin{bmatrix} C_x \\ - - - \\ C_\theta \end{bmatrix}' R_{ee}^{-1}(t) \tag{8.24}$$

or

$$\mathcal{K}(t) = \begin{bmatrix} K_x(t) \\ - - - \\ K_\theta(t) \end{bmatrix} = \begin{bmatrix} (\tilde{P}_{xx}C_x' + \tilde{P}_{x\theta}C_\theta')R_{ee}^{-1}(t) \\ - - - \\ (\tilde{P}_{\theta x}C_x' + \tilde{P}_{\theta\theta}C_\theta')R_{ee}^{-1}(t) \end{bmatrix} \tag{8.25}$$

where $K \in {}^{(N_x+N_\theta)\times N_y}$, $K_x \in R^{N_x \times N_y}$, $K_\theta \in R^{N_\theta \times N_y}$. With the gain determined, the corrected state/parameter estimates follow easily, since the innovations remain unchanged, that is,

$$e(t) = y(t) - \hat{y}(t|t-1) = y(t) - c[\hat{x}(t|t-1), \hat{\theta}(t|t-1)] \tag{8.26}$$

and therefore partitioning the corrected state equations, we have

$$\hat{X}(t|t) = \begin{bmatrix} \hat{x}(t|t) \\ - - - \\ \hat{\theta}(t|t) \end{bmatrix} = \begin{bmatrix} \hat{x}(t|t-1) \\ - - - \\ \hat{\theta}(t|t-1) \end{bmatrix} + \begin{bmatrix} K_x(t)e(t) \\ - - - \\ K_\theta(t)e(t) \end{bmatrix} \tag{8.27}$$

Finally, the corrected covariance expression is easily derived from the following partitions

$$\tilde{\mathcal{P}}(t|t) = \begin{bmatrix} \tilde{P}_{xx} & | & \tilde{P}_{x\theta} \\ - & - & - \\ \tilde{P}_{\theta x} & | & \tilde{P}_{\theta\theta} \end{bmatrix} - \begin{bmatrix} K_x(t) \\ - - - \\ K_\theta(t) \end{bmatrix} [C_x | C_\theta] \begin{bmatrix} \tilde{P}_{xx} & | & \tilde{P}_{x\theta} \\ - & - & - \\ \tilde{P}_{\theta x} & | & \tilde{P}_{\theta\theta} \end{bmatrix} \tag{8.28}$$

Performing the indicated multiplications leads to the final expression

$$\tilde{\mathcal{P}}(t|t) = \begin{bmatrix} \tilde{P}_{xx} - K_x C_x \tilde{P}_{xx} - K_x C_\theta \tilde{P}_{\theta x} & | & \tilde{P}_{x\theta} - K_x C_x \tilde{P}_{x\theta} - K_x C_\theta \tilde{P}_{\theta\theta} \\ - - - & - - - & - - - \\ \tilde{P}_{\theta x} - K_\theta C_x \tilde{P}_{xx} - K_\theta C_\theta \tilde{P}_{\theta x} & | & \tilde{P}_{\theta\theta} - K_\theta C_x \tilde{P}_{x\theta} - K_\theta C_\theta \tilde{P}_{\theta\theta} \end{bmatrix} \tag{8.29}$$

We summarize the parametrically adaptive model-based processor in predictor-corrector form in Table 8.1. We note that this algorithm is *not* implemented in this fashion, it is implemented in the numerically stable, "upper triangular-diagonal" or UD-factorized form as in *SSPACK_PC* [18]. Here we are just interested in the overall internal structure of the algorithm and the decomposition that evolves. This completes the development of the generic *ABSP*.

It is important to realize that besides its numerical implementation the *ABSP* is simply the *XBP* 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 *ABSP* evolves naturally from the standard *XBP* algorithm of Table 5.3. The *ABSP* of Table 8.1 indicates where to locate the partitions. That is, suppose we would like to extract the submatrix, $\tilde{P}_{\theta\theta}$, but the *XBP* 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 5 and investigate the performance of the parametrically adaptive Bayesian processor.

Example 8.1

Recall the discrete nonlinear trajectory estimation problem [20] of Chapter 5 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 $v(t) \sim \mathcal{N}(0, 0.09)$, $x(0) = 2.0$, $P(0) = 0.01$, $\triangle T = 0.01$ sec and $R_{ww} = 0$.

TABLE 8.1 *ABSP* Algorithm

Predictor/Corrector Form

PREDICTION

$$\hat{x}(t|t-1) = a[\hat{x}(t|t-1), \hat{\theta}(t-1|t-1)] + b[u(t-1)] \quad (\textit{state})$$

$$\hat{\theta}(t|t-1) = \hat{\theta}(t-1|t-1) \quad (\textit{parameter})$$

$$\begin{aligned}\tilde{P}_{xx}(t|t-1) &= A_x[\hat{x},\hat{\theta}]\tilde{P}_{xx}(t-1|t-1)A'_x[\hat{x},\hat{\theta}] + A_\theta[\hat{x},\hat{\theta},u]\tilde{P}_{\theta x}(t-1|t-1)A'_x[\hat{x},\hat{\theta}] \\ &\quad + A_x[\hat{x},\hat{\theta}]\tilde{P}_{x\theta}(t-1|t-1)A'_\theta[\hat{x},\hat{\theta},u] + A_\theta[\hat{x},\hat{\theta}]\tilde{P}_{\theta\theta}(t-1|t-1)A'_\theta[\hat{x},\hat{\theta},u] + R_{w_x w_x}(t-1)\end{aligned} \quad (\textit{state cov.})$$

$$\tilde{P}_{\theta\theta}(t|t-1) = \tilde{P}_{\theta\theta}(t-1|t-1) + R_{w_\theta w_\theta}(t-1) \quad (\textit{param. cov.})$$

$$\tilde{P}_{x\theta}(t|t-1) = A_x[\hat{x},\hat{\theta}]\tilde{P}_{x\theta}(t-1|t-1) + A_\theta[\hat{x},\hat{\theta},u]\tilde{P}_{\theta\theta}(t-1|t-1) \quad (\textit{cross cov.})$$

INNOVATION

$$e(t) = y(t) - \hat{y}(t|t-1) = y(t) - c[\hat{x}(t|t-1), \hat{\theta}(t|t-1)] \quad (\textit{innovation})$$

$$\begin{aligned}R_{ee}(t) &= C_x[\hat{x},\hat{\theta}]\tilde{P}_{xx}(t|t-1)C'_x[\hat{x},\hat{\theta}] + C_\theta[\hat{x},\hat{\theta},u]\tilde{P}_{\theta x}(t|t-1)C'_x[\hat{x},\hat{\theta}] \\ &\quad + C_x[\hat{x},\hat{\theta}]\tilde{P}_{x\theta}(t|t-1)C'_\theta[\hat{x},\hat{\theta},u] + C_\theta[\hat{x},\hat{\theta},u]\tilde{P}_{\theta\theta}(t|t-1)C'_\theta[\hat{x},\hat{\theta},u] + R_{vv}(t)\end{aligned} \quad (\textit{innovation covariance})$$

GAIN

$$K_x(t) = (\tilde{P}_{xx}(t|t-1)C'_x[\hat{x},\hat{\theta}] + \tilde{P}_{x\theta}(t|t-1)C'_\theta[\hat{x},\hat{\theta},u])R^{-1}_{ee}(t) \quad (\textit{state gain})$$

$$K_\theta(t) = (\tilde{P}_{\theta x}(t|t-1)C'_x[\hat{x},\hat{\theta}] + \tilde{P}_{\theta\theta}(t|t-1)C'_\theta[\hat{x},\hat{\theta},u])R^{-1}_{ee}(t) \quad (\textit{parameter gain})$$

CORRECTION

$$\hat{x}(t|t) = \hat{x}(t|t-1) + K_x(t)e(t) \quad (\textit{state})$$

$$\hat{\theta}(t|t) = \hat{\theta}(t|t-1) + K_\theta(t)e(t) \quad (\textit{parameter})$$

$$\tilde{P}_{xx}(t|t) = \tilde{P}_{xx}(t|t-1) - K_x(t)C_x[\hat{x},\hat{\theta}]\tilde{P}_{xx}(t|t-1) - K_x(t)C_\theta[\hat{x},\hat{\theta},u]\tilde{P}_{\theta x}(t|t-1) \quad (\textit{state covariance})$$

$$\tilde{P}_{\theta\theta}(t|t) = \tilde{P}_{\theta\theta}(t|t-1) - K_\theta(t)C_\theta[\hat{x},\hat{\theta},u]\tilde{P}_{x\theta}(t|t-1) - K_\theta(t)C_\theta[\hat{x},\hat{\theta},u]\tilde{P}_{\theta\theta}(t|t-1) \quad (\textit{parameter covariance})$$

$$\tilde{P}_{x\theta}(t|t) = \tilde{P}_{x\theta}(t|t-1) - K_x(t)C_x[\hat{x},\hat{\theta}]\tilde{P}_{x\theta}(t|t-1) - K_x(t)C_\theta[\hat{x},\hat{\theta},u]\tilde{P}_{\theta\theta}(t|t-1) \quad (\textit{cross covariance})$$

INITIAL CONDITIONS

$$\hat{x}(0|0) \quad \tilde{P}(0|0) \quad A[\hat{x},\theta] := \frac{\partial}{\partial x}A[x,\theta]\Big|_{\substack{x=\hat{x}(t|t-1)\\ \theta=\hat{\theta}(t|t-1)}} \quad C[\hat{x},\theta] := \frac{\partial}{\partial x}C[x,\theta]\Big|_{\substack{x=\hat{x}(t|t-1)\\ \theta=\hat{\theta}(t|t-1)}}$$

Here we generalize the problem to the case where the coefficients of the process are unknown leading to the *ABSP* 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: $\Theta_{true} = [0.05\ 0.04]'$. The *ABSP* can be applied to this problem by defining the parameter vector as

$$\Theta(t) = \Theta(t-1) \quad \text{(constant)}$$

and augmenting it to form the new state vector $X = [x'\ \ \theta_1\ \ \theta_2]'$. Therefore the process model becomes

$$\begin{aligned} X(t) &= \begin{bmatrix} (1 - \theta_1(t-1)\triangle T)x(t-1) + \theta_2(t-1)\triangle T x^2(t-1) \\ \theta_1(t-1) \\ \theta_2(t-1) \end{bmatrix} + w(t-1) \\ y(t) &= x^2(t) + x^3(t) + v(t) \end{aligned}$$

To implement the *ABSP* the required Jacobians are

$$\begin{aligned} &A[X(t-1)] \\ &\quad = \begin{bmatrix} [1 - \theta_1(t-1)\triangle T + 2\triangle T\theta_2(t-1)x(t-1)] & \triangle T)x(t-1) & \triangle T x^2(t-1) \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &C[X(t-1)] = [2x(t-1) + 3x^2(t-1) \quad 0 \quad 0] \end{aligned}$$

Using *SSPACK_PC* [18] the *ABSP* is applied to solve this problem for 1500 samples with $\triangle T = 0.01$ sec. Initially, we used the starting parameters:

$$\tilde{P}(0|0) = \text{diag}[100\ \ 100\ \ 100] \quad \text{and} \quad \hat{x}(0|0) = [2\ \ 0.055\ \ 0.044]'$$

The results of the *ABSP* run are shown in Fig. 8.2. We see the estimated state and parameter estimates in *b* and *c*. After a short transient (25 samples), the state estimate begins tracking the true state as evidenced by the innovations sequence in Fig. 8.2*c*. The parameter estimates slowly converge to their true values as evidenced by the plots. The final estimates are

$$\begin{aligned} \hat{\theta}_1 &= 0.0470 \\ \hat{\theta}_2 &= 0.0395 \end{aligned}$$

Part of the problem for the slow convergence results stems from the lack of sensitivity of the measurement, or equivalently, innovations to parameter variations

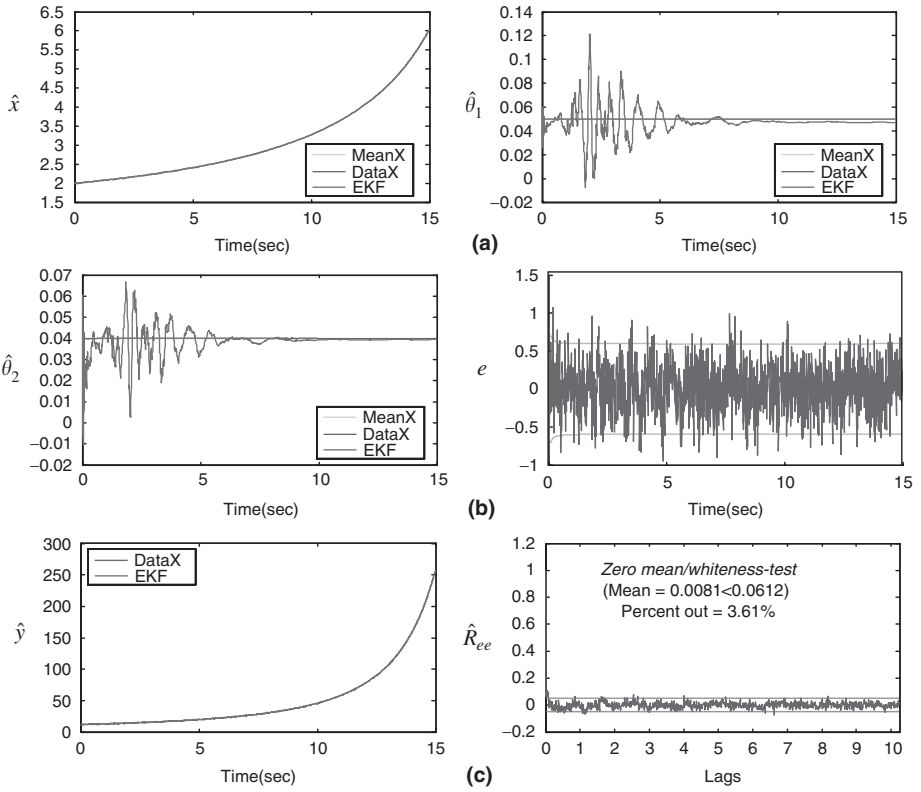


FIGURE 8.2 *XBP (EKF)* simulation. (a) Estimated state and parameter no. 1. (b) Estimated parameter no. 2 and innovation. (c) Predicted measurement and zero-mean/whiteness test ($0.008 < 0.061$ and 3.6% out).

in this problem. This is implied from the zero-mean/whiteness tests shown in *c*. The innovations are statistically white (3.6% out), and zero-mean ($0.008 < 0.061$). The filtered measurement is also shown in *c* as well. This completes the *ABSP* example. △△△

As pointed out by Ljung [1, 2, 21], it is important to realize that the *XBP* is suboptimal as a parameter estimator as compared to the recursive prediction error (*RPE*) method based on the Gauss-Newton (stochastic) descent algorithm. Comparing the processors in this context, we see that if a gradient term $[\nabla_\theta K(\Theta)]\, e(t)$ is incorporated into the *XBP* (add this term to A_θ), its convergence will be improved approaching the performance of the *RPE* algorithm (see Ljung [21] for details). We also note in passing that the nonlinear *BSP* in the form developed in Chapter 5 as well as the parametrically adaptive *ABSP* are all heavily employed as *neural networks*. For more details of this important application see Haykin [17]. Next we consider the development of the "modern" approach to Bayesian processor design using the unscented Bayesian processor of Chapter 6.

8.3.2 Modern Joint Bayesian Processor

The modern unscented processor offers a similar representation as the extended processor detailed in the previous subsection. Here we briefly outline its structure for solution to the *joint* problem and apply it to the trajectory estimation problem for comparison. We again start with the augmented state vector defined initially by sigma-points, that is,

$$\mathcal{X}(t) := \begin{bmatrix} x(t) \\ --- \\ \theta(t) \end{bmatrix} \tag{8.30}$$

$\mathcal{X} \in R^{(N_x+N_\theta)\times 1}$ and $x \in R^{N_x \times 1}$, $\theta \in R^{N_\theta \times 1}$. Substituting this augmented state vector into the *SPBP* relations of Table 6.1 yields the desired processor. We again draw the equivalences (as before):

$$\begin{aligned} \Pr\left(y(t)|x(t),\theta(t)\right) &\sim \mathcal{N}(\mathbf{c}[x(t),\theta(t)], R_{vv}(t)) \\ \Pr\left(\mathcal{X}(t)|Y_{t-1}\right) &\sim \mathcal{N}(\mathcal{X}(t|t-1), \tilde{\mathcal{P}}(t|t-1)) \\ \Pr\left(y(t)|Y_{t-1}\right) &\sim \mathcal{N}(\hat{y}_\theta(t|t-1), \tilde{R}_{e_\theta e_\theta}(t)) \end{aligned} \tag{8.31}$$

where

$$\begin{aligned} \mathcal{X}(t|t-1) &:= [\hat{x}(t|t-1) \mid \hat{\theta}(t|t-1)]' \\ \hat{y}_\theta(t|t-1) &:= \mathbf{c}[\hat{x}(t|t-1), \hat{\theta}(t|t-1)] \\ \tilde{\mathcal{P}}(t|t-1) &:= \begin{bmatrix} \tilde{P}_{xx}(t|t-1) & | & \tilde{P}_{x\theta}(t|t-1) \\ - & - & - \\ \tilde{P}_{\theta x}(t|t-1) & | & \tilde{P}_{\theta\theta}(t|t-1) \end{bmatrix} \\ \mathcal{R}_{e_\theta e_\theta}(t) &:= C[\hat{X}(t|t-1)]\tilde{\mathcal{P}}(t|t-1)C[\hat{X}(t|t-1)]' + R_{vv}(t) \end{aligned}$$

With this in mind it is possible to derive the internal structure of the *SPBP* in a manner similar to that of the *XBP*. But we will not pursue that derivation here. We just note that the sigma-points are also augmented to give

$$\mathcal{X}_i := \begin{bmatrix} \hat{x}(t|t-1) \\ --- \\ \hat{\theta}(t|t-1) \end{bmatrix} + \left((N_{\mathcal{X}} + \kappa) \begin{bmatrix} \tilde{P}_{xx}(t|t-1) & | & \tilde{P}_{x\theta}(t|t-1) \\ - & - & - \\ \tilde{P}_{\theta x}(t|t-1) & | & \tilde{P}_{\theta\theta}(t|t-1) \end{bmatrix} \right)^{\frac{1}{2}} \tag{8.32}$$

with the corresponding process noise covariance partitioned as

$$\mathcal{R}_{ww}(t-1) := \begin{bmatrix} R_{w_x w_x}(t-1) & | & \mathbf{0} \\ - & - & - \\ \mathbf{0} & | & R_{w_\theta w_\theta}(t-1) \end{bmatrix} \tag{8.33}$$

It also follows that the prediction-step becomes

$$\mathcal{X}_i(t|t-1) = \begin{bmatrix} \mathbf{a}[\hat{x}(t|t-1), \hat{\theta}(t|t-1)] + \mathbf{b}[\hat{\theta}(t|t-1), u(t-1)] \\ --- \\ \mathbf{a}[\hat{\theta}(t|t-1)] \end{bmatrix} \tag{8.34}$$

and in the multichannel (vector) measurement case we have that

$$\mathcal{Y}_i(t|t-1) = \begin{bmatrix} \mathbf{c}[\hat{x}(t|t-1), \hat{\theta}(t|t-1)] \\ --- \\ \mathbf{c}[\hat{\theta}(t|t-1)] \end{bmatrix} \tag{8.35}$$

Using the augmented state vector, we apply the "joint" approach to the trajectory estimation problem [20] and compare its performance to that of the *XBP*.

Example 8.2

Using the discrete nonlinear trajectory estimation problem of the previous example with unknown coefficients as before, we define the augmented sigma-point vector $\mathcal{X}(t)$ defined above and apply the *SPBP* algorithm of Table 6.1.

The process equations for this problem are:

$$x(t) = (1 - \theta_1 \triangle T)x(t-1) + \theta_2 x^2(t-1) + w(t-1)$$

with the identical measurement and covariances as before. The true parameters are: $\Theta_{true} = [0.05 \;\; 0.04]'$. The *SPBP* can be applied to this problem by defining the parameter vector as a constant and augmenting it to form the new sigma-point vector $\mathcal{X} = [x' \quad \theta_1 \quad \theta_2]'$. Therefore the process model becomes

$$\begin{bmatrix} x(t) \\ -- \\ \theta(t) \end{bmatrix} = \begin{bmatrix} (1 - \theta_1(t-1)\triangle T)x(t-1) + \theta_2(t-1)\triangle T x^2(t-1) \\ \theta_1(t-1) \\ \theta_2(t-1) \end{bmatrix} + w(t-1)$$

$$y(t) = x^2(t) + x^3(t) + v(t)$$

To implement the *SPBP* the sigma-points are selected as before for a Gaussian distribution using the scaled transformation with $\alpha = 1$, $\kappa = 0$ and $\beta = 2$ using the same initial conditions as the *XBP*.

Using *MATLAB* [18], the *ABSP* is applied to solve this problem for 1500 samples with $\triangle T = 0.01$ sec. The results of the *SPBP* run are shown in Fig. 8.3. We see the estimated state and parameter estimates in *a* and *b*. After a short transient, the state estimate begins tracking the true state as evidenced by the predicted measurement and innovations sequence in Fig. 8.3*c*. The parameter estimates converge to their true values as evidenced by the plots. The final estimates are: $\hat{\theta}_1 = 0.05$; $\hat{\theta}_2 = 0.04$. The processor appears to converge much faster than the *XBP* demonstrating its improved capability. This is implied from the zero-mean/whiteness tests shown in *c*.

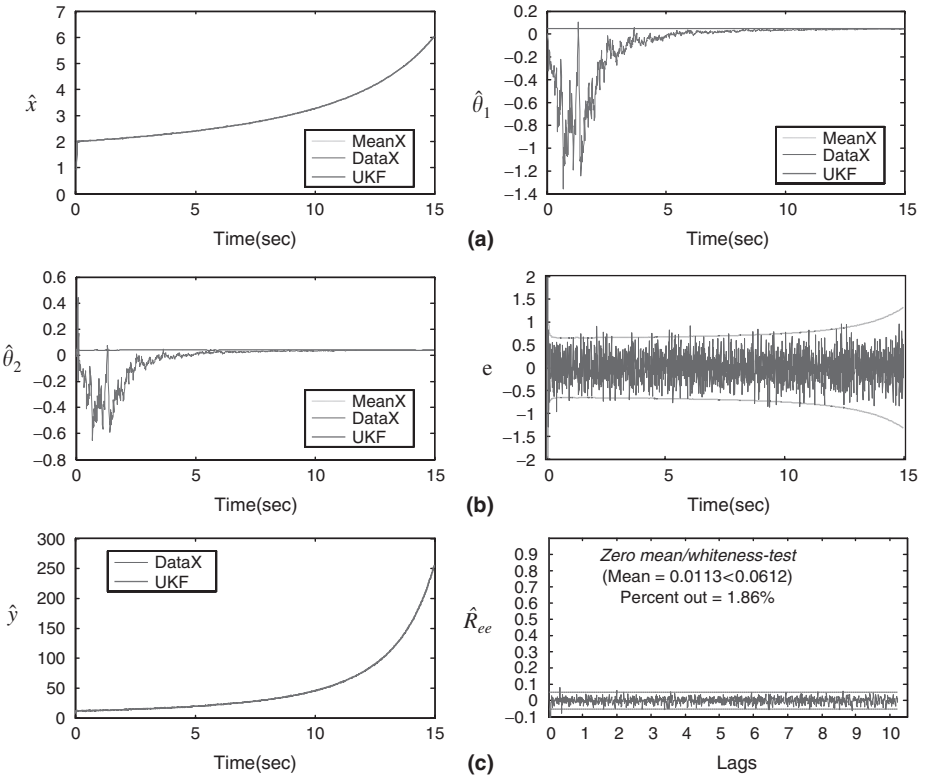


FIGURE 8.3 *SPBP* (*UKF*) simulation. (a) Estimated state and parameter no. 1. (b) Estimated parameter no. 2 and innovation. (c) Predicted measurement and zero-mean/whiteness test (0.0113 < 0.0612 and 1.86% out).

The innovations are statistically white (1.86% out) and zero-mean ($0.0113 < 0.0612$). The filtered measurement is also shown in c as well. This completes the *ABSP* example. △△△

We also note in closing that a "dual" rather than "joint" approach has evolved in the literature. Originally developed as a bootstrap approach, it is constructed by two individual (decoupled) processors: one for the state estimator and one for the parameter estimator which pass updated estimates back and forth to one another as they become available. This is a suboptimal methodology, but appears to perform quite well (see [22, 23] for more details). This completes our discussion of the joint modern approach, next we investigate the *SMC* approach to solving the joint problem.

8.4 PARTICLE-BASED JOINT BAYESIAN STATE/PARAMETRIC PROCESSORS

In this section we briefly develop the sequential Monte Carlo approach to solving the joint state/parametric processing problem. It is not surprising that the resulting

particle filtering technique does not perform very well especially for a non-dynamic or *static* parameter. In fact, after the initial time, if left alone, the *PF* will just assign the initial weight as unity and proceed to use the initial parameter estimate for the entire trajectory. As discussed previously for the state estimation problem, this occurs because the parameter has no mechanism to explore the associated parameter space for the optimal solution.[3] The most obvious solution to this particular problem is to artificially assign a random walk (pseudo dynamic) model with small variance to approximate small variations in the parameter forcing it to vary, that is,

$$\theta(t) = \theta(t-1) + w_\theta(t-1) \quad \text{for } w_\theta \sim \mathcal{N}(0, R_{w_\theta w_\theta}) \tag{8.36}$$

The process noise variance bounds the excursions of the random walk or equivalently the parameter space. Using artificial dynamics is the identical approach used for both the classical (*XBP*) and modern (*SPBP*) techniques used in the previous section. The only problem results when the parameter is truly *static* such as in financial models and its variations can have very large repercussions in the money and economic markets. Thus, a large variety of "off-line" *MC* methods have evolved [16], but we will not discuss them here since we are primarily interested in physical systems which typically have parametric uncertainties that are well modeled by the random walk or other variations. Of course, if the parametric relations are truly dynamic, then the joint approach incorporates parameter estimation and yields an optimal filtering solution to this joint problem.

Here we are concerned with the joint estimation problem consisting of setting a prior for θ and augmenting the state vector to solve the joint estimation problem as defined in Sec. 8.2 thereby converting the parameter estimation problem to one of optimal filtering. Thus, confining our discussion to state–space models and the Bayesian equivalence we develop the Bayesian approach using the relations:

$$\begin{aligned} x &\sim \mathcal{A}_x\left(x(t)|x(t-1),\theta(t-1)\right) \\ \theta &\sim \mathcal{A}_\theta\left(\theta(t)|\theta(t-1),x(t-1)\right) \\ y &\sim \mathcal{C}\left(y(t)|x(t),\theta(t)\right) \end{aligned} \tag{8.37}$$

Here we separate the state transition function into the individual vectors for illustrative purposes, but in reality (as we shall observe), they can be jointly coupled. The key idea is to develop the *PF* technique to estimate the joint posterior $\Pr\left(x(t),\theta(t)|Y_t\right)$ relying on the parametric posterior $\Pr\left(\theta(t)|Y_t\right)$ in the Bayesian decomposition. We will follow the approach outlined in [16, 33] starting with the full posterior distributions and proceeding to the filtering distributions.

Suppose it is possible to sample N_p-particles, $\{X_t(i),\Theta_t(i)\}$ for $i=1,\ldots,N_p$ from the joint posterior distribution where we define, $X_t := \{x(0),\ldots,x(t)\}$ and $\Theta_t := \{\theta(0),\ldots,\theta(t)\}$. Then the corresponding empirical approximation of the joint posterior is given by

[3] The idea of applying a particle filter to a problem that does not have much or any process noise is not practical, it is better to use other methods for this type of problem [6].

$$\hat{\Pr}(X_t, \Theta_t|Y_t) \approx \frac{1}{N_p} \sum_{i=1}^{N_p} \delta(X_t - X_t(i), \Theta_t - \Theta_t(i)) \tag{8.38}$$

and it follows directly that the filtering posterior (see Chapter 2) is given by

$$\hat{\Pr}(x(t), \theta(t)|Y_t) \approx \frac{1}{N_p} \sum_{i=1}^{N_p} \delta(x(t) - x_i(t), \theta(t) - \theta_i(t)) \tag{8.39}$$

Unfortunately it is not possible to sample directly from the full joint posterior $\Pr(X_t, \Theta_t|Y_t)$ at any time t. However one approach to mitigate this problem is by using the importance sampling approach of Chapter 2.

Suppose we define a (full) importance distribution, $q(X_t, \Theta_t|Y_t)$ such that $\Pr(X_t, \Theta_t|Y_t) > 0$ implies $q(\cdot) > 0$, then we define the corresponding importance weight (as before) by

$$W(X_t, \Theta_t) \propto \frac{\Pr(X_t, \Theta_t|Y_t)}{q(X_t, \Theta_t|Y_t)} \tag{8.40}$$

From Bayes' rule we have that the posterior can be expressed as

$$\Pr(X_t, \Theta_t|Y_t) = \frac{\Pr(Y_t|X_t, \Theta_t) \times \Pr(X_t, \Theta_t)}{\Pr(Y_t)} \tag{8.41}$$

Thus, if N_p-particles, $\{X_t(i), \Theta_t(i)\}$; $i = 1, \ldots, N_p$, can be generated from the importance distribution

$$\{X_t(i), \Theta_t(i)\} \rightarrow q(X_t, \Theta_t|Y_t) \tag{8.42}$$

then the empirical distribution can be estimated and the resulting normalized weights specified by

$$\mathcal{W}_t(i) = \frac{W(X_t(i), \Theta_t(i))}{\sum_{k=1}^{N_p} W(X_t(k), \Theta_t(k))} \quad \text{for } i = 1, \ldots, N_p \tag{8.43}$$

to give the desired empirical distribution of Eq. 8.38 leading to the corresponding filtering distribution of Eq. 8.39.

If we have a state transition model available, then for a fixed parameter estimate the state estimation problem is easily solved as before in Chapter 7. Therefore, we will confine our discussion to the parameter posterior distribution estimation problem, that is, marginalizing the joint distribution with respect to the states (that have already been estimated) gives

$$\Pr(\Theta_t|Y_t) = \int \Pr(X_t, \Theta_t|Y_y) dX_t \tag{8.44}$$

and it follows that

$$\mathcal{W}(\Theta_t) \propto \frac{\Pr(\Theta_t|Y_t)}{q(\Theta_t|Y_t)} \tag{8.45}$$

Assuming that a set of particles can be generated from the importance distribution as $\Theta_t \sim q(\Theta_t|Y_t)$, then we have the set of normalized weights

$$\mathcal{W}_t(\Theta(i)) = \frac{W(\Theta_t(i))}{\sum_{k=1}^{N_p} W(\Theta_t(k))} \quad \text{for } i = 1, \ldots, N_p \tag{8.46}$$

which is the joint "batch" importance sampling solution when coupled with the dynamic state vectors.

The sequential form of this joint formulation follows directly (as before in Chapter 2). We start with the factored form of the importance distribution and focus on the full posterior $\Pr(\Theta_t|Y_t)$, that is,

$$q(\Theta_t|Y_t) = \prod_{k=0}^{t} q(\theta(k)|\Theta_{k-1}, Y_k) \tag{8.47}$$

with $\Pr(\theta(0)|\Theta_{-1}, Y_t) \rightarrow \Pr(\theta(0)|Y_t)$.

Assuming that this factorization can be expressed recursively in terms of the previous step, $q(\Theta_{t-1}|Y_{t-1})$ and extracting the t-th term gives

$$q(\Theta_t|Y_t) = q(\theta(t)|\Theta_{t-1}, Y_t) \times \prod_{k=0}^{t-1} q(\theta(k)|\Theta_{k-1}, Y_k) \tag{8.48}$$

or simply

$$q(\Theta_t|Y_t) = q(\theta(t)|\Theta_{t-1}, Y_t) \times q(\Theta_{t-1}|Y_{t-1}) \tag{8.49}$$

With this in mind the weight recursion becomes

$$W(\Theta_t) = W(t) \times W(\Theta_{t-1}) \tag{8.50}$$

Applying Bayes' rule to the posterior, we define

$$\begin{aligned} W(t) &:= \frac{\Pr(y(t)|\Theta_t, Y_{t-1}) \times \Pr(\theta(t)|\Theta(t-1))}{\Pr(y(t)|Y_{t-1}) \times q(\theta(t)|\Theta_{t-1}, Y_t)} \\ &\propto \frac{\Pr(y(t)|\Theta_t, Y_{t-1}) \times \Pr(\theta(t)|\theta(t-1))}{q(\theta(t)|\Theta_{t-1}, Y_t)} \end{aligned} \tag{8.51}$$

As before in the state estimation problem we must choose an importance distribution before we can construct the algorithm. We can choose from

the *minimum variance* (optimal) using local linearization techniques given by $q(\theta(t)|\Theta_{t-1}, Y_t) \rightarrow \Pr(\theta(t)|\Theta_{t-1}, Y_t)$ which leads to the transition posterior

$$\Pr(\theta(t)|\Theta_{t-1}, Y_t) = \frac{\Pr(y(t)|\Theta_{t-1}, Y_{t-1}) \times \Pr(\theta(t)|\theta(t-1))}{\Pr(y(t)|\Theta_{t-1}, Y_{t-1})} \tag{8.52}$$

and therefore it follows using Eq. 8.51 that

$$W_{MV}(t) \propto \Pr(y(t)|\Theta_{t-1}, Y_{t-1}) = \int \Pr(y(t)|\Theta_t, Y_{t-1}) \times \Pr(\theta(t)|\theta(t-1))\, d\theta(t) \tag{8.53}$$

with (local linearization implementation of Chapter 7)

$$\Pr(y(t)|\Theta_t, Y_{t-1}) \sim \mathcal{N}(\hat{y}_\theta(t|t-1), R_{\epsilon_\theta \epsilon_\theta})$$

which can be obtained from any of the classical or modern techniques (Chapters 5 and 6).

The usual bootstrap approach can also be selected as the importance distribution leading to a simpler alternative with $q(\theta(t)|\Theta_{t-1}, Y_t) \rightarrow \Pr(\theta(t)|\theta(t-1))$ and the weight of Eq. 8.51 becomes

$$W_{BS}(t) = \Pr(y(t)|\Theta_t, Y_{t-1}) \tag{8.54}$$

From the pragmatic perspective, we must consider some practical approaches to implementing the processor for the joint problem. The first approach, when applicable, (not financial problems) is to incorporate the random walk model of Eq. 8.36 when reasonable [16]. We will use this approach for our case study to follow. Another variant is to use the "roughening" method that moves the particles (after resampling) by adding a Gaussian sequence of specified variance.[4]

The *kernel method* (regularized *PF* of Chapter 7) can also be applied to the joint problem. In the bootstrap implementation we can estimate the posterior distribution using the kernel density approach, that is, after resampling we have the empirical distribution given by

$$\hat{\Pr}(\theta(t)|Y_t) = \frac{1}{N_p} \sum_{i=1}^{N_p} \delta(\theta(t) - \hat{\theta}_i(t)) \tag{8.55}$$

The kernel technique consists of substituting for this degenerate distribution, the kernel density estimate

$$\hat{\Pr}(\theta(t)|Y_t) = \frac{1}{N_p} \sum_{i=1}^{N_p} \mathcal{K}(\theta(t) - \hat{\theta}_i(t)) \tag{8.56}$$

[4] Recall that the sequence is distributed $\epsilon \sim \mathcal{N}(0, \kappa \mathcal{M}_{ij} N_p^{-1/N_x})$ for κ a constant and $\mathcal{M}_{ij}$ the maximum distance between the i^{th} and j^{th} particles discussed previously of Sec. 7.5.

for $\mathcal{K}(\cdot)$ the kernel (e.g., Gaussian, triangle, etc.). Now a new set of parametric particles can be obtained by generating samples from $\theta_i(t) \sim \mathcal{K}(\theta(t))$. In the same manner as the standard bootstrap, this approach introduces diversity into the set of particles.

Yet another alternative is to introduce the *MCMC*-step (see chapter 7) to "move" the particles to the regions of high probability using a Markov chain with appropriate invariant distribution. Again the new particles are sampled according to a *MCMC* with joint distribution (when possible) $\Pr(X_t(i), \Theta_t(i)|Y_t)$ such that

$$(X_t(i), \Theta_t(i)) \sim \mathcal{K}_{MCMC}(X_t, \Theta_t|Y_t)$$

This completes the discussion of the joint state/parameter estimation problem using the *PF* approach, we emphasize that the algorithm of Chapter 7 may be used by merely augmenting the state vector with the parameter vector especially when a dynamic equation characterizing the parameter dynamics is available. Next we consider an example to illustrate this approach.

Example 8.3

Again we consider the trajectory estimation problem [20] using the particle filter technique. At first we applied the usual bootstrap technique and found what was expected, a collapse of the parameter particles giving an unsatisfactory result. Next we tried the "roughening" approach and the results were much more reasonable. We used a roughening factor or $\kappa = 5 \times 10^{-5}$ along with $N_p = 350$. The results are shown below in Fig. 8.4. We see both the estimated states and measurement along with the associated zero-mean/whiteness test. The result, although not as good as the modern approach, is reasonable with the final estimates converging to the static parameter values of true parameters: $\theta_1 = 0.05$ (0.034) and $\theta_2 = 0.04$ (0.039). The state and measurement estimates are quite good as evidenced by the zero-mean ($0.002 < 0.061$) and whiteness (1.76% out). We show the estimated posterior distributions for the states and parameters in Fig. 8.5 again demonstrating a reasonable solution. Note how the distributions are initially multi-modal and become unimodal as the parameter estimates converge to their true values as depicted in Fig. 8.5. △△△

8.5 CASE STUDY: RANDOM TARGET TRACKING USING A SYNTHETIC APERTURE TOWED ARRAY

Synthetic aperture processing is well-known in airborne radar, but not as familiar in sonar [35–40]. The underlying idea in creating a synthetic aperture is to increase the array length by motion, thereby, increasing spatial resolution (bearing) and gain in *SNR*. It has been shown that for stationary targets the motion induced bearing estimates have smaller variance than that of a stationary array [38, 41]. Here we investigate the case of *both* array and target motion. We define the acoustic *array space-time processing problem* as:

GIVEN a set of noisy pressure-field measurements from a horizontally towed array of L-sensors in motion, **FIND** the "best" (minimum error variance) estimate of the target bearings.

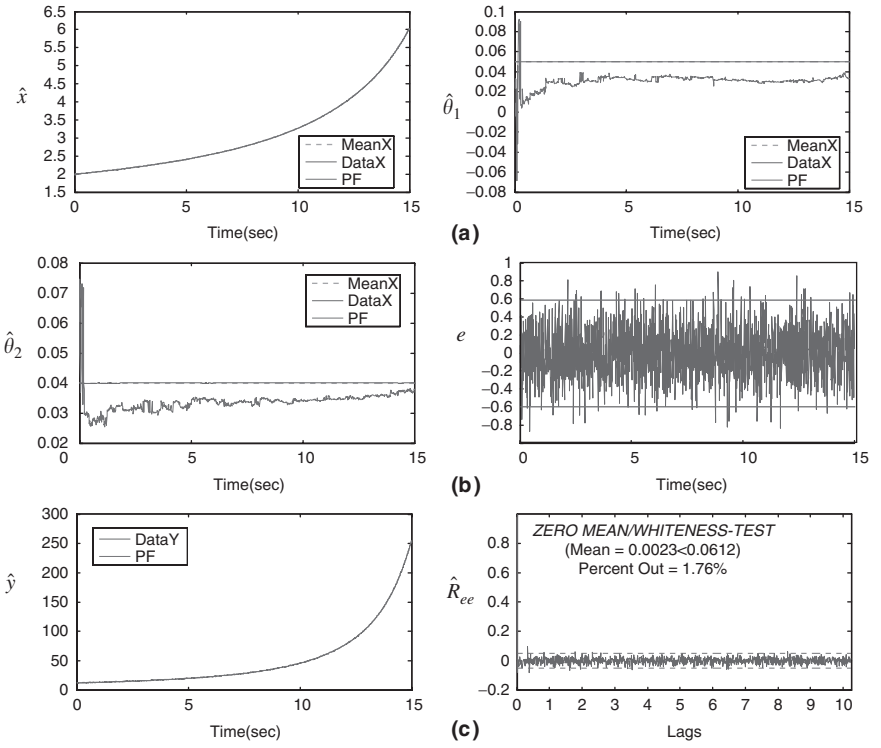


FIGURE 8.4 *PF* simulation. (a) Estimated state and parameter no. 1. (b) Estimated parameter no. 2 and innovation. (c) Predicted measurement and zero-mean/whiteness test (0.002 < 0.061 and 1.76% out).

We use the following nonlinear pressure-field measurement model for M monochromatic plane wave targets characterized by a corresponding set of temporal frequencies, bearings, and amplitudes, $[\{\omega_m\}, \{\theta_m\}, \{a_m\}]$. That is,

$$p(x, t_k) = \sum_{m=1}^{M} a_m e^{j\omega_m t_k - j\beta(x,t_k)\sin\theta_m} + n(t_k) \tag{8.57}$$

where $\beta(x, t_k) := k_o x(t_o) + vt_k$, $k_o = \frac{2\pi}{\lambda_o}$ is the wavenumber, $x(t_k)$ is the current spatial position along the x-axis in meters, v is the tow speed in m/sec, and $n(t_k)$ is the additive random noise. The inclusion of motion in the *generalized* wave number, β, 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.

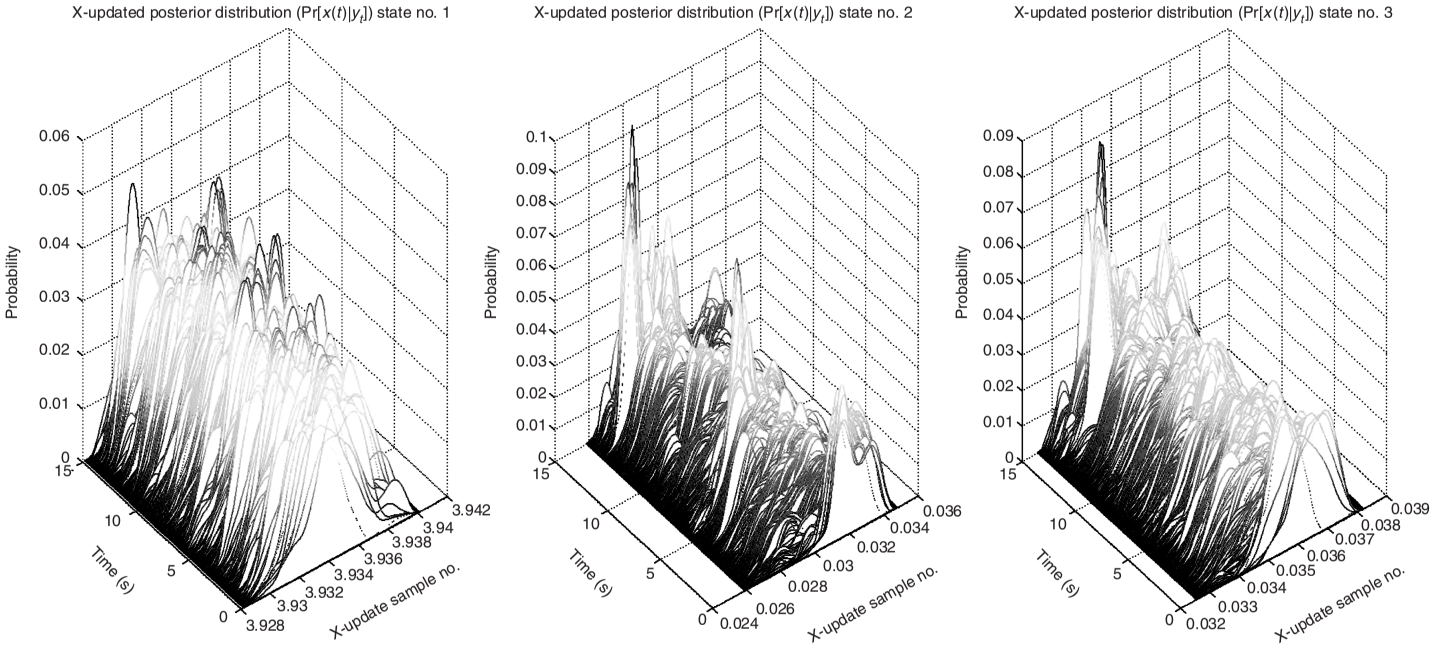


FIGURE 8.5 *PF* posterior distribution estimation. (a) Estimated state posterior. (b) Parameter no. 1 posterior. (c) Parameter no. 2 posterior.

If we further assume that the single sensor equation above is expanded to include an array of L-sensors, $x \to x_\ell$, $\ell = 1, \ldots, L$; then we obtain

$$\begin{aligned} p(x_\ell, t_k) &= \sum_{m=1}^{M} a_m e^{j\omega_m t_k - j\beta(x_\ell, t_k)\sin\theta_m} + n_\ell(t_k) \\ &\to \sum_{m=1}^{M} a_m \cos(\omega_m t_k - \beta(x_\ell, t_k)\sin\theta_m) + n_\ell(t_k) \end{aligned} \tag{8.58}$$

since our hydrophone sensors measure the real part of the complex pressure-field, the final nonlinear measurement model of the system can be written in compact vector form as

$$\mathbf{p}(t_k) = \mathbf{c}[t_k; \boldsymbol{\Theta}] + \mathbf{n}(t_k) \tag{8.59}$$

where $\mathbf{p}, \mathbf{c}, \mathbf{n} \in C^{L\times 1}$, are the respective pressure-field, measurement and noise vectors and $\boldsymbol{\Theta} \in \mathcal{R}^{M\times 1}$ represents the target bearings. The corresponding vector measurement model

$$\mathbf{c}_\ell(t_k; \boldsymbol{\Theta}) = \sum_{m=1}^{M} a_m \cos(\omega_m t_k - \beta(x_\ell, t_k)\sin\theta_m) \quad \text{for } \ell = 1, \ldots, L$$

Since we model the bearings as a random walk emulating random target motion, then the Markovian state–space model evolves from first differences as

$$\boldsymbol{\Theta}(t_k) = \boldsymbol{\Theta}(t_{k-1}) + \mathbf{w}_\theta(t_{k-1}) \tag{8.60}$$

Thus, the state–space model is linear with no explicit dynamics, therefore, the process matrix $A = I$ (identity) and the relations are greatly simplified.

Now let us see how a particle filter using the bootstrap approach can be constructed according to the generic algorithm of Table 7.2. For this problem, we assume the additive noise sources are Gaussian, so we can compare results to the performance of the approximate processor. We define the discrete notation, $t_{k+1} \to t + 1$ for the sampled-data representation.

Let us cast this problem into the sequential Bayesian framework, that is, we would like to estimate the *instantaneous* posterior filtering distribution, $\hat{\Pr}(x(t)|Y_t)$, using the *PF* representation to be able to perform inferences and extract the target bearing estimates. Therefore, we have that the transition probability is given by $(\Theta(t) \to x(t))$

$$\Pr(\theta(t)|\theta(t-1)) \longrightarrow \mathcal{A}(\theta(t)|\theta(t-1)) \sim \mathcal{N}(\Theta(t) : \mathbf{a}[\Theta(t-1)], \mathbf{R}_{w_\theta, w_\theta})$$

or in terms of our state transition (bearings) model, we have

$$\boldsymbol{\Theta}(t) = \mathbf{a}[\Theta(t-1)] + w_\theta(t-1) = \boldsymbol{\Theta}(t-1) + \mathbf{w}_\theta(t-1) \quad \text{for } \Pr(\mathbf{w}_\theta(t)) \sim \mathcal{N}(0, \mathbf{R}_{w_\theta w_\theta})$$

The corresponding likelihood is specified in terms of the measurement model $(y(t) \rightarrow p(t))$ as

$$\Pr(y(t)|x(t)) \longrightarrow \mathcal{C}(y(t)|x(t)) \sim \mathcal{N}(\mathbf{y}(t) : \mathbf{c}[\Theta(t)], \mathbf{R}_{vv}(t))$$

where we have used the notation: $z \sim \mathcal{N}(z : m_z, R_{zz})$ to specify the Gaussian distribution in random vector z. In terms of our problem, we have that

$$\ln \mathcal{C}(y(t)|x(t)) = \kappa - \frac{1}{2}(\mathbf{y}(t) - \sum_{m=1}^{M} a_m \cos(\omega_m t - \beta(t) \sin\theta_m))' \mathbf{R}_{vv}^{-1} \times (\mathbf{y}(t) - \sum_{m=1}^{M} a_m \cos(\omega_m t - \beta(t) \sin\theta_m))$$

with κ a constant, $\beta \in \mathcal{R}^{L\times 1}$ and $\beta(t) := [\beta(x_1, t)| \ldots |\beta(x_L, t)]'$, the dynamic wavenumber expanded over the array. Thus, the *SIR* algorithm becomes:

- Draw samples (particles) from the state transition distribution:

$$\Theta_i(t) \sim \mathcal{N}(\Theta(t)\text{: } \mathbf{a}[\Theta(t-1)], \mathbf{R}_{w_\theta, w_\theta})$$

$$w_{\theta_i}(t) \sim \Pr(w(t)) \sim \mathcal{N}(0, R_{w_{\theta_i} w_{\theta_i}}), \Theta_i(t) = \Theta_i(t-1) + w_{\theta_i}(t-1)$$

- Estimate the likelihood, $\mathcal{C}(\mathbf{y}(t)|\Theta(t)) \sim \mathcal{N}(\mathbf{y}(t)\text{: } \mathbf{c}[\Theta(t)], \mathbf{R}_{vv}(t))$ with $c_\ell(t; \Theta_i) = \sum_{m=1}^{M} a_m \cos(\omega_m t_k - \beta(x_\ell, t) \sin\Theta_{m,i}(t))$ for $\ell = 1, \ldots, L$ and $\Theta_{m,i}$ is the i^{th}-particle at the m^{th}-bearing angle;
- Update and normalize the weight: $\mathcal{W}_i(t) = W_i(t) / \sum_{i=1}^{N_p} W_i(t)$
- Resample: $\hat{N}_{eff}(t) \leq N_{thresh}$
- Estimate the instantaneous posterior: $\hat{\Pr}(\Theta(t)|Y_t) \approx \sum_{i=1}^{N_p} \mathcal{W}_i(t) \delta(\Theta(t) - \Theta_i(t))$
- Perform the inference by estimating the corresponding statistics: $\hat{\Theta}_{\text{map}}(t) = \arg\max \hat{\Pr}(\Theta(t)|Y_t)$; $\hat{\Theta}_{\text{mmse}}(t) = E\{\Theta(t)|Y_t\} = \sum_{i=1}^{N_p} \Theta_i(t) \hat{\Pr}(\Theta(t)|Y_t)$; $\hat{\Theta}_{\text{median}}(t) = \text{median}(\hat{\Pr}(\Theta(t)|Y_t))$.

Consider the following simulation of the synthetic aperture using a 4-element, linear towed array with "moving" targets using the following parameters:

Target: Unity amplitudes with temporal frequency is 50 Hz, *Wavelength* = 30 m, *Tow Speed* = 5 m/sec; *Array*: four (4) element linear towed array with 15 m spacing; *Particle filter*: $N_\theta = 4$ states (bearings), $N_y = 4$ sensors, $N = 250$ samples, $N_p = 250$ weights; *SNR*: is -10 dB; *Noise*: white Gaussian with: $R_{ww} = \text{diag}\,[2.5]$, $R_{vv} = \text{diag}\,[0.1414]$; *Sampling interval*: is 0.005 sec; *Initial Conditions*: (bearings and uncertainty) $\Theta_o = [45^\circ\ -10^\circ\ 5^\circ\ -75^\circ]'$, $P_o = \text{diag}\,(10^{-10})$.

The array simulation was executed and the targets moved according to a random walk specified by the process noise and sensor array measurements with -10 dB *SNR*. The results are shown in Fig. 8.6 where we see the noisy synthesized bearings (left) and four (4) noisy sensor measurements at the moving array. The bearing (state)

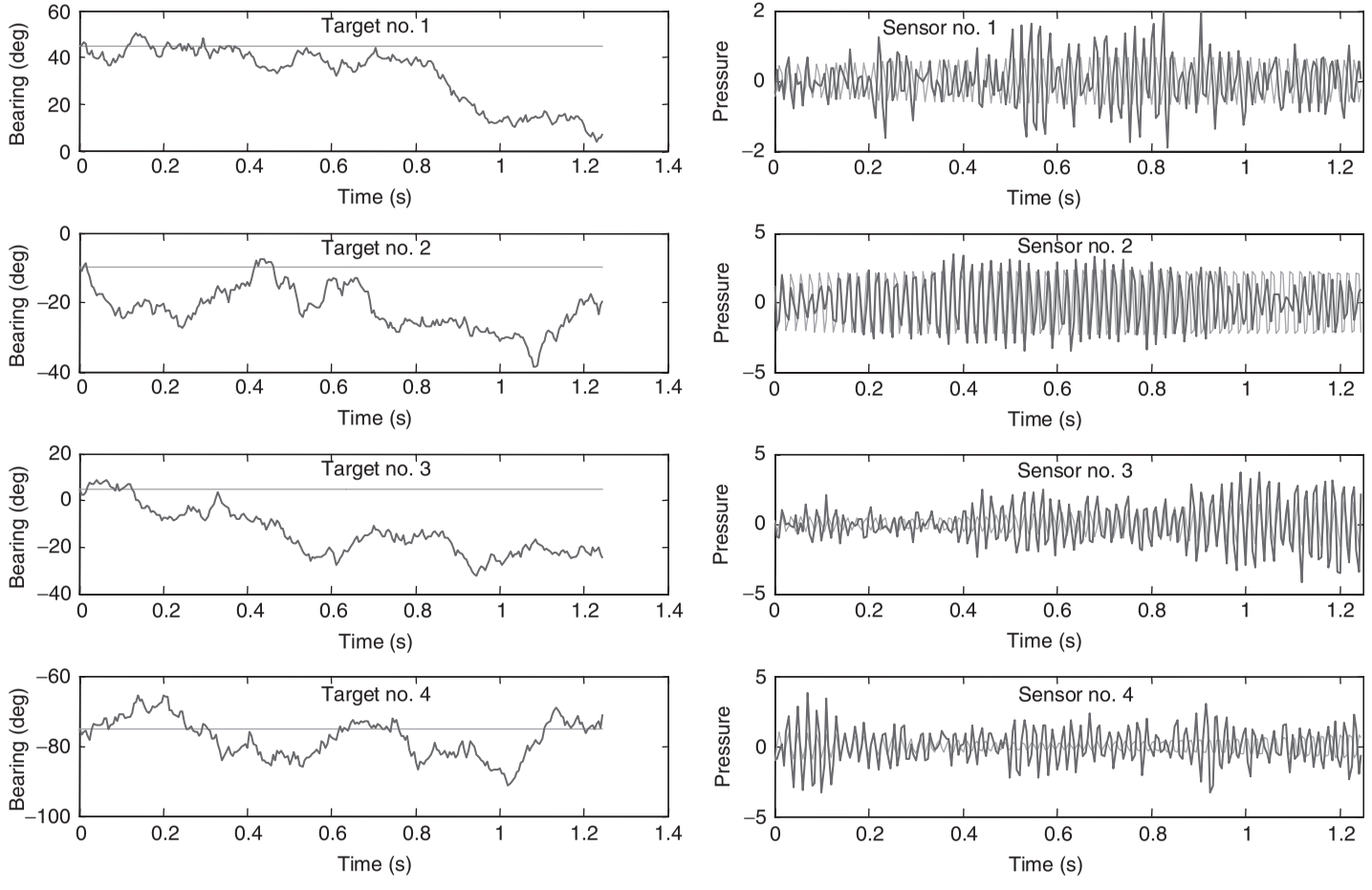


FIGURE 8.6 Synthetic aperture sonar tracking problem: Simulated target motion from initial bearings of 45°, −10°, 5° and −75° and array measurements (−10 dB *SNR*).

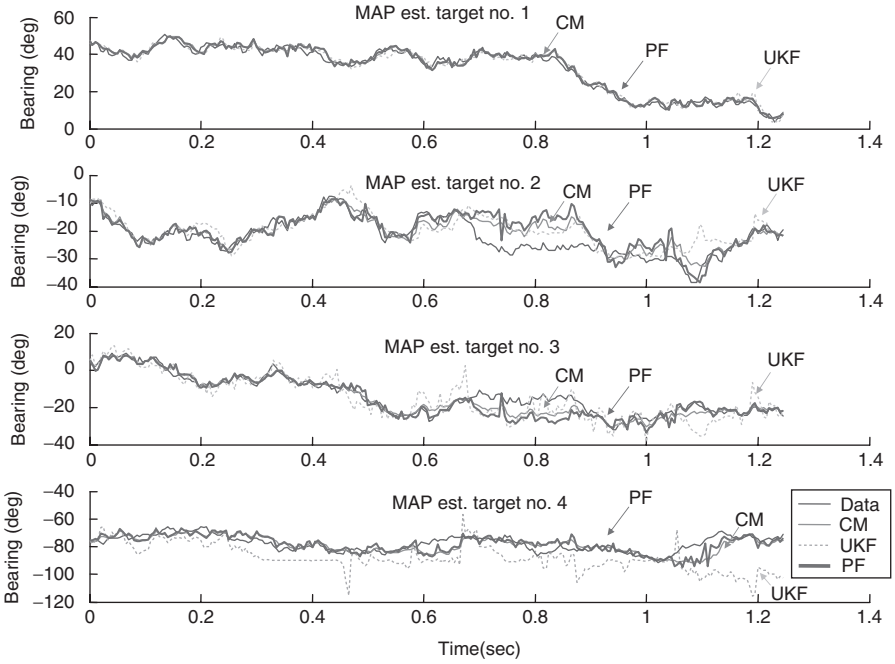


FIGURE 8.7 Particle filter bearing estimates for four targets in random motion: *PF* bearing (state) estimates and simulated target tracks (*UKF*, conditional mean, *MAP*).

estimates are shown in Fig. 8.7 where we observe the targets making a variety of course alterations. The *PF* (*MAP*) is able to track the target motions quite well while we observe the unscented Kalman filter (*UKF*) [18] unable to respond quickly enough and finally losing track completely for target no. 4. It should be noted that targets no. 2 and no. 4 "crossover" between 0.8 and 1.0 sec. The *PF* loses these tracks during this time period getting them confused but recovers by the 1 sec time step. Both the *MAP* and *MMSE* (*CM*) estimates using the estimated posterior provide excellent tracking. Note that these bearing inputs would provide the raw data for an *XY*-tracker [18]. The *PF* estimated or filtered measurements are shown in Fig. 8.8. As expected the *PF* tracks the measurement data quite well while the *UKF* is again in small error. Using the usual optimality tests for performance demonstrates that the *PF* processor works well, since each measurement channel is zero-mean and white with the *WSSR* lying below the threshold indicating a white innovations sequence demonstrating the tracking ability of the *PF* processor at least in a classical sense [18] as shown in Fig. 8.9. The instantaneous posterior distributions for the bearing estimates are shown in Fig. 8.10. Here we see the Gaussian nature of the bearing estimates generated by the random walk. Clearly, the *PF* performs quite well for this problem. Note also the capability of using the synthetic aperture, since we have only a 4-element sensor array, yet we are able to track 4 targets. Linear array theory implies with a static array that we should only be able to track $L - 1 = 3$ targets!

In this case study we have applied the bootstrap *PF* to an ocean acoustic synthetic aperture towed array target tracking problem to test the performance of a the particle filtering technique. The results are quite reasonable on this simulated data set.

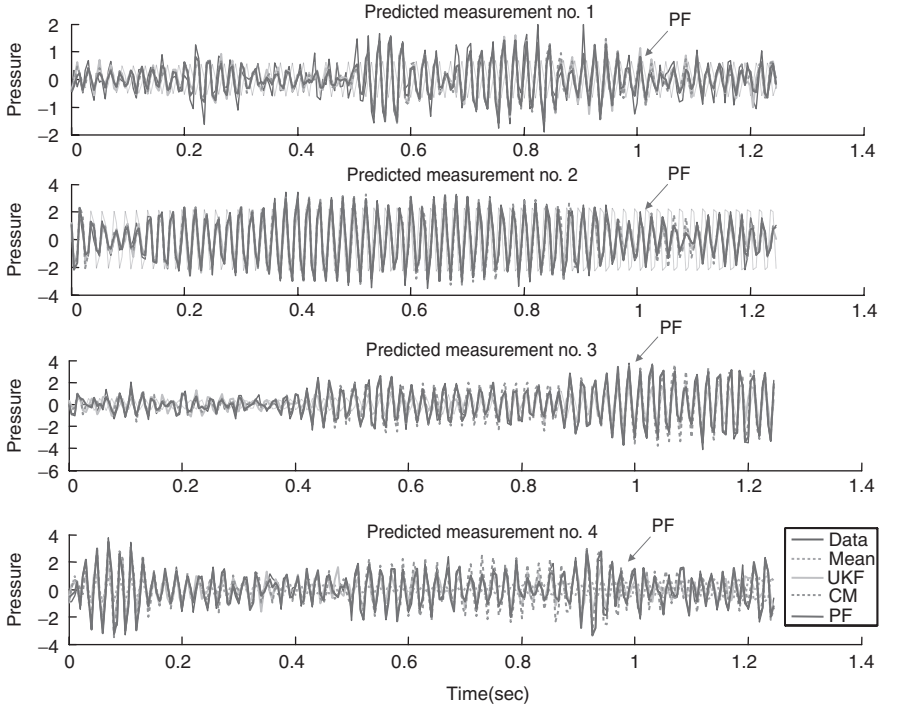


FIGURE 8.8 Particle filter predicted measurement estimates for four channel hydrophone sensor array.

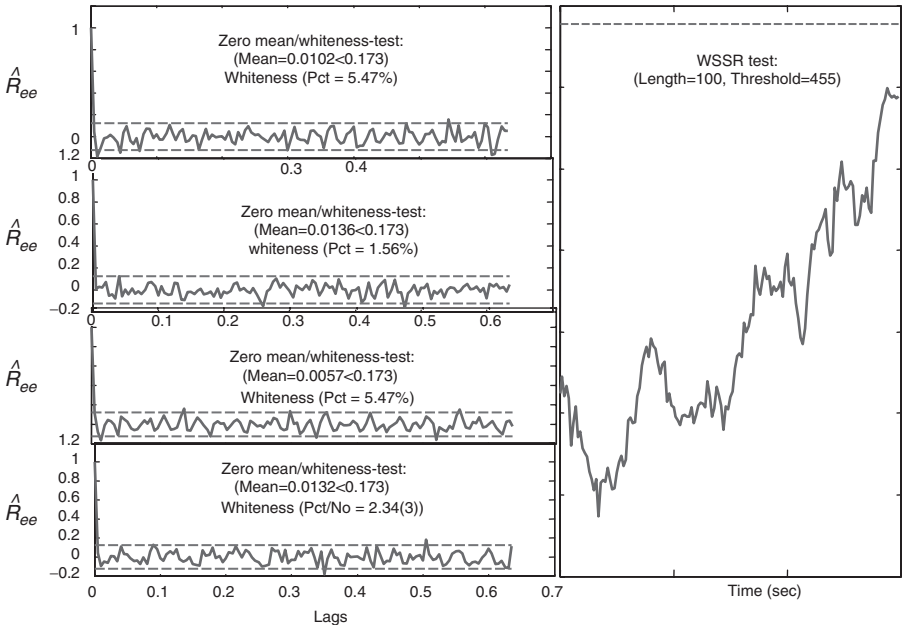


FIGURE 8.9 Particle filter classical performance metrics: zero-mean/whiteness tests for 45°, −10°, 5° and 75° targets as well the corresponding *WSSR* test.

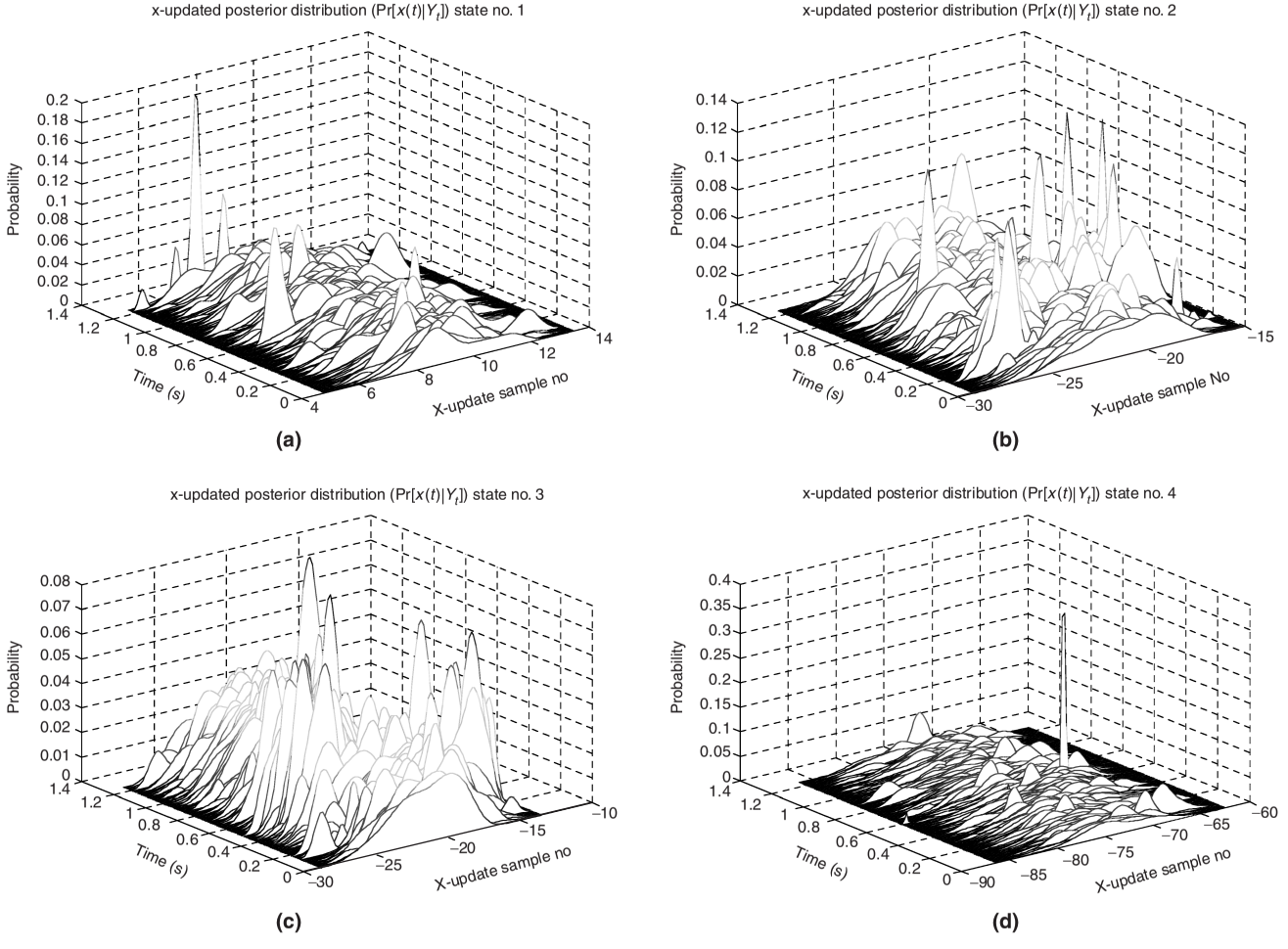


FIGURE 8.10 Particle filter instantaneous posterior bearing estimates: (a) 45° target no. 1. posterior. (b) −10° target no. 2 posterior. (c) 5° target no. 3. and (d) −75° target no. 4.

8.6 SUMMARY

In this chapter we have discussed the development of joint Bayesian state/parametric processors. Starting with a brief introduction, we defined the variety of problems based on the joint posterior distribution $\Pr(x(t), \theta(t)|Y_t)$ and its decomposition. We decided to focus on the joint problem of estimating *both* states and parameters simultaneously (on-line)—the most common problem of highest interest. We then briefly showed that all that is necessary for this problem is to define an "augmented" state consisting of the original state variables along with the unknown parameters typically modeled by a random walk when a dynamic parametric model is not available. This casts the joint problem into an optimal filtering framework. We then showed how this augmentation leads to a decomposition of the classical (*EKF*) processor and developed the "decomposed" form for illustrative purposes. The algorithm is implemented by executing the usual processor with the new augmented state vector. We also extended this approach to both the modern "unscented" and "particle-based" processors, again only requiring the state augmentation procedure to implement. It was shown that all of the processors required a random walk parametric model to function, while the particle filters could be implemented using the "roughening" (particle random walks) or any of the "move" techniques developed in Chapter 7 to track the parameters effectively. Besides applying these processors to the usual nonlinear trajectory estimation problem, we developed a case study for a synthetic aperture towed array and compared the modern to the particle-based processors.

MATLAB NOTES

SSPACK_PC is a 3^{rd} 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 [18] 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 post-processor (**SSPOST**) which 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 *UKF* including the unscented transformations. It also has included the new particle filter designs (see **http://choosh.ece.ogi.edu/rebel** for more details).

REFERENCES

1. L. Ljung, *System Identification: Theory for the User* (Englewood Cliffs, NJ: Prentice-Hall, 1987).
2. L. Ljung and T. Soderstrom, *Theory and Practice of Recursive Identification* (Boston: MIT Press, 1983).
3. T. Soderstrom and P. Stoica, *System Identification* (Englewood Cliffs, NJ: Prentice-Hall, 1989).
4. J. Norton, *An Introduction to Identification* (New York: Academic Press, 1986).
5. J. Liu, *Monte Carlo Strategies in Scientific Computing*, (New York: Springer-Verlag, 2001).
6. O. Cappe, E. Moulines and T. Ryden, *Inference in Hidden Markov Models*, (New York: Springer-Verlag, 2005).
7. J. Liu and M. West, "Combined parameter and state estimation in simulation-based filtering," in *Sequential Monte Carlo Methods in Practice* (A. Doucet, N. de Freitas and N. Gordon) pp. 197–223 (New York: Springer-Verlag, 2001).
8. A. Doucet, N. de Freitas and N. Gordon, *Sequential Monte Carlo Methods in Practice* (New York: Springer-Verlag, 2001).
9. S. Godsill and P. Djuric, "Special Issue: Monte Carlo methods for statistical signal processing." *IEEE Trans. Signal Proc.*, **50**, 173–499, 2002.
10. O. Cappe, S. Godsill and E. Moulines, "An overview of existing methods and recent advances in sequential Monte Carlo," *Proc. IEEE*, **95**, 5, 899–924, 2007.
11. G. Kitagawa and W. Gersch, *Smoothness Priors Analysis of Time Series* (New York: Springer-Verlag, 1997).
12. G. Kitagawa, "Self-organizing state space model," *J. Am. Statistical Assoc.*, **97**, 447, 1207–1215, 1998.
13. R. van der Merwe, A. Doucet, N. de Freitas and E. Wan, "The unscented particle filter," in *Advances in Neural Information Processing Systems 16* (Cambridge, MA: MIT Press, 2000).
14. N. Gordon, D. Salmond and A. Smith, "A novel approach to nonlinear non-gaussian Bayesian state estimation," *IEE Proc. F.*, **140**, 107–113, 1993.
15. S. Haykin and N. de Freitas, "Special Issue: Sequential state estimation: from Kalman filters to particle filters." *Proc. IEEE*, **92**, 3, 399–574, 2004.
16. C. Andrieu, A. Doucet, S. Singh, and V. Tadic, "Particle methods for change detection, system identification and control," *Proc. IEEE*, **92**, 6, 423–468, 2004.
17. S. Haykin, *Kalman Filtering and Neural Networks*. (New York: John Wiley, 2001).
18. J. Candy, *Model-Based Signal Processing*. (Hoboken, NJ: John Wiley/IEEE Press, 2006).
19. D. Simon, *Optimal State Estimation: Kalman H_∞ and Nonlinear Approaches* (Hoboken, NJ: John Wiley/IEEE Press, 2006).
20. A. Jazwinski, *Stochastic Processes and Filtering Theory*. (New York: Academic Press, 1970).
21. L. Ljung, "Asymptotic behavior of the extended Kalman filter as a parameter estimator for linear systems," *IEEE Trans. Auto. Control*, **AC-24**, 36–50, 1979.
22. R. van der Merwe, *Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models* OGI School of Science & Engr., Oregon Health & Science Univ., Ph.D. Dissertation, 2004.

23. A. Nelson, *Nonlinear Estimation and Modeling of Noisy Time-Series by Dual Kalman Filtering Methods* OGI School of Science & Engr., Oregon Health & Science Univ., Ph.D. Dissertation, 2000.

24. J. Candy, "Bootstrap particle filtering," *IEEE Signal Proc. Magz.*, **24**, 4, 73–85, 2007.

25. J. Rajan, P. Rayner and S. Godsill, "Bayesian approach to parameter estimation and interpolation of time-varying autoregressive processes using the Gibbs sampler," *IEE Proc-Vis. Image Signal Process.*, **144**, 4, 249–256, 1997.

26. N. Polson, J. Stroud and P. Muller, "Practical filtering with sequential parameter learning," *Univ. Chicago Tech. Rpt.*, 1–18, 2002.

27. C. Andrieu, A. Doucet, S. Singh and V. Tadic, "Particle methods for change detection, system identification and control," *Proc. IEEE*, **92**, 3, 423–438, 2004.

28. G. Storvik, "Particle filters in state space models with the presence of unknown static parameters," *IEEE Tran. Signal Proc.*, **50**, 2, 281–289, 2002.

29. P. Djuric, "Sequential estimation of signals under model uncertainty," in *Sequential Monte Carlo Methods in Practice* (A. Doucet, N. de Freitas and N. Gordon) pp. 381–400 (New York: Springer-Verlag, 2001).

30. D. Lee and N. Chia, "A particle algorithm for sequential Bayesian parameter estimation and model selection," *IEEE Tran. Signal Proc.*, **50**, 2, 326–336, 2002.

31. A. Doucet and V. Tadic, "Parameter estimation in general state-space models using particle methods," *Ann. Inst. Stat. Math.*, **55**, 2, 409–422, 2003.

32. C. Andrieu, A. Doucet and V. Tadic, "On-line parameter estimation in general state-space models," *Proc. IEEE Conf. Decision and Control*, pp. 332–337, 2005.

33. J. Vermaak, C. Andrieu, A. Doucet and S. Godsill, "Particle methods for Bayesian modeling and enhancement of speech signals," *IEEE Trans. Speech Audio Proc.*, **10**, 3, 173–185, 2002.

34. T. Schoen and F. Gustafsson, "Particle filters for system identification of state-space models linear in either parameters or states," *Linkoping University Report*, LITH-ISY-R-2518, 2003.

35. R. Williams, "Creating an acoustic synthetic aperture in the ocean," *J. Acoust. Soc. Am.*, **60**, 60–73, 1976.

36. N. Yen and W. Carey, "Applications of synthetic aperture processing to towed array data," *J. Acoust. Soc. Am.*, **60**, 764–775, 1976.

37. S. Stergiopoulus and E. Sullivan, "Extended towed array processing by an overlap correlator," *J. Acoust. Soc. Am.*, **86**, 764–775, 1976.

38. E. Sullivan, W. Carey and S. Stergiopoulus, "Editorial in special issue on acoustic synthetic aperture processing," *IEEE J. Ocean. Eng.*, **17**, 1–7, 1992.

39. D. Ward, E. Lehmann and R. Williamson, "Particle filtering algorithm for tracking and acoustic source in a reverberant environment," *IEEE Trans. Speech and Aud. Proc.*, **11**, 6, 826–836, 2003.

40. M. Orton and W. Fitzgerald, "Bayesian approach to tracking multiple targets using sensor arrays and particle filters," *IEEE Trans. Signal Proc.*, **50**, 2, 216–223, 2002.

41. E. Sullivan and J. Candy, "Space-time array processing: The model-based approach," *J. Acoust. Soc. Am.*, **102**, 5, 2809–2820, 1997.

PROBLEMS

8.1 Suppose we are given the following innovations model (in steady state)

$$\begin{aligned}\hat{x}(t) &= a\hat{x}(t-1) + ke(t-1)\\ y(t) &= c\hat{x}(t) + e(t)\end{aligned}$$

where $e(t)$ is the zero-mean, white innovations sequence with covariance, R_{ee}.

(*a*) Derive the Wiener solution using the spectral factorization method of Sec. 4.5.

(*b*) Develop the linear steady-state *BP* for this model.

(*c*) Develop the parametrically adaptive processor to estimate k and R_{ee}.

8.2 As stated in the chapter, the *XBP* convergence can be improved by incorporating a gain gradient term in the system Jacobian matrices, that is,

$$A_\theta^*[x,\theta] := A_\theta[x,\theta] + [\nabla_\theta K_x(\Theta)]e(t) \quad \text{for } \mathcal{K} := [K_x \mid K_\theta]$$

(*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 we would like to implement this modification, does there exist a numerical solution that could be used? If so, describe it.

8.3 Using the following *scalar* Gauss-Markov model

$$\begin{aligned}x(t) &= Ax(t-1) + w(t-1)\\ y(t) &= C\hat{x}(t) + v(t)\end{aligned}$$

with the usual zero-mean, R_{ww} and R_{vv} covariances.

(*a*) Let $\{A, C, K, R_{ee}\}$ be scalars, develop the *ABSP* solution to estimate A from noisy data.

(*b*) Can these algorithms be combined to "tune" the resulting hybrid processor?

8.4 Suppose we are given the following structural model

$$\begin{aligned}m\ddot{x}(t) + c\dot{x} + kx(t) &= p(t) + w(t)\\ y(t) &= \beta x(t) + v(t)\end{aligned}$$

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*: $\ddot{x}(t) \approx \frac{x(t)-2x(t-1)+x(t-2)}{\Delta_t^2}$).

(*b*) Suppose we would like to estimate the spring constant k from noisy displacement measurements, develop the *ABSP* to solve this problem.

(*c*) Transform the discrete Gauss-Markov model to the innovations representation. (*Hint*: Use the *KSP* equations of [18]).

(*d*) Solve the parameter estimation problem using the innovations model, that is, develop the estimator of the spring constant.

8.5 Given the *ARMAX* model

$$y(t) = -ay(t-1) + bu(t-1) + e(t)$$

with innovations covariance, R_{ee}:

(*a*) Write the expressions for the *ABSP* in terms of the *ARMAX* model.

(*b*) Write the expressions for the *ABSP* in terms of the state–space model.

8.6 Consider tracking a body falling freely through the atmosphere [18]. We assume it is falling down in a straight line towards a radar. The state vector is defined by: $x := [z \quad \dot{z} \quad \beta]$ where $\beta \sim \mathcal{N}(\mu_\beta, R_{\beta\beta}) = \mathcal{N}(2000, 2.5 \times 10^5)$ 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), ρ is the atmospheric density (with ρ_o (3.4×10^{-3}) density at sea level) and k_ρ a decay constant (2.2×10^4). 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 *ABSP* If so, write out the explicit algorithm in terms of the parametrically adaptive algorithm of this chapter.

(*b*) Develop the *XBP* for this problem and perform the discrete simulation using *MATLAB*.

(*c*) Develop the *LZ-BP* for this problem and perform the discrete simulation using *MATLAB*.

(*d*) Develop the *PF* for this problem and perform the discrete simulation using *MATLAB*.

8.7 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 θ.

(*b*) Expand $\mathbf{y} = \mathbf{h}(\theta)$ in a Taylor series about θ_o and incorporate the first order approximation into the *MAP* estimator (approximate).

(*c*) Expand $\mathbf{y} = \mathbf{h}(\theta)$ in a Taylor series about θ_o and incorporate the second order approximation into the *MAP* estimator (approximate).

(*c*) Develop and iterated version of both estimators in (b) and (c). How do they compare?

(*d*) Use the parametrically adaptive formulation of this problem assuming the measurement model is time-varying. Construct the *ABSP* assuming that θ is modeled by a random walk. How does this processor compare to the iterated versions?

8.8 Suppose 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

$$\begin{aligned} \mathcal{H}_o\colon y(t) &= v(t) \quad \text{for } v \sim \mathcal{N}(0, R_{vv}) \\ \mathcal{H}_1\colon y(t) &= s(t) + v(t) \end{aligned}$$

The signal is modeled by a Gauss-Markov model

$$s(t) = a[s(t-1)] + w(t-1) \quad \text{for } w \sim \mathcal{N}(0, R_{ww})$$

(*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 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 *ABSP* to jointly estimate the signal and parameter.)

(*c*) Calculate a sequential form of the likelihood ratio above by letting the batch of measurements, $N \to t$. Calculate the corresponding threshold and construct the detector (binary hypothesis test). Note there are two thresholds for this type of detector.

8.9 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:

$$s(t) = \sqrt{2P} \sin [\omega_c t + k_p m(t)] \quad \text{(PM)}$$

or

$$s(t) = \sqrt{2P} \sin [\omega_c t + 2\pi k_f \int_{-\infty}^{t} m(\tau) d\tau] \quad \text{(FM)}$$

where P is a constant, ω_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 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 *XBP* design.

(*b*) Construct an equivalent receiver for the FM system.

(*c*) Assume that the message amplitude parameter α is unknown, construct the *ABSP* receiver for the PM system to jointly estimate the message and parameter.

(*d*) Under the same assumptions as (c), construct the *ABSP* 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?

8.10 We are given the population model of the Chapter 7 case study and would like to "parameterize" it for adaptive processing, since we know the parameters are not very well known. The state transition and corresponding measurement model are given by

$$x(t) = \frac{1}{2}x(t-1) + \frac{25x(t-1)}{1+x^2(t-1)} + 8\cos(1.2(t-1)) + w(t-1)$$

$$y(t) = \frac{x^2(t)}{20} + v(t)$$

where $\triangle t = 1.0$, $w \sim \mathcal{N}(0, 10)$ and $v \sim \mathcal{N}(0, 1)$. The initial state is Gaussian distributed with $\bar{x}(0) \sim \mathcal{N}(0.1, 5)$.

In terms of the nonlinear state–space representation, we have

$$a[x(t-1)] = \frac{1}{2}x(t-1) + \left(\frac{25x(t-1)}{1+x^2(t-1)}\right)$$

$$b[u(t-1)] = 8\cos(1.2(t-1))$$

$$c[x(t)] = \frac{x^2(t)}{20}$$

(*a*) Choose the model constants: 25, 8, 0.5 and $\frac{1}{20}$ as the unknown parameters, reformulate the state estimation problem as a parameter estimation problem with unknown parameter vector, Θ and a random walk model with corresponding process noise variance, $R_{ww} = diag[1 \times 10^{-6}]$.

(*b*) Develop the joint *SPBP* algorithm to solve this problem. Run the *SPBP* algorithm and discuss the performance results.

(*c*) Develop the joint *PF* algorithm to solve this problem. Run the *PF* algorithm and discuss the performance results.

(*d*) Choose to "move" the particles using the roughening approach, how do these results compare to the standard bootstrap algorithm?

(*e*) Develop the joint linearized (*UKF*) *PF* algorithm to solve this problem. Run this *PF* algorithm and discuss the performance results.

9

DISCRETE HIDDEN MARKOV MODEL BAYESIAN PROCESSORS

9.1 INTRODUCTION

In this chapter we develop discrete (event) hidden Markov models. All of the Bayesian processors we have discussed are, in fact, hidden Markov processors, since the internal states are usually not measured directly and are therefore "hidden" by definition, but the distinguishing factor is the type of underlying process governing the sequence. In fact, the (state) transition matrix is a "probability" matrix with specific properties that distinguish it uniquely from other dynamic systems. These discrete representations of stochastic processes find enormous application in the speech, economics, biomedical, communications and music areas where coding approaches are prevalent. We discuss the development of the basic processor and investigate a case study in communications to demonstrate the design and application.

9.2 HIDDEN MARKOV MODELS

A discrete-time hidden Markov model (*HMM*) is a stochastic representation (model) of a process that can be used for simulation, modeling and estimation much the same as the state–space model is used for dynamic (physical) systems. These models are prevalent in acoustics, biosciences, climatology, control, communications, econometrics, text recognition, image processing, signal processing and speech processing [1]. Perhaps its distinguishing feature is that it is a "probabilistic model" in the sense that it is driven by internal probability distributions for both states and observations or equivalently measurements. Here the state transition matrix prevalent in linear systems theory is still valid and also called a transition matrix, but it is a discrete

Bayesian Signal Processing. By James V. Candy

probability matrix with rows summing to unity and in some cases (doubly stochastic) columns summing to unity as well. The underlying structure from which the *HMM* evolves is the Markov chain of Chapter 3 along with the sequential Bayesian recursions of Chapter 2. We start with the idea of a Markov chain and its decomposition basics leading to *HMM*.

9.2.1 Discrete-Time Markov Chains

A discrete-time Markov chain is characterized by a state variable that changes at certain time instances [2–4]. At each time-step t the state is defined by $x(t) \in \mathcal{X}$ (state–space) and $\mathcal{X} = \{\mathcal{X}_1, \ldots, \mathcal{X}_{N_x}\}$. The probability that at time t the chain occupies state i is defined by $\Pr(x_i(t))$. The dynamics of the Markov chain are represented by its *transition probability*, $a_{mn}(t-1,t) := \Pr(x_n(t)|x_m(t-1))$ where $x_i(t) := \{x(t) = \mathcal{X}_i\}$. This expression means that the probability that the state at time t is $\mathcal{X}_n$ given that it is currently in state $\mathcal{X}_m$ at time $t-1$ for $(\mathcal{X}_m, \mathcal{X}_n) \in \mathcal{X}$. Here the key Markovian assumption is that the transition probability a_{mn} applies *whenever* state $\mathcal{X}_m$ is "visited" independent of the "past" and the *path* or previous states employed to reach $\mathcal{X}_m$. This is merely a statement of the Markovian property that

$$a_{mn}(t-1,t) = \Pr(x_n(t)|x_m(t-1), \ldots, x_\ell(0)) = \Pr(x_n(t)|x_m(t-1)) \quad \forall \quad t \text{ and } (\mathcal{X}_m, \mathcal{X}_n) \in \mathcal{X} \tag{9.1}$$

Further, if the chain is *homogeneous-in-time*, then $a_{mn}(t-1,t)$ depends only on the time difference (in general) and therefore the transition probability is *stationary* such that $a_{mn}(t-1,t) \to a_{mn}$ with $a_{mn} \geq 0$ and $\sum_{n=1}^{N_x} a_{mn} = 1$ [2].

Summarizing, a *discrete-time Markov chain* is characterized by:

- a finite set of known N_x-states: $\mathcal{X} = \{\mathcal{X}_1, \ldots, \mathcal{X}_{N_x}\}$;
- a non-negative set of *state*-transition probabilities for $(\mathcal{X}_m, \mathcal{X}_n) \to \{a_{mn}\}$; and
- a sequence of random variables: $x_m(0), x_n(1), \ldots \in \mathcal{X}$ that satisfy the Markovian property:

$$a_{mn}(t-1,t) = \Pr(x_n(t)|x_m(t-1)) \,\forall\, t \text{ and all states } (\mathcal{X}_m, \mathcal{X}_n) \in \mathcal{X}$$

The elements of the homogeneous chain are embedded in the $N_x \times N_x$ state transition probability matrix, $A = [a_{mn}]$; $m, n = 1, \ldots, N_x$. The chain can be specified pictorially by a *directed graph* with *nodes* representing states and *arcs* or *arrows* corresponding to the transition probabilities as illustrated in Fig. 9.1.

Example 9.1

Suppose we are given a two-state ($N_x = 2$) Markov chain with transition probability

$$a_{mn} = \Pr(x_n(t)|x_m(t-1)); \quad m, n = 1, 2$$

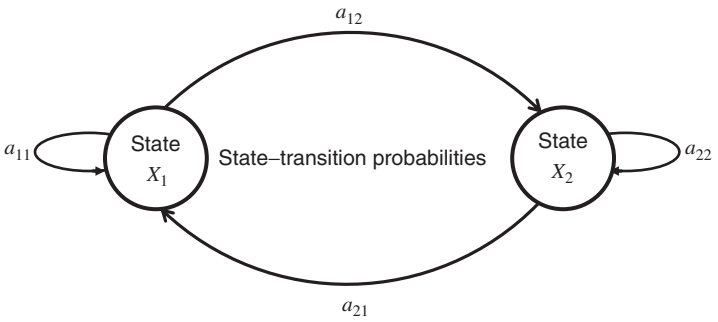


FIGURE 9.1 Directed graph representation of two-state Markov chain.

and $a_{mn} = \{0.55, 0.45, 0.25, 0.75\}$. Construct the state-transition probability matrix A and the corresponding directed graph. The transition probability is given by:

$$A = \begin{bmatrix} 0.55 & 0.45 \\ 0.25 & 0.75 \end{bmatrix}$$

The resulting graph is shown in Fig. 9.1. △△△

9.2.2 Hidden Markov Chains

A *hidden* Markov model (*HMM*) is simply a Markov chain in which all of the states are *not* observed—some are hidden. In this case we introduce the observation or measurement or output process where only a *subset* of the states are observed directly. Thus, the essential difference between a Markov chain and a hidden Markov model is that for a *HMM* there is *not* a one-to-one correspondence between the states and observations (output measurements). It is not possible to tell which state the model was in by merely observing the outputs of the chain. We illustrate the structural model in Fig. 9.2a. Note that when the states are directly observed in a, then the observations and states are identical.

Thus, we differentiate the hidden Markov chain or *HMM* from the Markov chain by introducing an *observation* or *measurement* process [5–8]. Here the state sequence is *not* known, that is, it is *hidden* in the measurement sequence. Thus, at every time-step t, the system generates a measurement or observation $y(t)$ according to a probability distribution that depends on the state, $x(t)$. The number of observations N_y corresponds to a known distinct set, that is, at time t the observation is $y(t) \in \mathcal{Y}$ (observation space) with $\mathcal{Y} = \{\mathcal{Y}_1, \ldots, \mathcal{Y}_{N_y}\}$. We define the corresponding "discrete" *observation probability* (likelihood) by[1]:

$$c_{k\ell}(t,t) := \Pr(y_\ell(t)|x_k(t)) \tag{9.2}$$

[1] This probability expression has two subscripts to annotate the discrete state ($x_k(t)$) and the discrete measurement or observation ($y_\ell(t)$). Most references assume a continuous observation and use the notation $c_k(y(t))$ [9–12].

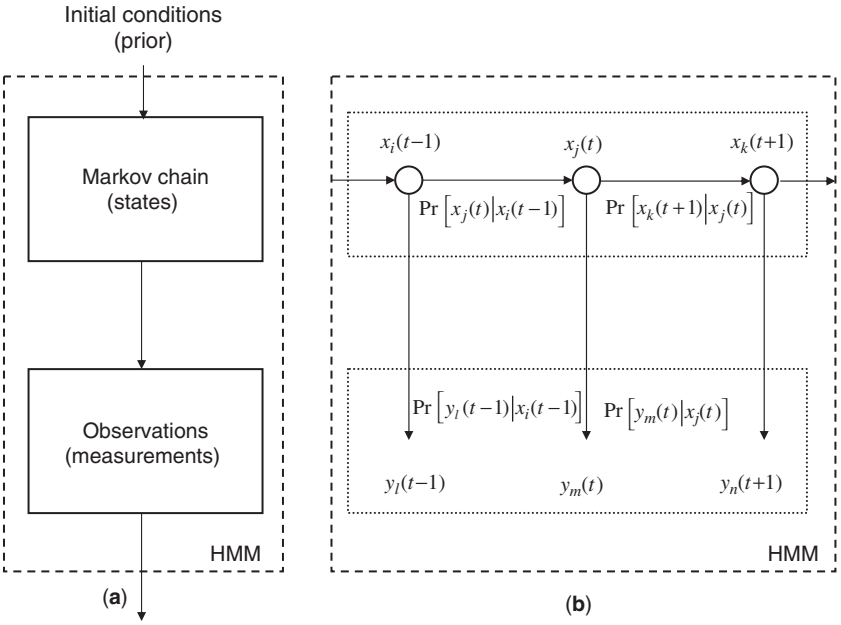


FIGURE 9.2 Hidden Markov model structure: (a) Markov chain (states) and observations (measurements). (b) Markov chain with state-transition probabilities and observations with measurement probabilities (likelihoods).

Again for the homogeneous case, $c_{k\ell}(t,t) \to c_{k\ell}$ and $c_{k\ell} \geq 0$ and $\sum_{\ell=1}^{N_y} c_{k\ell} = 1$. As in the chain, we have the associated $N_y \times N_x$ *observation probability* matrix given by $C = [c_{k\ell}]$ for $k = 1, \ldots, N_x$; $\ell = 1, \ldots, N_y$ with $C \in R^{N_y \times N_x}$. The final ingredient to characterize the *HMM* is the prior or initial probability distribution given by $\Pr(x(0) = \mathcal{X}_i(0))$; $i = 1, \ldots, N_x$ which represents the initial probability of the chain.

Summarizing a *hidden Markov model* is specified by the model $\Sigma := \{A, C, \Pr(x_i(0))\}$ (homogeneous case) where:

- The *state-transition probability* matrix is:

$$A = [a_{mn}] = \Pr(x_n(t)|x_m(t-1)); \quad m, n = 1, \ldots, N_x;$$

- The *observation probability* matrix is:

$$C = [c_{k\ell}] = \Pr(y_\ell(t)|x_k(t)) \quad \text{for } k = 1, \ldots, N_x; \quad \ell = 1, \ldots, N_y \text{ and}$$

- The *prior* probability is:

$$\Pr(x_i(0)); \quad i = 1, \ldots, N_x.$$

where N_x, N_y are the number of states and observations (measurements), respectively (see Fig. 9.2*b*).

We must realize a subtle point that in contrast to dynamic physical systems where the states and measurements can be any real value or number, the *HMM* states or observations can *only* assume pre-defined integer values, $\mathcal{X} = \{\mathcal{X}_1, \ldots, \mathcal{X}_{N_x}\}$ and $\mathcal{Y} = \{\mathcal{Y}_1, \ldots, \mathcal{Y}_{N_y}\}$—this is very important to comprehend. It is the transition and observation probabilities that drive the occurrence of an individual state and observation event, since both are merely (integer) realizations of model outputs. For example, the mapping or quantization of a "real" physical communications signal to a binary coded representation takes the form of a sequence of a 0 or 1 integer value at each time-step which are mapped to the observation sequence (see Sec. 9.7).

It should also be noted that with the addition of the observation process, it is simple to define an underlying *HMM state–space model* as:

$$\begin{aligned} x(t) &= Ax(t-1) + w(t-1) \\ y(t) &= Cx(t) + v(t) \end{aligned} \tag{9.3}$$

where w, v are uncorrelated (white) sequences (noise) with x, y the usual state and measurement sequences and associated initial conditions $x(0)$, all specified by the *HMM* above of Fig. 9.2*b*.

Note that the additive sequences (noise) are *not* necessarily Gaussian and therefore the linear Bayesian processor (Kalman filter) is *not* optimum in this case. However, it has been shown [13] that under some (sufficient) conditions (stationarity, etc.) that an optimal minimum error variance estimator (Kalman filter) can be constructed based on a stochastic realization of a *HMM*. The results of this design are capable of providing reasonable estimates of the *HMM* states and observations. It is also important to understand that exists state–space representations in which *both* discrete *HMM* models are combined with dynamic (physical) state–space systems. For instance, one prevalent form is termed *switching* models in which the discrete *HMM* determines which underlying dynamic model applies at a given time-step. This is an approach frequently used in target tracking problems [14] to provide multiple model choices.

9.3 PROPERTIES OF THE HIDDEN MARKOV MODEL

In this section we investigate some of the underlying probabilistic properties of the discrete *HMM*. To no surprise it matches our Bayesian processor development of the previous chapters. After all, once placed in the state–space representation, all Bayesian properties should hold. We start with the joint distribution.

The *HMM* is a probabilistic model of the joint collection of random variates (Y_t, X_t). Critical properties of the *HMM* rely on basic Bayesian methodologies and developments. Perhaps the most useful notions inherited from the Markov chain theory are the two major properties of conditional independence that are used over and over again

along with Bayes' rule. That is, under a *HMM* there are two assumptions enabling the development of the underlying techniques:

1. The hidden variables are first-order Markov[2]: $\Pr(x(t)|X_{t-1}, Y_{t-1}) = \Pr(x(t)|x(t-1))$ (state-transition); and
2. The observation is independent of other variates given the state (at time t): $\Pr(y(t)|Y_{t-1}, X_t) = \Pr(y(t)|x(t))$ (likelihood).

These properties imply that the underlying joint distribution can be expanded as:

$$\begin{aligned}\Pr(Y_t, X_t) &= \Pr(y(t), Y_{t-1}, x(t), X_{t-1}) \\ &= \Pr(y(t), x(t)|Y_{t-1}, X_{t-1}) \times \Pr(Y_{t-1}, X_{t-1}) \end{aligned} \tag{9.4}$$

Continuing to apply Bayes' rule to this expression gives

$$\begin{aligned}\Pr(Y_t, X_t) &= [\Pr(y(t)|x(t), X_{t-1}, Y_{t-1}) \times \Pr(x(t)|X_{t-1}, Y_{t-1})] \\ &\quad \times \Pr(Y_{t-1}, X_{t-1}) \end{aligned} \tag{9.5}$$

or finally

$$\Pr(Y_t, X_t) = \Pr(y(t)|x(t)) \times \Pr(x(t)|x(t-1)) \times \Pr(Y_{t-1}, X_{t-1}) \tag{9.6}$$

where we have applied the conditional independence properties of the *HMM*. From the chain rule and these independence properties, we can expand this expression even further to obtain

$$\Pr(Y_t, X_t) = \prod_{k=0}^{t} \Pr(y(k)|x(k)) \times \prod_{k=2}^{t} \Pr(x(k)|x(k-1)) \times \Pr(x(0)) \tag{9.7}$$

Thus, in order to characterize a *HMM* we require the following probabilities:

- Prior: $\Pr(x(0))$;
- Transition: $\Pr(x(t)|x(t-1))$; and
- Likelihood: $\Pr(y(t)|x(t))$.

These quantities correspond to the classic[3] definition of a hidden Markov model [10]. The underlying Markov chain is usually assumed to be homogeneous-in-time with associated stochastic state transition matrix defined before by $A = a_{mn} = \Pr(x_n(t)|x_m(t-1))\ \forall t$ and the observation (measurement) probability is

[2] Any N^{th}-order Markov process can be transformed to a first-order process [15].

[3] Classical notation: $\pi_o \rightarrow Pr(x(0))$; $a_{ij} \rightarrow \Pr(x_j(t)|x_i(t-1))$; and $b_i(y(t)) \rightarrow \Pr(y(t)|x_i(t))$ (continuous observation) or $b_{ij} \rightarrow \Pr(y_j(t)|x_i(t))$ (discrete observation).

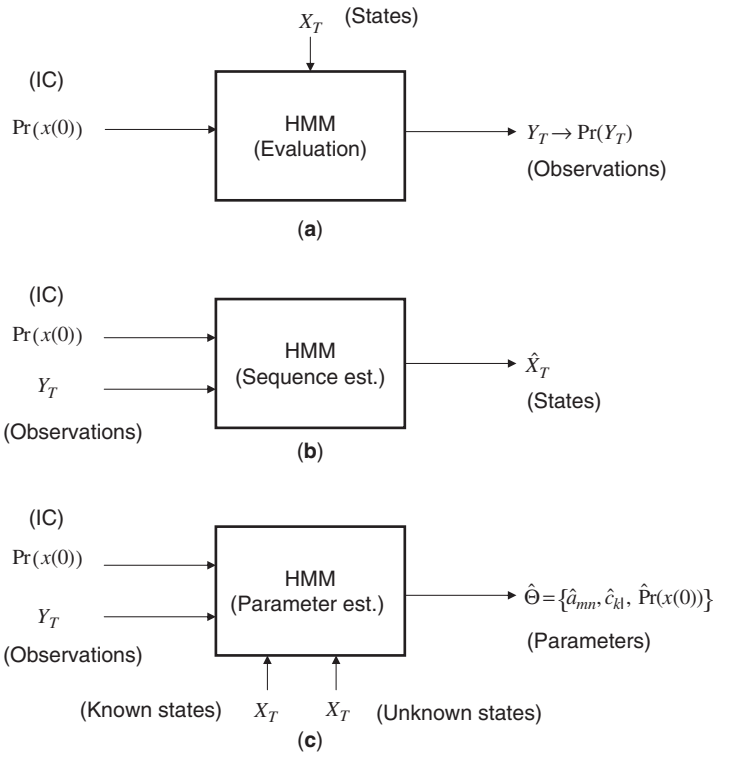


FIGURE 9.3 HMM basic problems: (a) Evaluation problem. (b) Sequence estimation problem. (c) Parameter estimation problem.

given by $C = [c_{k\ell}] = \Pr(y_\ell(t)|x_k(t))$. The *HMM*-parameters are usually specified by $\Sigma = (A, C, \Pr(x_i(0)))$. Here the measurements Y_t are *observed* and the states or internal variables are *hidden*. In order to *generate* (simulate) samples from the *HMM*, the initial "state" distribution is generated followed by the likelihood (prior $\rightarrow$ transition $\rightarrow$ likelihood). It is important to understand that each measurement sample simulated requires new state samples, that is, two synthesized measurement samples originated from two different states in the hidden Markov chain.

9.4 *HMM* OBSERVATION PROBABILITY: EVALUATION PROBLEM

With these *HMM* properties available, we can now pose the first problem of interest. With the model known and a set of observations available, how can we evaluate the performance of the *HMM* to faithfully synthesize observations? One way to approach this problem is to estimate the corresponding observation probability $\Pr(Y_T)$ and use it to "validate" that the model and observations are compatible (see Fig. 9.3a). This approach is especially useful when we are to compare or "match" different models

to the same observation sequence and search for that model which provides the best match. Thus, calculating the total observation probability provides a solution to the *evaluation problem* of *HMM*, that is,

GIVEN the observation sequence, Y_T and *HMM* parameters Σ, *FIND* the total observation probability, $\Pr(Y_T)$ for $Y_T = \{y(0), \ldots, y(T)\}$.

The *observation probability* is obtained by marginalizing (summing over) the total probability

$$\Pr(Y_t) = \sum_{X_t} \Pr(Y_t, X_t) = \sum_{X_t} \left(\prod_{k=0}^{t} \Pr(y(k)|x(k)) \times \prod_{k=2}^{t} \Pr(x(k)|x(k-1)) \times \Pr(x(0)) \right) \tag{9.8}$$

Blindly computing this summation is a very inefficient method to estimate the desired probability. Instead, we factor the states using their first-order Markov property (conditional independence) such that

$$\Pr(Y_t) = \sum_{x(t), x(t-1)} \Pr(Y_t, x(t), x(t-1)) \tag{9.9}$$

but continuing the expansion over Y_t we have

$$\begin{aligned} \Pr(Y_t, x(t), x(t-1)) &= \Pr(Y_{t-1}, x(t-1), y(t), x(t)) \\ &= \Pr(y(t), x(t)|Y_{t-1}, x(t-1)) \times \Pr(x(t-1), Y_{t-1}) \end{aligned}$$

or

$$\begin{aligned} \Pr(Y_t, x(t), x(t-1)) = \Pr(y(t)|x(t), Y_{t-1},\ x(t-1)) &\times \Pr(x(t)|Y_{t-1}, x(t-1)) \\ &\times \Pr(x(t-1), Y_{t-1}) \end{aligned}$$

which yields the final expression

$$\Pr(Y_t, x(t), x(t-1)) = \Pr(y(t)|x(t)) \times \Pr(x(t)|x(t-1)) \times \Pr(Y_{t-1}, x(t-1)) \tag{9.10}$$

Marginalizing, we obtain

$$\begin{aligned} \Pr(Y_t, x(t)) &= \sum_{x(t-1)} \Pr(Y_t, x(t), x(t-1)) \\ &= \sum_{x(t-1)} \Pr(y(t)|x(t)) \times \Pr(x(t)|x(t-1)) \times \Pr(Y_{t-1}, x(t-1)) \end{aligned} \tag{9.11}$$

Define the *forward operator* as $\mathcal{F}_{\mathcal{X}}(t) := \Pr(Y_t, x(t) = \mathcal{X})$, then rewriting Eq. 9.11 we have

$$\begin{aligned}\mathcal{F}_k(t) &= \Pr(Y_t, x_k(t)) \\ &= \Pr(y(t)|x_k(t)) \sum_{\ell} \Pr(x_k(t)|x_\ell(t-1)) \times \Pr(Y_{t-1}, x_\ell(t-1))\end{aligned}$$

or substituting for the previous time-step, we obtain the *forward recursion* for the *HMM*

$$\mathcal{F}_k(t) = \Pr(y(t)|x_k(t)) \sum_{\ell} \Pr(x_k(t)|x_\ell(t-1)) \times \mathcal{F}_\ell(t-1) \tag{9.12}$$

Now, if we assume a stationary chain, then $A = [a_{k\ell}]$, $C = [c_{k\ell}]$, and this result can be expressed in terms of transition probabilities simply as

$$\mathcal{F}_k(t) = \sum_{\ell} a_{k\ell} \times c_{k\ell} \times \mathcal{F}_\ell(t-1) \quad \text{for } c_{k\ell} = \Pr(y_\ell(T)|x_k(T)) \tag{9.13}$$

Clearly marginalizing over $x(t)$ gives the total *observation probability* as

$$\Pr(Y_T) = \sum_{k} \mathcal{F}_k(T) = \sum_{x_k(t)} \Pr(Y_T, x_k(t)) \tag{9.14}$$

Thus, we have the *forward recursion algorithm* for *HMM* that can be used to obtain the total observation probability as:

- Initialize: $\mathcal{F}_k(0) = \Pr(x_k(0)) \times c_{k0}$;
- Recursion: $\mathcal{F}_k(t) = \sum_\ell a_{k\ell} \times c_{k\ell} \times \mathcal{F}_\ell(t-1)$;
- Termination: $\Pr(Y_T) = \sum_k \mathcal{F}_k(T)$.

This algorithm will be used not only to estimate the observation probability as the solution to the evaluation problem, but also to combine with another recursion to estimate model states and parameters.

Before we close this section, consider the following example of simulating a *HMM*.

Example 9.2

Suppose we have a discrete binary signal with the two-states ($N_x = 2$) specified by: $\{x_1(t) = 1,\ x_2(t) = 2\}$. The observation is also discrete with $N_y = 3$ and specified by $\{y_1(t) = 1,\ y_2(t) = 2, y_3(t) = 3\}$. The transition and observation probabilities are given by:

$$a_{mn} = \Pr(x_n(t)|x_m(t-1)); \quad m, n = 1, 2 \text{ and } c_{kn} = \Pr(y_k(t)|x_n(t)); \quad k = 1, 2, 3$$

with $a_{mn} = \{0.6, 0.4; 0.3, 0.7\}$ and $c_{kn} = \{0.50, 0.25, 0.25; 0.35, 0.25, 0.40\}$. Construct the state transition probability matrix A and the corresponding directed graph.

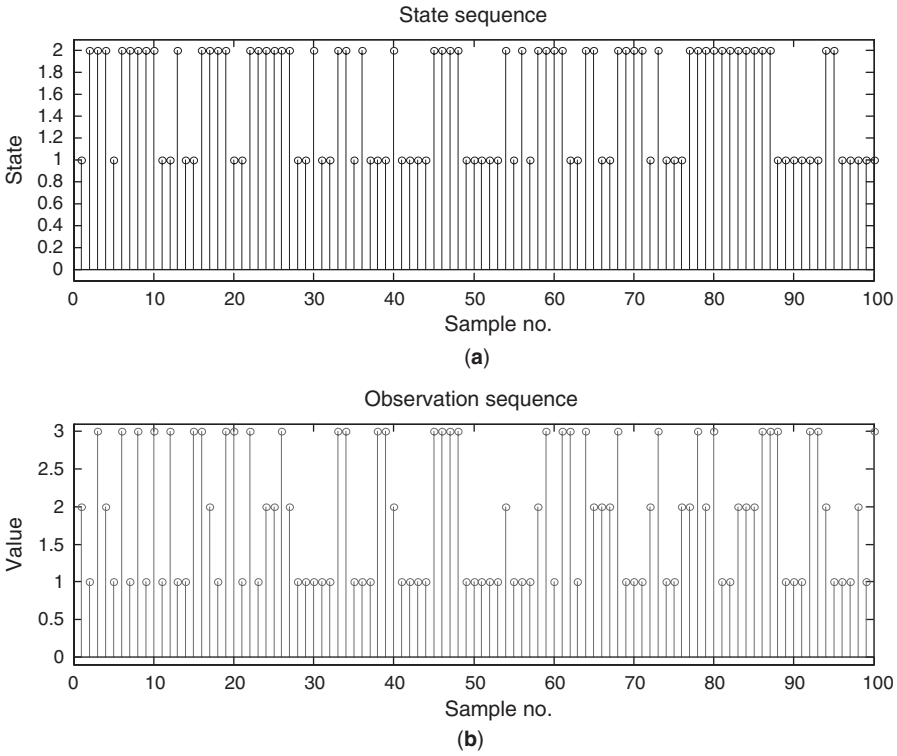


FIGURE 9.4 *HMM* realization of a discrete two-state Markov chain and observation: (a) Hidden state simulation. (b) Observation simulation. Note that both states/observations can assume only integer values governed by the transition and observation probabilities.

The transition probability is given by:

$$A = \begin{bmatrix} 0.6 & 0.4 \\ 0.3 & 0.7 \end{bmatrix}; \quad C = \begin{bmatrix} 0.50 & 0.25 & 0.25 \\ 0.35 & 0.25 & 0.40 \end{bmatrix}$$

The resulting directed graph is identical to that in Fig. 9.1. The transition probability implies that once a particular state is occupied, it is more than likely to remain in that state since $a_{11} = 0.6$ and $a_{22} = 0.7$ rather than transition to the other states, $a_{12} = 0.4$ and $a_{21} = 0.3$. So we expect the state transitions to essentially have longer time-steps with fewer transitions. The observation probability on the other hand seems almost equally likely to transition with $c_{11} = 0.5$ implying that when occupying state 1 it is most probable that the observation output will be 1 with $c_{23} = 0.4$ next, that is, if in state 2 then the output 3 is most likely.

Using *MATLAB* a simulation was performed for 100 samples with the results shown in Fig. 9.4. The state transitions are shown in *a* and appears to conform the intuition afforded by the transition probability, while the observations also follow

as well. The evaluation problem can be solved by estimating the corresponding total observation probability which is $\ln \Pr(Y_T) = -106$ and then making additional runs with various *HMM* for comparison. △△△

9.5 STATE ESTIMATION IN *HMM*: THE VITERBI TECHNIQUE

In this section we develop solutions to the state estimation problem for two cases of interest: (1) *individual* hidden state estimation, that is, the state estimate at a given time-step; and (2) *entire* sequence or "all" time-steps hidden state estimation problem. Both of these problems lead to reconstructing the entire sequence of hidden states strictly from the observations and *HMM*. Think of receiving a signal and being asked to retrieve or recognize the individual symbols from a coded sequence. Here the hidden states are the embedded sequence and the observations are the noisy digitized measurements.

9.5.1 Individual Hidden State Estimation

State estimation in *HMM* provides a methodology by which the hidden variables or states embedded within the hidden Markov model can be extracted from knowledge of the model parameters Σ and the noisy observations as illustrated in Fig. 9.3b. The basic problem to be solved is:

GIVEN the observation sequence, Y_T and *HMM* parameters Σ, *FIND* the "best" (*MAP*) estimate $\hat{x}_k(t)$ of the hidden state at time t based on the posterior distribution $\Pr(x_k(t)|Y_T)$, that is,

$$\hat{x}_k(t) = \arg\max_{x_k} \Pr(x_k(t)|Y_T) \quad \text{for } x_k(t) \Leftrightarrow x(t) = \mathcal{X}_k$$

The solution to this estimation problem is analogous to the forward recursion algorithm and incorporates the so-called backward algorithm, since it proceeds sequentially backwards in time (smoothing). The solution which follows is accomplished by decomposing or partitioning the total observation sequence into two sub-sequences: $Y_t = \{y(0), \ldots, y(t)\}$ and $Y_{t+1:T} := \{y(t+1), \ldots, y(T)\}$. To see this let us investigate the solution to the state estimation problem assuming uninformative priors

$$\Pr(x_k(t)|Y_T) = \frac{\Pr(Y_T, x_k(t))}{\Pr(Y_T)} \propto \Pr(Y_T, x_k(t)) \tag{9.15}$$

Partitioning Y_T as above we have, $Y_T = \{Y_t, Y_{t+1:T}\}$ which can be used to decompose the joint distribution as

$$\Pr(Y_T, x_k(t)) = \Pr(Y_t, Y_{t+1:T}, x_k(t)) = \Pr(Y_t, x_k(t)) \times \Pr(Y_{t+1:T}|Y_t, x_k(t))$$

Applying the Markov property along with the conditional independence properties of the *HMM*, we have

$$\Pr(Y_T, x_k(t)) = \Pr(Y_t, x_k(t)) \times \Pr(Y_{t+1:T}|x_k(t)) \tag{9.16}$$

Defining the *backward operator* as $\mathcal{B}_k(t) := \Pr(Y_{t+1:T}|x_k(t))$, then we can write the marginalization

$$\Pr(Y_{t+1:T}|x_k(t)) = \sum_{x_\ell(t+1)} \Pr(x_\ell(t+1), y(t+1), Y_{t+2:T}|x_k(t)) \tag{9.17}$$

and applying the Bayes' rule

$$\begin{aligned}\Pr(Y_{t+1:T}|x_k(t)) = &\sum_{x_\ell(t+1)} \Pr(Y_{t+2:T}|x_\ell(t+1), y(t+1), x_k(t)) \\ &\times \Pr(y(t+1), x_\ell(t+1)|x_k(t))\end{aligned} \tag{9.18}$$

Now using the Markovian independence properties of the *HMM* and expanding the last term using the Bayes' rule, we obtain

$$\begin{aligned}\Pr(Y_{t+1:T}|x_k(t)) = &\sum_{x_\ell(t+1)} \Pr(Y_{t+2:T}|x_\ell(t+1)) \times \Pr(y(t+1)|x_\ell(t+1)) \\ &\times \Pr(x_\ell(t+1)|x_k(t))\end{aligned} \tag{9.19}$$

Using the definition of the backward operator, we obtain the final *backward recursion* as:

$$\mathcal{B}_k(t) = \sum_\ell a_{k\ell} \Pr(y(t+1)|x_\ell(t+1)) \mathcal{B}_\ell(t+1) \quad \text{for } t = T-1, T-2 \ldots, 1, 0 \tag{9.20}$$

with $\mathcal{B}_k(0) = 1 \; \forall \; k$. This relation, when coupled with the forward operator can also be used to calculate the desired posterior probability for state estimation, since

$$\Pr(Y_T, x_k(t)) = \Pr(Y_t, x_k(t)) \times \mathcal{B}_k(t) = \mathcal{F}_k(t) \times \mathcal{B}_k(t) \tag{9.21}$$

Thus, we have by marginalization that the total observation probability can be estimated by

$$\Pr(Y_T) = \sum_k \mathcal{F}_k(T) \times \mathcal{B}_k(t) \tag{9.22}$$

and therefore the *posterior distribution* is given by:

$$\Pr(x_k(t)|Y_T) = \frac{\mathcal{F}_k(t) \times \mathcal{B}_k(t)}{\sum_k \mathcal{F}_k(T) \times \mathcal{B}_k(t)} \tag{9.23}$$

leading to the desired state estimate, $\hat{x}_{MAP}(t)$.

With this information available, we now have the solution to the *individual* hidden state estimation problem using the *backward recursion algorithm* as:

- Initialize: $\mathcal{B}_k(0) = 1 \; \forall \; k$;
- Recursion: $\mathcal{B}_k(t) = \sum_\ell a_{k\ell} \Pr(y(t+1)|x(t+1)\mathcal{B}_\ell(t+1)$;
- Termination: $\Pr(Y_T) = \sum_k \mathcal{F}_k(T) \times \mathcal{B}_k(t)$; and
- Posterior: $\Pr(x_k(t)|Y_T) = \frac{\mathcal{F}_k(t) \times \mathcal{B}_k(t)}{\Pr(Y_T)}$; and
- Estimation: $\hat{x}_k(t) = \arg\max_{x_k} \Pr(x_k(t)|Y_T)$.

Together the forward–backward recursions are the *key* ingredient to estimating the *HMM* parameters from noisy observation data as well which will be discussed subsequently, but first we consider extending the *MAP* state estimation for the individual state ($\hat{x}_k(t)$) to the problem of estimating the *entire* sequence of hidden states ($\hat{X}_T$) from the observation data, (Y_T).

An example follows to demonstrate the forward–backward recursion in estimating the posterior distribution.

Example 9.3

Suppose we have the discrete binary signal ($N_x = 2$) specified by: $\{x_1(t) = 1, x_2(t) = 2\}$ and the discrete observation specified by $\{y_1(t) = 1,\ y_2(t) = 2,\ y_3(t) = 3\}$. Here we attempt to "decode" the message from the observations, that is, consider the binary state sequence generated by the *HMM* along with the corresponding the observations and we wish to extract the coded state sequence at each time-step using the forward–backward approach discussed above.

We apply *MATLAB* (**hmmdecode**) to perform the estimation using the forward–backward approach for the 100 observation samples. Using the synthesized output data and corresponding transition and observation probability matrices, we can estimate the corresponding posterior distribution for each individual state, $\Pr(x_k(t)|Y_t)$ with the results shown in Fig. 9.5. The combined state transitions are shown in *a* and the posterior state probabilities in *b* and *c*, respectively. We can see that the estimated probabilities "match" the states reasonably well with the state transitioning according to the estimated posterior at each time-step, that is, when the *HMM* is in state 1, the posterior probability is high relative to that of state 2 and visa-versa. This completes the decoding example. Next we consider estimating the entire state sequence.

△△△

9.5.2 Entire Hidden State Sequence Estimation

The maximum *a posteriori* estimation of the state at time t using the forward–backward recursion algorithm above can be extended to reconstruct the entire hidden state sequence which provides a more meaningful solution when attempting to extract a critical coded message from a hostile environment or accurately extracting a DNA sequence for forensic analysis. Unfortunately, estimating the *individually* "most likely" states as in the previous subsection does not imply that the *entire* sequence is estimated with minimal probability of error. Therefore, the state estimation problem must be based on jointly estimating "all" states in the sequence to obtain the optimal solution. The individual state estimates at each time-step of the forward–backward algorithm minimize the error probabilities of individual states maximizing the expected number of correctly estimated states [16]. However, we are interested in estimating the *entire* state sequence, that is, we would like to solve the following problem:

GIVEN the observation sequence, Y_T and *HMM* parameters Σ, *FIND* the "best" (*MAP*) estimate the sequence $\hat{X}_T$ where $\hat{X}_T = \{\hat{x}_k(0), \ldots, \hat{x}_k(T)\}$ of the entire hidden

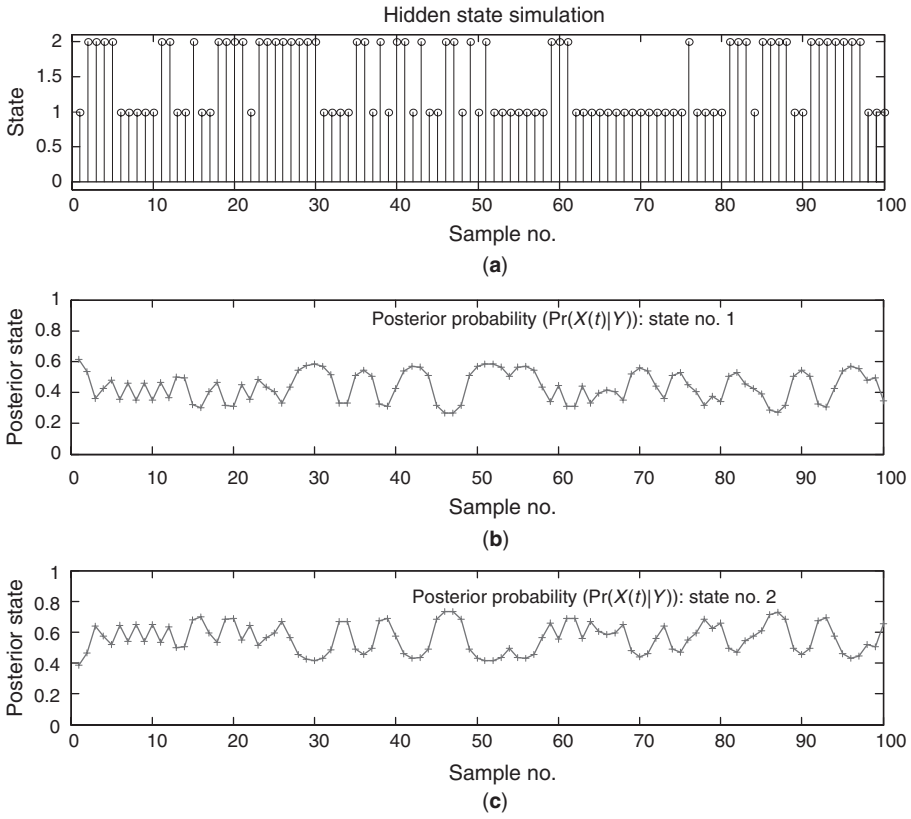


FIGURE 9.5 *HMM* realization of a discrete two-state Markov chain and observation: (a) Hidden state sequence realization. (b) State 1 posterior probability estimate, $\hat{\Pr}(x_1(t)\,|\,Y_t)$. (c) State 2 posterior probability estimate, $\hat{\Pr}(x_2(t)\,|\,Y_t)$.

state sequence from time-step 0 to time-step T based on the posterior distribution $\Pr(X_T|Y_T)$, that is,

$$\hat{X}_T = \arg\max_{X_T} \Pr(X_T|Y_T)$$

Since we are seeking a sequential solution to this problem, we must track the estimate at time-step t. Following the development in [16], for each $x(t)$ a partial sequence of length $t+1$ is defined for each possible state; therefore, there are N_x-partial sequences for each t. An alternative is to use the joint distribution, since the observation sequence is fixed in length Y_T, that is, we require

$$\hat{X}_{t-1} = \arg\max_{X_{t-1}} \Pr(X_{t-1}, x(t), Y_t) \quad \text{for } x(t) \text{ endpoint} \tag{9.24}$$

At each time-step, the maximal path (X_{t-1}) problem terminating in $x(t)$ given Y_t is transformed into the maximization problem of finding the best path ending in $x(t+1)$

given Y_{t+1}. This follows directly by applying the chain rule to the joint probability distribution

$$\begin{aligned}\Pr(X_{t+1}, Y_{t+1}) &= \Pr(x(t+1), X_t, y(t+1), Y_t) \\ &= \Pr(x(t+1), y(t+1)|X_t, Y_t) \times \Pr(X_t, Y_t) \end{aligned} \tag{9.25}$$

applying Bayes' rule along with the conditional independence properties of the chain gives

$$\Pr(X_{t+1}, Y_{t+1}) = \Pr(y(t+1)|x(t+1)) \times \Pr(x(t+1)|x(t)) \times \Pr(X_t, Y_t) \tag{9.26}$$

Recursively maximizing the probability gives

$$\begin{aligned}\max_{X_t} \Pr(X_{t+1}, Y_{t+1}) &= \max_{X_t}\{\Pr(y(t+1)|x(t+1)) \times \Pr(x(t+1)|x(t)) \times \Pr(X_t, Y_t)\} \\ &= \Pr(y(t+1)|x(t+1)) \times \max_{X_t}\{\Pr(x(t+1)|x(t)) \times \Pr(X_t, Y_t)\} \\ &= \Pr(y(t+1)|x(t+1)) \times \max_{x(t)} \left\{\Pr(x(t+1)|x(t)) \right. \\ &\qquad \left. \times \max_{X_{t-1}} \{\Pr(X_t, Y_t)\}\right\}\end{aligned}$$

Now this gives us a recursion with $\mathcal{V}(x(t)) := \max_{X_{t-1}}\{\Pr(X_t, Y_t)\}$

$$\mathcal{V}(x(t+1)) = \Pr(y(t+1)|x(t+1)) \times \arg\max_{x(t)}\{\Pr(x(t+1)|x(t)) \times \mathcal{V}(x(t))\} \tag{9.27}$$

Defining the smoothing variable as

$$\mathcal{U}(x(t)) := \max_{X_{t-1}}\{\Pr(x(t+1)|x(t)) \times \mathcal{V}(x(t))\} \tag{9.28}$$

enables us to construct the entire *state (sequence) estimation* or equivalently the *Viterbi algorithm* as [16]:

- Initialize:

$$\begin{aligned}\mathcal{V}(x(0)) &= \Pr(x(0)), \Pr(y(0)|x(0)) \\ \mathcal{U}(x(0)) &= 0 \quad \text{for } x(0) = 1, \ldots, N_x\end{aligned}$$

- Recursion:

$$\begin{aligned}\mathcal{V}(x(t)) &= \Pr(y(t)|x(t)) \times \max_{x(t-1)}\{\Pr(x(t)|x(t-1)) \times \mathcal{V}(x(t-1))\} \\ \mathcal{U}(x(t)) &= \max_{x(t-1)}\{\Pr(x(t)|x(t-1)) \times \mathcal{V}(x(t-1))\} \quad \text{for } x(t) = 1, \ldots, N_x; \\ t &= 2, \ldots, T\end{aligned}$$

- Termination:

$$P = \max_{x(T)}\{\mathcal{V}(x(T))\}$$
$$\hat{x}(T|T) = \arg\max_{x(T)}\{\mathcal{V}(x(T))\}$$

- Smoothing: $\hat{x}(t|T) = \mathcal{U}(\hat{x}(t+1)|T))$ for $t = T-1, T-2, \ldots, 0$

The Viterbi algorithm uses these recursions and smoothing relations to estimate the "optimal path" and has on the same order of operations as the forward algorithm discussed in the previous section. It has proved to be an extremely popular and robust algorithm to perform decoding. Consider the following example of a path estimate.

Example 9.4

Using the discrete binary signal and observation of the previous example we would like to "decode" the entire message from the observations, that is, we wish to extract the "entire" coded sequence at each time-step using the Viterbi approach discussed above.

We apply *MATLAB* (**hmmviterbi**) to perform the optimal entire state sequence (path) estimation using the Viterbi algorithm for the 100 observation samples. Using the synthesized output data and corresponding transition and observation probability matrices, as before we obtain the estimation results shown in Fig. 9.6. Here we can observe the path (dark solid line) which corresponds to the sequence estimation. Note that it is a simple path but contains most of the states and leads directly to the desired result. The matching capability of this approach is captured by estimating the percentage of the time that the actual sequence agrees with the estimated. For this simulation, the estimated matches the actual sequence 52% of the time. △△△

This completes the state sequence estimation example, next we consider estimating the model parameters.

9.6 PARAMETER ESTIMATION IN *HMM*: THE EM/BAUM-WELCH TECHNIQUE

The most challenging problem in *HMM* is the development of the model in the first place. Just as in dynamic (physical) systems theory [17, 18], the system identification/parameter estimation problem is still a highly researched problem especially for nonlinear systems. The basic estimation problem consists of two major issues: (1) estimation of the underlying internal structure (interconnections, state assignments, etc.); and (2) *HMM* parameter estimation consisting of the transition and observation probabilities and initial conditions assuming that the *internal structure* of (1) is *known a priori*.

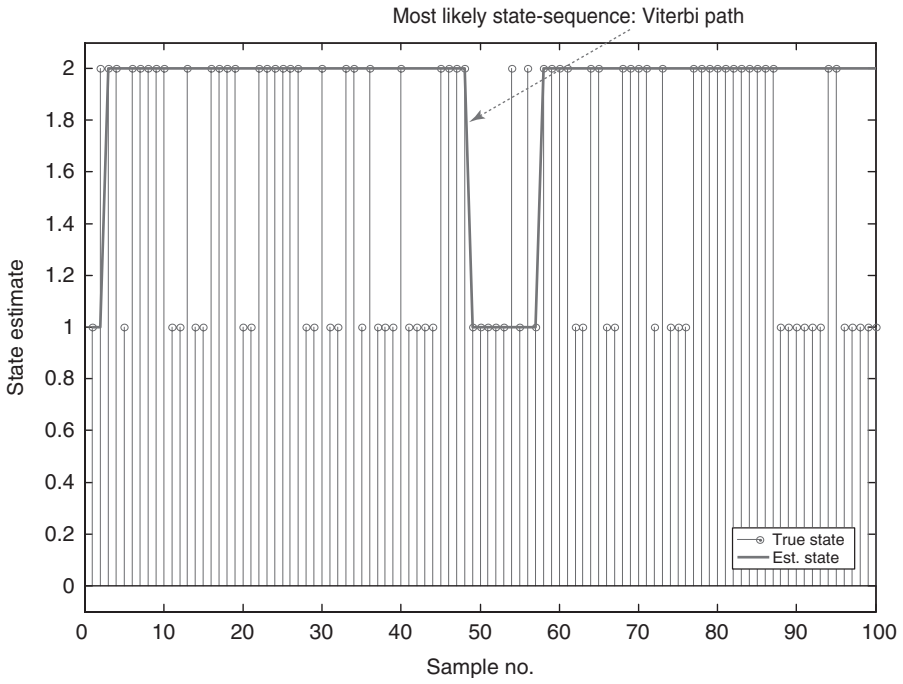


FIGURE 9.6 *HMM* realization of a discrete two-state Markov chain sequence and the results of the *Viterbi path* estimation with an 52% match (estimated-to-actual states).

For dynamic physical systems, the identification of the internal structural model usually evolves from first principles where relations governing the phenomenology are assembled. For non-physical systems, parametric models (with interconnections) are assumed (e.g., *ARMA*) and then the solution to the parameter estimation problem follows [19]. *HMM* are very similar from this perspective. Their structure is developed from an internal probabilistic representation that is application driven. For instance, the well-known problem in signal processing of recovering a transmitted random telegraph signal from noisy observations is representative. Here the problem is to "decode" the signal into a sequence of zeros or ones with the probability distribution of the transitions (zero-to-one) assumed known (Poisson). A *HMM* can be structurally developed to model this problem quite easily [2]. Another example is the decoding of DNA strings for forensic analysis. Here the same modeling principle applies to develop the internal structural model [5]. In any case developing the internal model is usually the task of the phenomenologist, whether a physical or non-physical system, and the next step is to "fit" the parameters of this structure to the model—the primary focus of this section. Thus, we discuss the development of the parameter estimation techniques applied to estimate the parameters of a *HMM*. Here we assume the number of states and measurements are known along with the internal structure and the problem becomes a matter of "fitting" these well-defined parameters (transition

probabilities, observation probabilities and initial conditions) to the known model internal structure.

Parameter estimation for *HMM* has been a difficult and challenging problem, especially in an on-line environment [1]. The original efforts of Baum-Welch [20] have led to the general expectation-maximization (*EM*) algorithm (see Chapter 2) which is a very powerful iterative approach using likelihood estimation techniques to solve this problem [21, 22]. Here we briefly outline the iterative approach and then show how the algorithm is a special case of *EM*.

The basic *HMM parameter estimation problem* is (see Fig. 9.3c):

GIVEN a set or J-sets[4] of observation sequences, $\{Y_t(j)\}$; $j = 1, \ldots, J$ along with the underlying *HMM* internal structure Σ, **FIND** the "best" (*MAP*) estimate of the underlying parameters, $\hat{\Theta}_{MAP} := \{a_{mn}, c_{k\ell}, P(0)\}$ maximizing the posterior distribution, $\Pr(\Theta|Y_t)$.

We will discuss this problem in two parts: (1) state sequence is *known a priori*; and (2) state sequence is *unknown* [5].

9.6.1 Parameter Estimation with State Sequence Known

When the state sequence is "known" *a priori*, then the parameter estimation problem is much simpler as in the case in physical systems for the design of optimal inputs for system identification [18].

We define the following posterior probability

$$\mathcal{P}_{mn}(t, t+1) := \Pr(x_m(t), x_n(t+1)|Y_T, \Theta)$$

This posterior is the probability of the joint event that a path (state sequence) passes through state m at time-step t and through state n at $t+1$ *and* the *HMM* generates a sequence of observations Y_T given the model parameters, Θ.

To analyze this probability further, we apply Bayes' rule and then partition the data as before

$$\begin{aligned}\mathcal{P}_{mn}(t, t+1) &= \frac{\Pr(x_m(t), x_n(t+1), Y_T|\Theta)}{\Pr(Y_T|\Theta)} \\ &= \frac{\Pr(x_m(t), x_n(t+1), Y_t, Y_{t+1:T}|\Theta)}{\Pr(Y_T|\Theta)} \end{aligned} \tag{9.29}$$

now applying Bayes' rule to the numerator results in

$$\begin{aligned}\Pr(x_m(t), x_n(t+1), Y_t, Y_{t+1:T}|\Theta) &= \Pr(x_n(t+1), Y_{t+1:T}|Y_t, x_m(t), \Theta) \\ &\quad \times \Pr(Y_t, x_m(t)|\Theta) \end{aligned} \tag{9.30}$$

[4] These sets are called training sets a term that evolves from the classification/neural-net technical area.

The last term is just $\mathcal{F}_m(t)$, the forward operator (with the parameter set Θ given). Now concentrating on the remaining term of this expression, we extract the $y(t+1)$ from the data term and apply Bayes' rule again to obtain

$$\begin{aligned}&\Pr(x_n(t+1), y(t+1), Y_{t+2:T}|Y_t, x_m(t), \Theta)\\ &\quad = \Pr(Y_{t+2:T}|x_n(t+1), y(t+1), Y_t, x_m(t), \Theta)\\ &\qquad \times \Pr(x_n(t+1), y(t+1)|Y_t, x_m(t), \Theta)\end{aligned} \tag{9.31}$$

where the last term above decomposes further to

$$\begin{aligned}\Pr(x_n(t+1), y(t+1)|Y_t, x_m(t), \Theta) \;=\; &\Pr(y(t+1)|x_n(t+1), Y_t, x_m(t), \Theta)\\ &\times \Pr(x_n(t+1)|x_m(t), Y_t, \Theta)\end{aligned} \tag{9.32}$$

enabling us to simplify each of these terms individually to yield:

$$\begin{aligned}\Pr(Y_{t+2:T}|x_n(t+1), y(t+1), Y_t, x_m(t), \Theta) \;&\rightarrow\; \Pr(Y_{t+2:T}|x_n(t+1), \Theta)\\ \Pr(x_n(t+1), y(t+1)|Y_t, x_m(t), \Theta) \;&\rightarrow\; \Pr(y(t+1)|x_n(t+1), \Theta)\\ &\quad \times \Pr(x_n(t+1)|x_m(t), \Theta)\end{aligned} \tag{9.33}$$

and therefore we obtain the expression:

$$\begin{aligned}\mathcal{P}_{mn}(t, t+1) \;=\; &\Pr(Y_t, x_m(t)) \times \Pr(x_n(t+1)|x_m(t), \Theta)\\ &\times \Pr(y(t+1)|x_n(t+1), \Theta) \times \Pr(Y_{t+2:T}|x_n(t+1), \Theta)/\Pr(Y_T|\Theta)\end{aligned}$$

Finally, substituting for the known parameters, we have the desired result

$$\mathcal{P}_{mn}(t, t+1) = \frac{\mathcal{F}_m(t) \times a_{mn} \times c_{kn} \times \mathcal{B}_n(t+1)}{\Pr(Y_T|\Theta)} \tag{9.34}$$

where $\mathcal{F}_k(t)$ encompasses the *past* history ending at time t and state m while $\mathcal{B}_n(t+1)$ accounts for the path's *future* which at time $t+1$ is at state n evolving until the end. The product term $(a_{mn} \times c_{kn})$ takes into account the current activity at t with discrete observation $y_k(t+1) \rightarrow y(t+1)$.

With this term determined, we then define the *posterior distribution* from Eq. 9.23 as:

$$\mathcal{O}_m(t) := \Pr(x_m(t)|Y_T) \tag{9.35}$$

which is a probability of the joint event that a path passes through state m at time-step t *and* the *HMM* generates a sequence of observations Y_T given the model parameters, Θ.

Note that both probabilities are related, since one can be obtained through marginalization of the other

$$\mathcal{O}_m(t) = \sum_{n=1}^{N_x} \mathcal{P}_{mn}(t, t+1) \tag{9.36}$$

Summing both these quantities "across time" enables us to obtain the *expected number of times* in state m and the expected number of transitions *away* from state m for Y (see [15] for more details)

$$\sum_{t=1}^{T-1} \mathcal{O}_m(t) \tag{9.37}$$

Similarly, the expected number of *transitions* from state m to state n for Y is given by

$$\sum_{t=1}^{T-1} \mathcal{P}_{mn}(t) \tag{9.38}$$

Thus, using these expectations and counting estimation of probabilities [5], we are able to obtain the *Baum-Welch* estimates:

$$\begin{aligned} \hat{P}_m(0) &= \mathcal{O}_m(1) \\ \hat{a}_{mn} &= \frac{\sum_{t=1}^{T-1} \mathcal{P}_{mn}(t)}{\sum_{t=1}^{T-1} \mathcal{O}_m(t)} \\ \hat{c}_{kn} &= \frac{\sum_{t=1}^{T} \mathcal{P}_{mn}(t)}{\sum_{t=1}^{T} \mathcal{O}_m(t)} \quad \text{such that } y(t) = \mathcal{Y}_k \end{aligned} \tag{9.39}$$

where

$P_m(0)$ is the expected number of times in state, $x_n(t)$ at $t = 1$;

a_{mn} is the expected number of transitions from state, $x_m(t)$ to state $x_n(t)$ over the expected transitions in state $x_m(t)$; and

c_{kn} is the expected number of times in state, $x_n(t)$ and observing $y_k(t)$ over the expected number of times in state $x_n(t)$.

Thus, when all of the paths are known (this case), then it is possible to *count* the number of times each particular transition or output observation is applied in a set of training data. It has been shown that counting functions, say $N_{mn}(x(t))$ for the state transitions and $N_{kn}(y(t))$ for the output observations provide maximum likelihood estimates for the desired model parameters [5], such that

$$\hat{a}_{mn} = \frac{N_{mn}(x(t))}{\sum_n N_{mn}(x(t))} \quad \text{and} \quad \hat{c}_{kn} = \frac{N_{kn}(y(t))}{\sum_n N_{kn}(y(t))} \tag{9.40}$$

Next we consider the unknown path case and combine the above results to establish the algorithm.

9.6.2 Parameter Estimation with State Sequence Unknown

In this section we consider the case where the state sequence is "not known" [5] and must be determined using the current parameter estimates available. When the paths are unknown for training sequences, a closed-form equation is nonexistent

and therefore some type of iterative approach must be applied (e.g., *EM* [22–30]). The *EM/Baum-Welch* approach precisely solves the *HMM* parameter estimation problem in an iterative manner. It first estimates the counting functions, $N_{mn}(x(t))$ for states and $N_{kn}(y(t))$ for observations by considering possible paths for the training sequences using current model parameters Θ and then calculates the new estimates using Eq. 9.40. The algorithm continues to iterate until the log-likelihood function, $\ln \Pr(Y_T|\Theta)$ no longer increases with each iteration. Baum [20] has shown that the overall log-likelihood increases with each iteration indicating convergence to a local maximum.

More precisely, this technique estimates the counting functions, $N_{mn}(x(t))$ and $N_{kn}(y(t))$ as the expected number of times each transition or output is utilized from the *given* training sequences. It also uses the identical forward/backward operators as before (see Section 9.5) using the posterior probability $\mathcal{P}_{mn}(t, t+1)$ of Eq. 9.34. From this relation, we can derive the *expected number of times* that a_{mn} is used by summing over all possible positions and over all training sequences $Y_T(j)$; $j = 1, \ldots, J$. We can also use the training sequences to derive the *expected number of times the observation occurs* to obtain:

$$N_{mn}(x(t)) = \sum_j \frac{1}{\Pr(Y_T|\Theta)} \sum_t \mathcal{F}_m(t,j) \times a_{mn} \times c_{kn} \times \mathcal{B}_n(t+1,j)$$

$$N_{k\ell}(y(t)) = \sum_j \frac{1}{\Pr(Y_T|\Theta)} \sum_t \mathcal{F}_k(t,j) \times \mathcal{B}_k(t,j) \tag{9.41}$$

Once these expectations are estimated, the model parameters are updated as above and these new estimates are used in the counting functions. We summarize the *EM/Baum-Welch* algorithm as:

- Initialization: $\hat{\Theta}_0$, $N_{mn}(x(t))$ and $N_{k\ell}(y(t))$;
- Forward/Backward Recursions: $\mathcal{F}_k(t,j)$ and $\mathcal{B}_k(t,j)$ of Eqs. 9.12 and 9.20;
- Counting functions: $N_{mn}(x(t))$ and $N_{k\ell}(y(t))$ of Eq. 9.41;
- Parameter Estimation: $\hat{\Theta} = \{a_{mn}, c_{kn}, P(x(0))\}$ of Eq. 9.40;
- Likelihood: $\Pr(Y_T|\hat{\Theta})$; and
- Termination: $\Pr(Y_T|\hat{\Theta}) < \tau$ for τ a threshold.

This completes the algorithm. It should be noted that an alternative approach to searching over all paths is to use the *Viterbi paths* providing the most probable paths for all of the training sequences. However, this approach does not maximize the true likelihood. It is known that Viterbi training does *not* perform as well as the Baum-Welch, but it is still popular when applied to decoding problems.

There are also a number of techniques that practitioners use to enhance numerical performance and convergence of this technique. These include: (1) logarithmic transformation of the product probabilities to create sums [5]; (2) scaling both forward and backward operators [10]; (3) initial conditions; and (4) training data issues [23]. In closing, we note that the *Baum-Welch* algorithm is just a special case of the *EM*

algorithm of Sec. 2.3. That is, the E-step of the *EM* algorithm is given by [15]:

$$\begin{aligned}\text{E-step: } Q(\Theta, \hat{\Theta}_{i-1}) &= \sum_x \ln \Pr(x|Y_T, \Theta) \times \Pr(x|Y_T, \hat{\Theta}_{i-1}) \\ &= \sum_x \ln \hat{P}(x(0)) \Pr(Y_T, x(t)|\hat{\Theta}) + \sum_x \left(\sum_{t=1}^{T} \ln \hat{a}_{mn}(t-1, t) \right) \\ &\quad \times \Pr(Y_T, x(t)|\hat{\Theta}) + \sum_x \left(\sum_{t=1}^{T} \ln \hat{c}_{kn}(t, t) \right) \times \Pr(Y_T, x(t)|\hat{\Theta})\end{aligned} \tag{9.42}$$

where $\Pr(Y_T, x(t)|\hat{\Theta}) = P(x(0)) \prod_{t=0}^{T} \hat{c}_{kn}(t, t) \times \prod_{t=1}^{T} \hat{a}_{mn}(t, t+1)$. Optimizing these terms leads precisely to the expressions in Eq. 9.39 (see [10] or [15] for details). Thus, the E-step of the *EM* consists of estimating the required expectations using the forward/backward recursions which completely determines $Q(\Theta, \hat{\Theta})$ and the maximum (M-step) consists of substituting these terms into the corresponding likelihood.

Example 9.5

Again using discrete binary signal and the discrete observation of the previous examples, we perform the parameter estimation of the transition and observation probabilities, first using, the "known" (actual) state sequence and then generating an ensemble of training sequences ($N = 25$) to perform the *EM*/*Baum-Welch* algorithm (**hmmtrain**) with a maximum of 500 iterations and a error tolerance of 1×10^{-4}. Here we also use the *MATLAB* maximum likelihood estimation with the "known" state sequence (**hmmestimate**) as well to compare performance. The resulting parameter estimates and percentage errors are:

EM/BAUM-WELCH PARAMETER ESTIMATES

$$A_{TRU} = \begin{bmatrix} 0.6 & 0.4 \\ 0.3 & 0.7 \end{bmatrix}; \qquad C_{TRU} = \begin{bmatrix} 0.50 & 0.25 & 0.25 \\ 0.35 & 0.25 & 0.40 \end{bmatrix}$$

$$\hat{A}_{BW} = \begin{bmatrix} 0.62 & 0.38 \\ 0.30 & 0.7 \end{bmatrix}; \qquad A_{\%ERR} = \begin{bmatrix} 3 & 4 \\ 0 & 0 \end{bmatrix}$$

$$\hat{C}_{BW} = \begin{bmatrix} 0.51 & 0.29 & 0.19 \\ 0.34 & 0.22 & 0.44 \end{bmatrix}; \qquad C_{\%ERR} = \begin{bmatrix} 3 & 18 & 23 \\ 3 & 13 & 11 \end{bmatrix}$$

Thus, the parameter estimates are quite reasonable under these conditions. Note the initial probability matrices are automatically established by this implementation in *MATLAB*; however, it is possible to alter them if desired.

MAXIMUM LIKELIHOOD PARAMETER ESTIMATES

$$A_{TRU} = \begin{bmatrix} 0.6 & 0.4 \\ 0.3 & 0.7 \end{bmatrix}; \qquad C_{TRU} = \begin{bmatrix} 0.50 & 0.25 & 0.25 \\ 0.35 & 0.25 & 0.40 \end{bmatrix}$$

$$\hat{A}_{ML} = \begin{bmatrix} 0.67 & 0.33 \\ 0.36 & 0.64 \end{bmatrix}; \qquad A_{\%ERR} = \begin{bmatrix} 12 & 18 \\ 21 & 9 \end{bmatrix}$$

$$\hat{C}_{ML} = \begin{bmatrix} 0.45 & 0.23 & 0.32 \\ 0.47 & 0.19 & 0.34 \end{bmatrix}; \qquad C_{\%ERR} = \begin{bmatrix} 9 & 9 & 28 \\ 34 & 23 & 15 \end{bmatrix}$$

Again the estimates appear quite reasonable. Longer sequences can be employed to improve the estimates even further. We see that there is a distinct advantage when the state sequence is *known a priori*, because the training sequences are *not* required. The Viterbi initialization was also executed on this data, but it did not perform near as well as the Baum-Welch technique. △△△

This completes the section, next we consider a case study.

9.7 CASE STUDY: TIME-REVERSAL DECODING

In this section, we consider applying the Viterbi algorithm to decode a message transmitted through a hostile environment with reverberations along with the processor and decoding algorithm. Acoustic time-reversal (*T/R*) communications is an application area motivated by the recent theoretical advances in *T/R* theory [30]. Although perceived by many in signal processing as simply an application of matched-filter theory, a *T/R* receiver offers an interesting solution to the communications problem for a highly reverberant channel. This case study briefly describes an acoustic communications experiment of data gathered in air and its associated signal processing. The experiment is developed to evaluate the performance of a point-to-point *T/R* receiver designed to extract a transmitted code information sequence propagating in a hostile, highly reverberant environment. These results are merely used to "synthesize" a *HMM* based on the raw/quantized acoustic measurements and then used to extract the transition and observation probabilities for simulation and evaluation. Even though this case study is based on real data, it is only chosen to illustrate the application of *HMM* techniques *after* data is simulated through the *HMM* process (evaluation).

From a signal processing perspective, *T/R* processing appears to be an application of matched-filtering in which the output signal-to-noise ratio (*SNR*) is maximized. This *T/R* replicant is then cross-correlated with the noisy received signal to produce the optimal filtered output [31]. However, it becomes more complicated in the spatio-temporal case in which the optimal matched-filter must not only match the transmitted temporal function, but also the corresponding spatio-temporal channel medium impulse response or so-called Green's function. It has been shown that time-reversal techniques are applicable to spatio-temporal phenomena that satisfy a wave-type equation possessing the time reversal invariance property [30]. Thus, time-reversal is the dynamic broadband analog of the well-known phase conjugate mirror used to focus narrowband monochromatic waves. It represents the "optimal" spatio-temporal matched filter in the sense of maximizing the output SNR. It is essentially a technique, which can be used to "remove" the aberrations created by an

inhomogeneous or random channel. In communications, the *T/R* receiver can overcome the inherent noise created by the medium providing the enhancement required to extract the transmitted information sequence. Here we ignore the array aspects of *T/R* by considering only point-to-point communications. In this case study the realization of a *T/R* receiver is briefly discussed and applied to a noisy microphone measurement in a hostile environment. It is then used to estimate the required transition and observation matrices for eventual synthesis/analysis.

For time-reversal, the matched-filter in additive white noise is identical to that posed above with a "known" Green's function of the medium replacing the known signal replicant [31]. The Green's function, $g(r, r_o; t)$, is the result of a point-to-point communication link between a station (source) at r_o to a master station (receiver) at r. In this case, the matched-filter solution is again found by maximizing, SNR_{out}, leading to the solution that is satisfied with equality at some time T. If the resulting filter response is, $f(t)$, then the solution is given by

$$f(t) = g(r, r_o; T - t) \tag{9.43}$$

Thus, for *T/R*, the optimal matched-filter solution is the time-reversed Green's function from the link station-to-master station (source-to-receiver) or visa versa. Comparing these results with the standard matched-filter solution found in the literature, the Green's function of the channel is time-reversed rather than the transmitted replicant signal as in radar or sonar. Note that since *T/R* theory requires reciprocity [30], the result of Eq. 9.43 is valid for both transmission and reception, that is, $g(r, r_o; T - t) \leftrightarrow g(r_o, r; T - t)$. Note also that if an array is included to sample the spatial field or transmit a wave, then these results include the focus at link station (source) position, r_o, yielding the optimal, spatio-temporal matched-filter solution, $g(r_\ell, r_o; T - t)$ at sensor position, r_ℓ.

So we see that in transmitting a coded signal (state sequence) through a disruptive medium the distorting effects can be mitigated by time-reversing the estimated media Green's function and creating an effective receiver. The details of this mechanism are discussed in [31–33] and is beyond the scope of this case study. Here we just describe one of the variety of receiver types that can be used, once the Green's function is estimated from pilot signals transmitted from transmitter (speaker) to receiver (microphone) producing $\hat{g}(r, r_o; T - t)$.

With the estimated Green's function or impulse response available, we choose to apply time-reversal processing on reception [31] to the noisy received data, $y(t) = g(r; t) * i(t)$ with $i(t)$ the coded information (state) sequence. On reception, the estimated Green's function is reversed and convolved with the receiver input to give

$$R(t) = z(t) * \hat{g}(r; -t) = g(r; t) * i(t) * \hat{g}(r; -t) = C_{g\hat{g}}(t) * i(t) \tag{9.44}$$

where $C_{g\hat{g}}$ is the estimated autocorrelation function of the medium possessing all of the reflection and scattering information—but modified for code signal enhancement. We show a typical *T/R* receiver output that was used to "synthesize" a discrete state and output sequence for transition and observation probability estimates. We show the

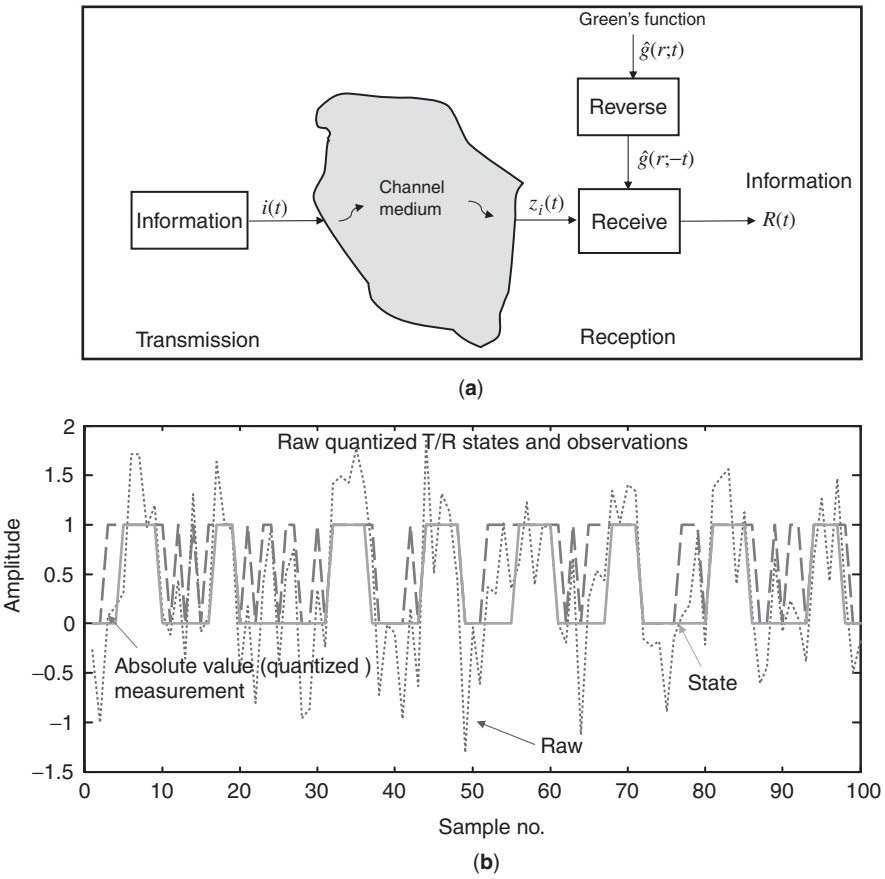


FIGURE 9.7 *T/R* processor acoustic microphone data: (a) *T/R* receiver structure. (b) Raw measurement data, synthesized observation and state data input for *HMM* parameter estimation.

receiver structure in Fig. 9.7a where the time-reversed Green's function is convolved with the received data and then quantized to recover the code. Actual *T/R* processed data is shown in *b* along with quantized state and observation sequence extracted for illustrative purposes and eventual application of the *HMM* techniques.

The results of processing these quantized sequences using the *EM*/*Baum-Welch* algorithm are:

EM/BAUM-WELCH PARAMETER ESTIMATES

$$\hat{A}_{BW} = \begin{bmatrix} 0.42 & 0.58 \\ 0.28 & 0.72 \end{bmatrix}; \qquad \hat{C}_{BW} = \begin{bmatrix} 1 & 0 \\ 0.22 & 0.78 \end{bmatrix}$$

Next we used these extracted probability matrices to synthesize "realistic" *T/R* data for *HMM* processing, the results are shown in Fig. 9.8. With this available we proceed

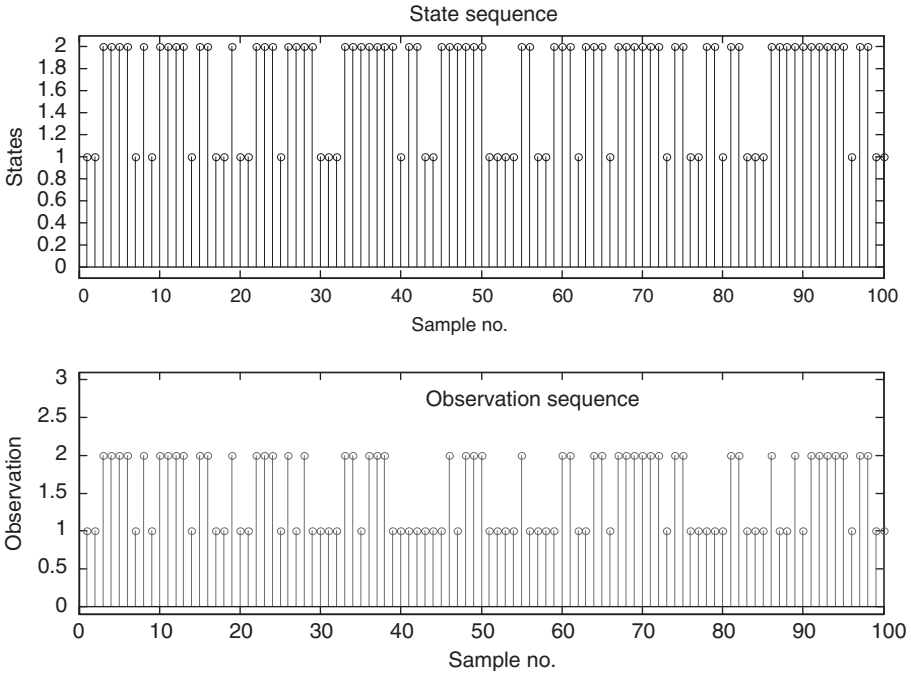


FIGURE 9.8 *HMM* realization of the *T/R* discrete Markov chain and observation: (a) Hidden state sequence realization. (b) Observation realization.

as before and estimate the individual states as depicted by the posterior probabilities in Fig. 9.9. These results are quite reasonable as can be observed by comparing samples with the aligned probability functions at each time-step. Next the entire state sequence was estimated using the "most likely" Viterbi approach and the results are illustrated in Fig. 9.10, where both the actual (synthesized time-reversed) state sequences are shown along with the Viterbi result superimposed. The agreement is quite good with an 85% matching (in-time) of the actual with the estimated states.

With these synthesized probability matrices, we performed the parameter estimation approach as before to give

EM/BAUM-WELCH PARAMETER ESTIMATES

$$A_{TRU} = \begin{bmatrix} 0.42 & 0.58 \\ 0.28 & 0.72 \end{bmatrix}; \qquad C_{TRU} = \begin{bmatrix} 1 & 0 \\ 0.22 & 0.78 \end{bmatrix}$$

$$\hat{A}_{BW} = \begin{bmatrix} 0.45 & 0.55 \\ 0.27 & 0.73 \end{bmatrix}; \qquad A_{\%ERR} = \begin{bmatrix} 6 & 5 \\ 4 & 2 \end{bmatrix}$$

$$\hat{C}_{BW} = \begin{bmatrix} 1 & 0 \\ 0.24 & 0.76 \end{bmatrix}; \qquad C_{\%ERR} = \begin{bmatrix} 0 & 0 \\ 8 & 2 \end{bmatrix}$$

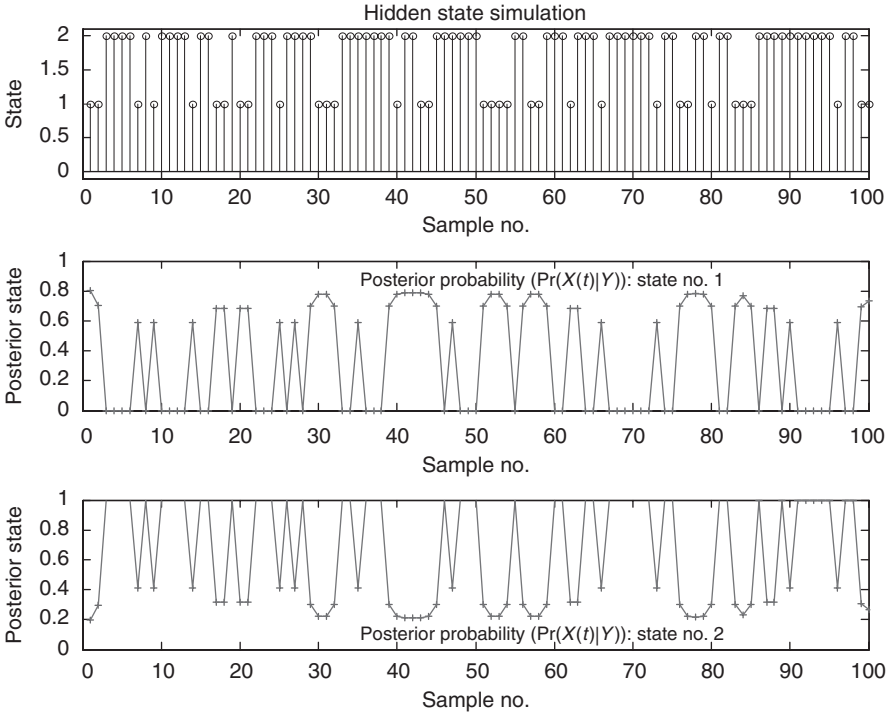


FIGURE 9.9 *HMM* estimation of the *T/R* discrete two-state Markov chain and observation: (a) Hidden state sequence realization. (b) State 1 posterior probability estimate, $\hat{\Pr}(x_1(t) \mid Y_t)$. (c) State 2 posterior probability estimate, $\hat{\Pr}(x_2(t) \mid Y_t)$.

Thus, the parameter estimates are quite reasonable under these conditions. Note the initial probability matrices are automatically established by this implementation in *MATLAB*; however, it is possible to alter them if desired.

MAXIMUM LIKELIHOOD PARAMETER ESTIMATES

$$A_{TRU} = \begin{bmatrix} 0.42 & 0.58 \\ 0.28 & 0.72 \end{bmatrix}; \qquad C_{TRU} = \begin{bmatrix} 1 & 0 \\ 0.22 & 0.78 \end{bmatrix}$$

$$\hat{A}_{ML} = \begin{bmatrix} 0.42 & 0.58 \\ 0.29 & 0.71 \end{bmatrix}; \qquad A_{\%ERR} = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}$$

$$\hat{C}_{ML} = \begin{bmatrix} 1 & 0 \\ 0.24 & 0.76 \end{bmatrix}; \qquad C_{\%ERR} = \begin{bmatrix} 0 & 0 \\ 11 & 3 \end{bmatrix}$$

The estimates are quite reasonable. We see that there is a distinct advantage when the state sequence is known *a priori*, then the training sequences are not required.

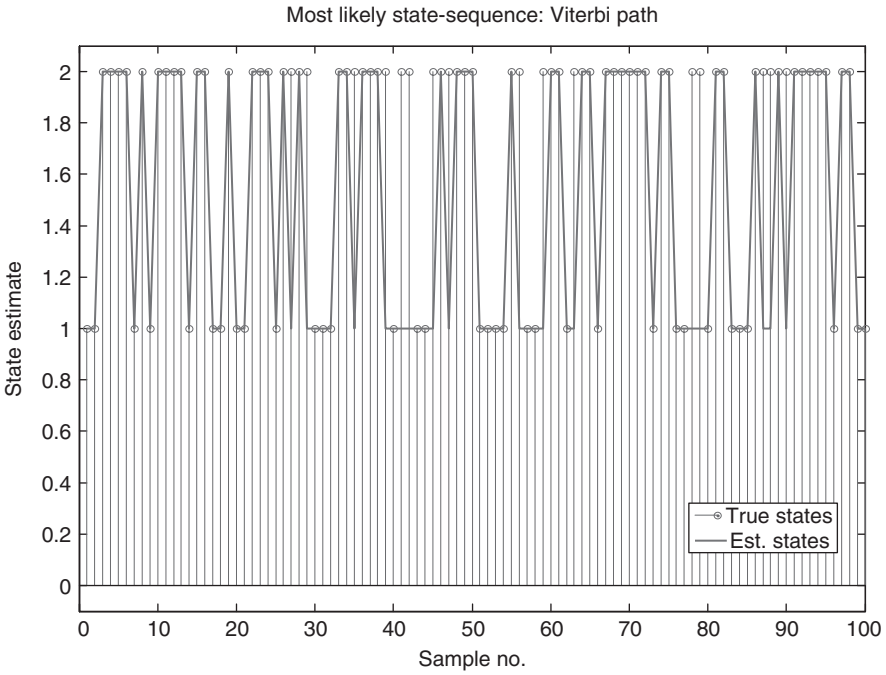


FIGURE 9.10 Entire state sequence estimation using the Viterbi algorithm illustrating the most likely path with an 82% match between the actual state sequence and the estimated.

Again the Viterbi initialization was also executed on this data but it did not perform very well. This completes the case study.

9.8 SUMMARY

In this chapter we have introduced the concept of hidden Markov models and illustrated their internal characteristics through a state-space representation. We first developed the concepts of Markov and hidden Markov chains and showed how they were related. Next, we investigated properties of the *HMM* illustrating how the Bayesian concepts easily transfer over to this discrete representation. We next investigated the three fundamental problems along with some variations: (1) the evaluation (simulation) problem; (2) the state estimation problem; and (3) the parameter estimation problem. A careful analysis of each led us to the popular *Viterbi* decoding technique and the specialized *EM* algorithm popularly called the *Baum-Welch* technique. We concluded with a case study to decode a transmitted coded sequence from data enhanced by a time-reversal processor.

MATLAB NOTES

MATLAB has a *Statistics Toolbox* that incorporates the capability to develop and process hidden Markov models (*HMM*) along with demonstrations and a tutorial. **hmmgenerate** synthesizes a sequence for a *HMM*, while the command **hmmdecode** calculates the posterior state probabilities of a given sequence (Sec. 9.3). The most likely (entire) state path can be estimated using the Viterbi algorithm of Sec. 9.5 (**hmmviterbi**). Training sequences can be used to solve the *HMM* parameter estimation problem (Sec. 9.6) while the *EM/Baum-Welch* technique is used to estimate the model parameters using the **hmmestimate** command. The *PDF* estimators include the usual histogram (**hist**) as well as the sophisticated kernel density estimator (**ksdensity**) offering a variety of kernel (window) functions (Gaussian, etc.).

There also exists *NETLAB* which is a free *MATLAB* software package that includes *HMM* algorithms (see [24] for details and website) as well as the *MATLAB*-based software package (free) called the "HMM toolbox" by K. Murphy (*http:www.cs.ubc.ca/murphyk/Software/HMM/hmm_usage.html*) which can be downloaded for use. The *EM* algorithm is well documented [20–22, 25–29] and an integral part of each of these packages.

REFERENCES

1. O. Cappe, "Ten years of HMMs", *www.tsi.enst.fr/cappe/docs/hmmmbib...*, 2001.
2. A. Papoulis and S. Pillai, *Probability, Random Variables and Stochastic Processes*, *4th Ed.* (Englewood Cliffs, NJ: McGraw-Hill, 2002).
3. R. Hogg, J. McKlean and A. Craig, *Introduction to Mathematical Statistics*, *6th Ed.* (Englewood Cliffs, NJ: Prentice-Hall, 2005).
4. D. Bertsekas and J. Tsitsiklis, *Introduction to Probability*, M.I.T. Lecture Notes, Course 6.041–6.043, 2000.
5. R. Durbin, S. Eddy, A. Krough and G. Mitchison, *Biological Sequence Analysis* (Cambridge, UK: Cambridge University Press, 1990).
6. D. MacKay, *Information Theory, Inference and Learning Algorithms* (Cambridge, UK: Cambridge Univ. Press, 2006).
7. R. Elliott, L. Aggoun and J. Moore, *Hidden Markov Models* (New York: Springer, 1994).
8. O. Cappe, E. Moulines and T. Ryden, *Inference in Hidden Markov Models* (New York: Springer-Verlag, 2005).
9. L. Rabiner and B. Juang, "An introduction to hidden Markov models," *IEEE ASSP Mag.*, **3**, 1, 4–16, 1986.
10. L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, **77**, 2, 257–286, 1989.
11. A. Poritz, "Hidden Markov models: A guided tour." *Proc. IEEE Confr. Acoustics, Speech and Sig. Proc.*, 7–13, 1988.
12. L. Rabiner and B. Juang, *Fundamentals of Speech Recognition* (Englewood Cliffs, NJ: Prentice-Hall, 1993).

13. L. White, "A comparison between optimal and Kalman filtering for hidden Markov processes," *IEEE Sig. Proc. Letters*, **5**, 5, 124–126, 1998.
14. S. Fruhwirth-Schnatter, *Finite Mixture and Markov Switching Models* (New York: Springer, 2006).
15. J. Bilmes, "What HMMs Can Do?" *Univ. Wash. Elect. Engr.*, UWEETR-2002-0003, 2002.
16. F. van der Heijden, R. Duin, D. de Ridder and D. Tax, *Classification, Parameter Estimation and State Estimation* (Hoboken, NJ: John Wiley, 2004).
17. G. Goodwin and R. Payne, *Dynamic System Identification* (New York: Academic Press, 1976).
18. L. Ljung, *System Identification: Theory for the User* (Englewood Cliffs, NJ: Prentice-Hall, 1987).
19. J. Candy, *Model-Based Signal Processing* (Hoboken, NJ: John Wiley/IEEE Press, 2006).
20. L. Baum, T. Petrie, G. Soules and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov Chains," *Ann. Math. Statist.*, **41**, 1, 164–171, 1970.
21. G. McLachlan and T. Krishnan, *The EM Algorithm and Extensions* (Hoboken, NJ: Wiley, 1997).
22. T. Moon, "The expectation-maximization algorithm," *IEEE Sig. Proc. Mag.*, **13**, 6, 47–60, 1996.
23. S. Theodoridis and K. Koutroumbas, *Pattern Recognition* (New York: Academic Press, 1999).
24. I. Nabney, *NETLAB Algorithms for Pattern Recognition* (New York: Springer, 2001).
25. J. Bilmes, "A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models," *Internat. Comp. Sci. Instit.*, TR-97-021, 1998.
26. A. Dempster, N. Laird and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. R. Statist. Soc.*, **B**, **39**, 1, 1–38, 1977.
27. R. Duda and P. Hart, *Pattern Classification* (New York: Wiley, 2000).
28. C. Wu, "On the convergence properties of the EM algorithm," *Ann. Statist.*, **11**, 1, 95–103, 1983.
29. R. Redner and H. Walker, "Mixture densities, maximum likelihood and the EM algorithm," *SIAM Rev.*, **26**, 2, 195–239, 1984.
30. M. Fink, "Time reversal in acoustics," *Contemporary Physics*, **37**, (2), 95–109, 1996.
31. J. Candy, A. Meyer, A. Poggio and B. Guidry, "Time reversal processing for an acoustic communications experiment in a hostile reverberant environment," *J. Acoust. Society Amer.*, **115**, (4), 1621–1631, 2004.
32. J. Candy, A. Poggio, D. Chambers, C. Robbins and B. Guidry, "Multichannel time reversal communications a hostile reverberant environment," *J. Acoust. Society Amer.*, **118**, (4), 2339–2354, 2005.
33. J. Candy, D. Chambers, C. Robbins, B. Guidry, A. Poggio, F. Dowla and C. Hertzog, "Wideband multichannel time-reversal acoustic communications highly reverberant environments," *J. Acoust. Society Amer.*, **120** (2), 838–851, 2006.
34. G. Kitagawa and W. Gersch, *Smoothness Priors Analysis of Time Series* (New York: Springer-Verlag, 1996).
35. A. Doucet, N. de Freitas and N. Gordon, *Sequential Monte Carlo Methods in Practice* (New York: Springer, 2001).

PROBLEMS

9.1 Suppose we have a noisy binary channel with input (symbol) x and output (symbol) y such that $x \in \mathcal{X} = \{0, 1\}$ and $y \in \mathcal{Y} = \{0, 1\}$. The transition probability matrix is:

$$A = \begin{bmatrix} \Pr(y = 0|x = 0) & \Pr(y = 0|x = 1) \\ \Pr(y = 1|x = 0) & \Pr(y = 1|x = 1) \end{bmatrix} = \begin{bmatrix} 0.85 & 0.15 \\ 0.15 & 0.85 \end{bmatrix}$$

Assume we observe the symbol $y = 1$ and $x \sim \Pr(x(0)) = [0.9 \;\; 0.1]'$, then

(*a*) What is posterior probability of x?

(*b*) Let $x = 1$, what is the posterior if this is true?

(*c*) Let $x = 0$, what is the posterior if this is true?

(*d*) Suppose $y = 0$, what are the corresponding posteriors of x?

9.2 A fly [4] moves along a straight line in unit increments. At each time period it moves one unit to the left with probability 0.3, one unit to the right with probability 0.3 and stays in place with probability 0.4. A spider is hiding at positions 1 and m: if the fly lands there, it is captured by the spider and the process ends.

(*a*) Construct a Markov chain model, assuming the fly starts at positions $2, \ldots, m - 1$.

(*b*) Sketch the corresponding directed graph.

(*c*) What is the probability of the following state evolution sequence: $\Pr(\mathcal{X}_1 = 2, \mathcal{X}_3 = 3, \mathcal{X}_4 = 4|\mathcal{X}_2)$?

9.3 Consider placing a ball in one of N compartments at each event. Each compartment can hold multiple balls. Let $x_i;\ i = 0, \ldots, N$ be the state where k compartments are occupied. At the next event, the next ball can go into one of the occupied compartments with probability k/N or an empty compartment.

(*a*) Create a Markov chain for this problem.

(*b*) What is the state diagram?

(*c*) What is the transition matrix, A.

9.4 Suppose we have a discrete state-space and discrete-time measurement system (sampled-data). Defining the discrete (finite) states as $x_i(t)$ and the measurements as $y(t)$, then:

(*a*) What is the state prediction probability, $\Pr(x_i(t)|Y_{t-1})$?

(*b*) What is the state posterior distribution, $\Pr(x_i(t)|Y_t)$?

(*c*) Sketch out the steps of the Bayesian filtering operation at times: $t - 1$ and t.

9.5 Autoregressive ($AR(N_a)$) models (all-pole) occupy a large part of the signal processing literature especially in speech applications [10].

(*a*) An *AR* model is an example of a Markov chain on a continuous space, show that the *AR*(1) model forms a Markov chain, that is,

$$y(t) = a_1 y(t-1) + \epsilon(t), \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

(*b*) Show an $AR(N_a)$ model also forms a Markov chain. (*Hint*: Place the *AR* model in state-space form).

(*c*) Does an $ARMA(N_a, N_a)$ model form a Markov chain, as well?

9.6 Autoregressive ($AR(N_a)$) *switching* models also occupy a significant part of the time series literature [14]. It is a model where the mean can switch between two values, μ_0 and μ_1. It enables the time series with several regimes or local nonstationarities to be represented as:

$$\begin{aligned}\Pr(y(t)|y(t-1), x(t), x(t-1)) &\sim \mathcal{N}(\mu_{x(t)} + a_1(y(t-1) - \mu_{x(t-1)})), \sigma^2) \\ \Pr(x(t)|x(t-1)) &\sim p_{x(t-1)}\mathcal{I}_{x(t-1)}(x(t)) \\ &\quad + (1 - p_{x(t-1)})\mathcal{I}_{1-x(t-1)}(x(t))\end{aligned}$$

where the hidden state, $x(t)$ takes values in $\{0, 1\}$ with initial values, $x(0) = \mu_0$ and $y(0) = 0$, and $\mathcal{I}$ is an indicator function.

(*a*) What is the sampling distribution, $\Pr(x(t)|x(t-1), x(t+1), y(t), y(t-1), y(t+1))$?

(*b*) Using the priors, $\Pr(\mu_1 - \mu_0) \sim \mathcal{N}(0, \gamma^2)$, $\Pr(a, \sigma^2)$ with $p_0, p_1 \sim \mathcal{U}(0, 1)$ simulate the *AR* switching model for $N = 500$ samples. What are the final parameter estimates for: $\{\mu_0, \mu_1, a, p_1, p_2\}$?

9.7 Consider a 4-state Markov switching model [34] with the set of discrete states given by $\mathcal{S} = \{1, 2, 3, 4\}$ representing a Markov chain with transition matrix:

$$A = \begin{bmatrix} 1-3\alpha & \alpha & \alpha & \alpha \\ \beta & 1-\beta-2\gamma & \gamma & \gamma \\ \beta & \gamma & 1-\beta-2\gamma & \gamma \\ \beta & \gamma & \gamma & 1-\beta-2 \end{bmatrix}$$

The observation sequence is specified by a set of *AR*(2) models:

$$y(t) = a_{1i}y(t-1) + a_{2i}y(t-2) + \epsilon_i(t) \quad \text{for } \mathcal{S}_i; \;\; i = 1, \ldots, 4$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$

(*a*) What is the (state) prediction probability for this model, $\Pr(s(t)|Y_{t-1})$?

(*b*) What is the (state) filtering posterior probability for this model, $\Pr(s(t)|Y_t)$?

(*c*) What is the likelihood for the following set of parameters describing this model, $\Theta = \{\alpha, \beta, \gamma, \sigma^2, a_{1i}, a_{2i}\}; i = 1, \ldots, 4$?

(*d*) Simulate the system with the following parameters $\{\alpha, \beta, \gamma, \sigma^2\} = 0.0033$, 0.016, 0.002, 0.1 and $\{a_{1i}, a_{2i}\} = \{(1.785, -0.903), (1.344, -0.903), (1.386, -0.640), (0.800, -0.640)\}$ for $N = 1000$ samples.

9.8 Suppose we have a *HMM* model with discrete state–space defined by $\mathcal{X} = \{x_i(t) = \mathcal{X}_i\}$ $i = 1, \ldots, N_x$ with state transition matrix A. We would like to establish a parameter estimation problem given the observation sequence, Y_T. Assume that the complete likelihood is given by $\Pr(y, x|\theta)$ and the density is given by: $\Pr(y|x_i, \theta) = \mathcal{N}(\mu_{x(t)}, \Sigma)$ with known diagonal covariance and unknown mean. Develop the *EM* algorithm to estimate the parameter μ, that is,

(*a*) What is the Q-step?

(*b*) What is the M-step?

9.9 State–space models are perhaps the most important class of linear dynamic systems characterized by:

$$\begin{aligned} x(t+1) &= Ax(t) + w(t) \\ y(t) &= Cx(t) + v(t) \end{aligned}$$

Develop the *EM* algorithm for this class of model based on the usual Gauss-Markov assumptions.

(*a*) Suppose we are asked to estimate the A and C parameters. What is corresponding the E-step? (*Hint*: Use gradient techniques.)

(*b*) What is the associated M-step?

10

BAYESIAN PROCESSORS FOR PHYSICS-BASED APPLICATIONS

In this chapter we develop a set of Bayesian signal processing applications based on the underlying physical phenomenology generating the measured process. The complexity of the process and the desire for enhanced signals drives the design primarily indicated by the process model. We motivate each application, succinctly develop the process and measurement models, develop the *BSP* and analyze its performance. More details on any of the designs can be obtained from the references for the interested reader. The main objective is to demonstrate the applicability of the Bayesian approach to a variety of interesting applications and analyze their performance.

10.1 OPTIMAL POSITION ESTIMATION FOR THE AUTOMATIC ALIGNMENT

The alignment of high energy laser beams for fusion experiments demand high precision and accuracy of the underlying positioning algorithms whether it be for actuator control or monitoring the beam line for potential anomalies. This section discusses the development of on-line, optimal position estimators in the form of Bayesian processors to achieve the desired results. Here we discuss the modeling, development, implementation and processing of Bayesian processors applied to both simulated and measured beam line data.

10.1.1 Background

Alignment of a high power beam is a complex and critical process requiring precise and accurate measurements. Misalignment of such a beam could easily destroy costly optics causing a deleterious disruption of an entire experiment. The alignment of large

Bayesian Signal Processing. By James V. Candy

operative, short pulse, laser systems is a significant and costly endeavor dating back to the late sixties and early seventies [1]. Contemporary imaging systems employ high-resolution video cameras to image and accurate position control systems to align the beam. This approach estimates the current beam position from the image, adjusts mirrors relative to an accurate reference measurement of physical beam center and minimizes their difference or deviation [2–9]. However, even with these sophisticated measurements, the beam position estimate is still a function of the inherent beam noise caused by the internal beam line gas turbulence as well as instrumentation noise [10]. Here we introduce the idea of post-processing these uncertain measurements using advanced signal processing techniques to "enhance" the raw data and "detect" any beam line anomalies during laser system operations [11].

High power, tightly focused laser beams are required to achieve successful ignition and therefore fusion at the Lawrence Livermore National Laboratory (LLNL) National Ignition Facility (NIF) [12]. These beams must simultaneously focus precisely on a nanoscale target capsule to succeed. Therefore, there are a large number of alignment measurements that must be performed along the NIF beam line to assure that the pointing and alignment control system centers the beam in order to provide the maximum energy on the fusion target located in the associated chamber [12–14]. An automatic alignment (AA) system was designed and implemented to assure successful deployment of the high energy beam in each of the 192 beam lines. However, since a variety of techniques are provided to perform the alignment, there is a quantifiable uncertainty associated with each technique that may or may not meet the desired accuracy and precision specifications associated at each control point. Therefore, there is a need for a post-processing technique, which accepts as input an uncertain position measurement and provides as output an improved position estimate (see Fig. 10.1). As illustrated in the figure, the measured image position estimate is compared to the reference image estimate providing an position error that is input to the mirror control system that moves the associated mirrors correcting the beam position.

Perhaps the most challenging of all beam line measurements are those made on the KDP (potassium dihydrogen phosphate) crystals. These crystals are critical elements

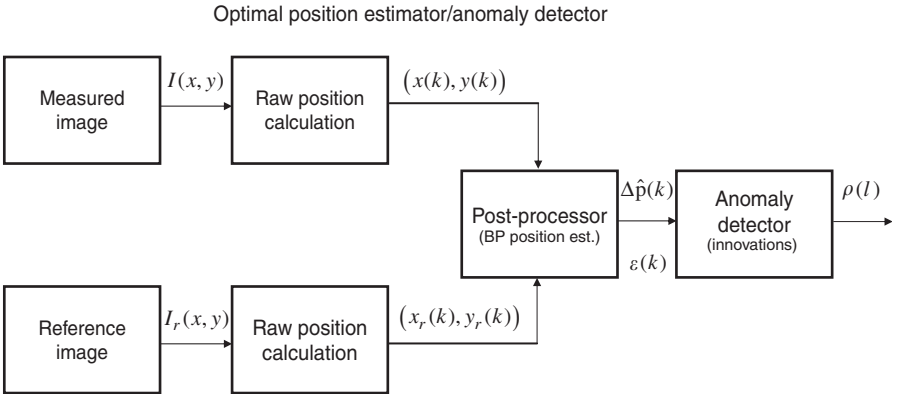


FIGURE 10.1 Optimal position estimation with anomaly detector: Bayesian post-processing and innovations-based anomaly detection.

used to double or triple the frequency of the laser beam as it passes providing a shorter wavelength. The higher frequency determined from the target physics enables laser plasma interactions for fusion. The NIF final optics assembly matches lenses to this particular frequency producing the beam that is tightly focused on the target capsule required to achieve fusion ignition. In order for the KDP crystals to optimally double or triple the laser operating frequency, they must be precisely positioned at the appropriate angle. This is one of the critical tasks of the AA system. The KDP measurement consists of using a charge coupled device (CCD) imaging camera, which produces a noisy back reflection image of a diagnostic alignment beam. The noise is caused by the camera itself as well as uncertainties that are due to small pointing errors made during the measurement. A sophisticated two-dimensional (2D) phase-only, matched-filter [15] algorithm was developed to provide the initial raw position estimates.

The image acquired from the CCD imaging camera produces both noisy measurement and reference images. A precise reference image is used to provide the desired fiducial that is used by the alignment system. Corrections to align the measured image with the reference is accomplished using the dedicated control loops that adjust pointing mirror stepping motors until the deviations between both reference and measured positions are within acceptable limits [14]. Ultimately, the goal is to make this difference zero assuring proper beam alignment. Fig. 10.2 shows a sequence of measured KDP images with position estimates (O) along with the corresponding reference image measurement (+). Note how the control loop adjusts the KDP measurements (O) until the centroid position converges to the reference position (+). The objective, therefore, of the control loop is to adjust the beam mirrors such that both measured and reference positions completely overlap (zero deviation) as shown in the bottom of this figure. Thus, the smaller the XY-deviations, the closer the beam is to

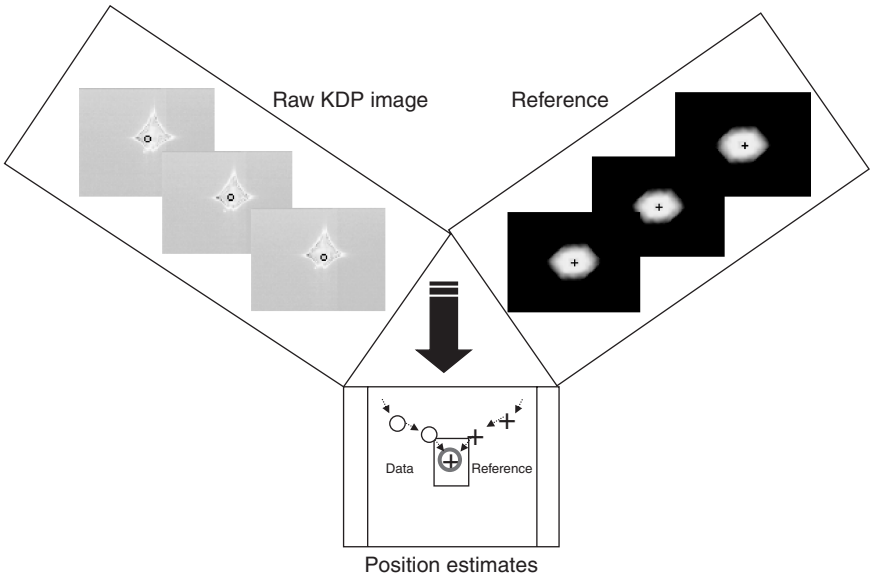


FIGURE 10.2 Raw KDP crystal back-reflection image and reference position estimates.

the centerline reference assuring a tightly focused, high energy beam on target—the goal of the alignment system. Should, for some reason, an anomaly develop in any beam line, it may not be possible to align for a particular shot. The timely detection of beam line anomalies are necessary to avoid future problems, which, if left unmitigated could result in less than optimum performance of the laser. In this work, we show how a Bayesian processor (*BP*) can be used to detect anomalies, on-line, during the calibration phase of a laser shot.

Thus, we discuss the feasibility of applying a Bayesian processor as an on-line post-processing technique to improve the final position estimates provided by a variety of estimators and detect anomalies in a high energy laser beam line [11]. These estimators are used to align high-powered laser beams for experiments at NIF. We first motivate a stochastic model of the overall process and measurement system. Next, we discuss the underlying theory for both position estimation and anomaly detection. With the theory in hand, we develop the processor for the KDP application through ensemble statistics, simulation and application to real measurement and reference data.

10.1.2 Stochastic Modeling of Position Measurements

The typical beam position estimator is accomplished by calculating the centroid of the measured images [18]. The point is that these measured positions are derived from the associated noisy CCD images. We define the true centroid positions by the position vector $\mathbf{p}(k) := [\mathbf{x}(\mathrm{k}) \mid \mathbf{y}(\mathrm{k})]'$, where $\mathbf{p} \in R^{N_p \times 1}$ and k is the sample time. Since quite a number of images are acquired daily, a large data base consisting of position estimates are available, say, $P(t) = \{\mathbf{p}(k)\}; k = 1, \ldots, K$ and the position estimates are to be updated continuously. Since we know that the images are contaminated with noise and uncertainty, a more reasonable position measurement model is given by

$$\mathbf{z}(k) = C\mathbf{p}(k) + \mathbf{v}(k) \tag{10.1}$$

where $\mathbf{z}, \mathbf{v} \in \mathbf{R}^{N_z \times 1}$, $C \in \mathbf{R}^{N_z \times N_p}$ and $\mathbf{v} \sim N(0, \mathbf{R}_{vv})$, that is, the measurement noise is assumed zero mean, multivariate Gaussian with covariance matrix, $\mathbf{R}_{vv} \in R^{N_z \times N_z}$.

We also note that since the CCD camera uses the identical beam line to measure both images, we model the position estimates as piecewise constants contaminated with beam line noise besides that noise contributed by the measurement systems. This process noise can be considered fluctuations caused by the inherent system optical transfer functions and turbulence caused by the argon gas filled housing during the laser beam propagation (boiling noise). Therefore, we assume that the contaminated position measurement is represented as

$$\dot{\mathbf{p}}(t) = \mathbf{0} + \mathbf{w}(t) \quad \text{with} \quad \mathbf{p}(0) = [x(0) \mid \mathrm{y}(0)]' \tag{10.2}$$

in continuous time, but if we discretize over the k-sample images, then using first differences we obtain

$$\dot{\mathbf{p}}(t) \approx \frac{\mathbf{p}(t_{k+1}) - \mathbf{p}(t_k)}{\Delta t_k} = \mathbf{w}(t_k) \quad \text{for } \Delta t_k := t_{k+1} - t_k \tag{10.3}$$

Substituting into Eq. 10.2, we obtain the following Gauss-Markov position model

$$\mathbf{p}(t_{k+1}) = \mathbf{p}(t_k) + \Delta_{t_k}\mathbf{w}(t_k) \tag{10.4}$$

where $\mathbf{p}, \mathbf{w} \in \mathbf{R}^{N_p \times 1}$ and $\mathbf{w} \sim \mathbf{N}(0, \mathbf{R}_{ww})$ along with the accompanying measurement model of Eq. 10.1. This completes the basic description of the underlying Gauss-Markov model. We note in passing that for a complete description of the general model including means, $\mathbf{m}_p(k), \mathbf{m}_z(k)$ with their associated covariances see [11] for more details.

Since we have both final measured and reference position estimates, we re-define a more convenient position vector by grouping the positions as

$$\mathbf{p}(k) := \begin{bmatrix} x(k) \\ y(k) \\ --- \\ x_r(k) \\ y_r(k) \end{bmatrix} \quad \text{and} \quad \mathbf{w}(k) := \begin{bmatrix} w_x(k) \\ w_y(k) \\ --- \\ w_{x_r}(k) \\ w_{y_r}(k) \end{bmatrix}$$

where $\mathbf{p}(0) \sim \mathrm{N}(\overline{\mathbf{p}}(0), \overline{\mathrm{R}}_{\mathrm{pp}}(0))$ are the respective measurement and reference positions coordinates (pixels) with the corresponding process noise covariance matrix given by

$$\mathbf{R}_{ww} = \begin{bmatrix} \overline{\mathbf{R}}_{ww} & | & 0 \\ - & - & - \\ 0 & | & \overline{\mathbf{R}}_{w_r w_r} \end{bmatrix}$$

since each of the measured and reference images are uncorrelated.

This type of formulation enables us to estimate the process noise covariances independently for each image as well as characterize their uncertainties individually. Since the control loop jointly uses both the reference and measured images to produce its final centroid position estimates, we model the measurement matrix as the deviation (difference) between these data contaminated with independent measurement noise, that is, our measurement model of Eq. 10.1 becomes

$$\mathbf{z}(k) = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \mathbf{p}(k) + \mathbf{v}(k)$$

with measurement covariance matrix

$$\mathbf{R}_{vv} = \begin{bmatrix} \sigma_{vv}^2 & | & 0 \\ - & - & - \\ 0 & | & \sigma_{v_r v_r}^2 \end{bmatrix}$$

and $\sigma_{vv}^2 = \sigma_{xx}^2 + \sigma_{yy}^2$ and $\sigma_{v_r v_r}^2 = \sigma_{x_r x_r}^2 + \sigma_{y_r y_r}^2$. Note that the deviations are defined by

$$\Delta\mathbf{p}(k) := C\mathbf{p}(k) = \begin{bmatrix} \Delta x(k) \\ \Delta y(k) \end{bmatrix} = \begin{bmatrix} x(k) - x_r(k) \\ y(k) - y_r(k) \end{bmatrix}$$

This completes the section on position (uncertainty) modeling, next we consider the development of the optimal position estimator.

10.1.3 Bayesian Position Estimation and Detection

It is well-known that the optimal solution to the position estimation problem under the Gauss-Markov model assumption is provided by the Bayesian (state–space) processor or Kalman filter of Chapter 5. This solution can be considered a predictor-update design in which the processor uses the model (random walk) to predict in absence of a measurement and then updates the estimate when the measurement becomes available. Under the assumption of a perfect model, that is, the model embedded in the process exactly matches the process, the *BP* is capable of achieving the optimal minimum (error) variance estimate under the Gaussian assumptions.

For the position estimation problem, the *BP* takes on the following predictor-update form where $t_k \rightarrow k$:

$$\begin{aligned}
&\textit{Prediction:} && \hat{\mathbf{p}}(k+1|k) = \hat{\mathbf{p}}(k|k) \\
&\textit{Innovation:} && \varepsilon(k+1) = \mathbf{z}(k+1) - \hat{\mathbf{z}}(k+1|k) = \mathbf{z}(k+1) - \mathbf{C}\hat{\mathbf{p}}(k+1|k) \\
&\textit{Update:} && \hat{\mathbf{p}}(k+1|k+1) = \hat{\mathbf{p}}(k+1|k) + \mathbf{G}(k+1)\varepsilon(k+1) \\
&\textit{Gain:} && \mathbf{G}(k+1) = \widetilde{\mathbf{P}}(k+1|k)\mathbf{C}^T\mathbf{R}_{\varepsilon\varepsilon}^{-1}(k+1)
\end{aligned}$$

for $\hat{\mathbf{p}}(k+1|k) := E\{\mathbf{p}(k)|\mathbf{p}(k-1), \ldots, \mathbf{p}(1)\}$ and $\hat{\mathbf{z}}(k+1|k)$, the underlying conditional means and $\mathbf{G}(k+1) \in \mathbf{R}^{N_p \times N_z}$ is the corresponding gain matrix calculated from the error covariance matrix, $\widetilde{\mathbf{P}}(k+1|k) = \mathrm{Cov}(\widetilde{\mathbf{p}}(k+1|k))$ with position error $\widetilde{\mathbf{p}}(k+1|k) := \mathbf{p}(k+1) - \hat{\mathbf{p}}(k+1|k)$. The innovations covariance matrix is defined by $\mathbf{R}_{\varepsilon\varepsilon}(k+1)$.

Recall that for a Gauss-Markov representation a necessary and sufficient condition for the *BP* to be optimal is that the innovations sequence be zero mean and white (uncorrelated)—conditions that we test during processor design. It is possible to exploit this property of the innovations to detect anomalies in the system revealing potential problems in the beam line in pseudo real-time—a large asset when attempting to automate the alignment system. We apply two detection techniques to monitor the position estimates from the daily measurements: zero-mean/whiteness detector and the weighted-sum-squared-residual (*WSSR*) detector (see Sec. 5.7 for details). Both of these techniques rely on the assumption that the position deviations in the loop should change little during the measurement cycles. The underlying idea in anomaly detection is that the processors are "tuned" during calibration to operate in an optimal manner (innovations zero-mean/white). However, when an anomaly occurs, the innovations no longer maintain their optimal statistical properties leading to a "change from normal" and an anomaly is declared. With this detection, a decision must be made to either classify the anomaly or perform some other action. The main point to realize is that since we are using a recursive-in-time *BP*, the innovations represent "how well the position model and its underlying statistics represent the raw data." If it is an optimal fit, then the innovations should be zero-mean and statistically white (uncorrelated) and the detectors evolve naturally.

This completes the description of the *BP* and the associated anomaly detectors, next we consider a simulation of the data and processor to predict the expected performance.

10.1.4 Application: Beam Line Data

In this section, we discuss the application of optimal position estimators based on daily historical position estimates (data base) provided by the KDP algorithm. The data are a record of 48 days of final positions output from both the accurate reference system (reference data) and the estimated final position output from the KDP back-reflection data coupled to a phase-only, matched filter [15] imaging algorithm. The motivation for applying the Bayesian approach to this data set is to obtain a more accurate and precise estimate of error deviations from the reference characterizing the overall control loop performance in that position of the beam line. Theoretically, the position deviations between the reference and the KDP estimates should ideally be zero, but because of the beam line noise and variations, CCD camera limitations and control system tolerances, this is not the case. Instead the overall error statistics are used to bound the performance and assure that they remain within design specification.

10.1.4.1 BP Design In the design of the *BP* the usual procedure is to: (1) develop the required models; (2) simulate a set of position data characterized by any of the *a priori* information available; (3) develop the minimum error variance design; and (4) apply the processor to the available data set evaluating its performance. We developed the basic model set assuming a Gauss-Markov structure as in the previous subsection. These parameters of this model were estimated from the statistics of a large ensemble (>5000) of images and performance statistics of the KDP centroiding algorithm. We used this information to construct the initial simulation of the *BP* to "match" with the historical data available (48 days) and the corresponding ensemble statistics.

Thus, our approach is to first perform a simulation of the measurement process using estimated statistics from the data and then apply it to the actual data. During the simulation phase, we are able to analyze the performance of the *BP* and assure ourselves that all of the models and statistics are correct. We expect to obtain the minimum variance estimates, if not achieved, then it is usually an implementation issue. Once the simulation and model adjustments have been made, we apply the processor to the measured data.

10.1.4.2 Simulation Now that we have developed the models and have some estimates from the ensemble statistics of the database, we are now able to perform a Gauss-Markov simulation to assess the feasibility of the *BP*. We simulated a set of data based on the following parameters using the mean XY-position estimates, that is, $x_i = \mu_{x_i} \pm 1.96\sigma_i$ for both the measured and reference data: $\mathbf{x}(0) = [344 \pm 8,\ 270 \pm 13.8,\ 344 \pm 7.8,\ 270 \pm 14.4]'$. We used the mean values as the initial position estimates with low error variances (1×10^{-6}). We chose an uncorrelated measurement (deviation) noise covariance matrix as: $\mathbf{R}_{vv} = \text{diag}[0.0076 \quad 0.0078]$. The process

noise covariance selected is

$$\mathbf{R}_{ww} = \begin{bmatrix} \overline{\mathbf{R}}_{ww}(1) & 0 \\ 0 & \overline{\mathbf{R}}_{ww}(2) \end{bmatrix} \quad \text{for } \overline{\mathbf{R}}_{ww}(1) = \begin{bmatrix} 50.0 & -26.5 \\ -26.5 & 50.0 \end{bmatrix};$$

$$\overline{\mathbf{R}}_{ww}(2) = \begin{bmatrix} 50.0 & -37.7 \\ -37.7 & 50.0 \end{bmatrix}$$

The synthesized positions have small variations due to the process noise, but are essentially constants (mean values). The simulated noisy DX, DY-deviation measurements (dotted line) including the process and measurement noise are shown in Fig. 10.3. Next we executed the *BP* over this data set and the position deviations (solid line) are also shown in the figure. Here we see that much of the raw measurement and process noise have been removed by the processor and only the deviation errors are

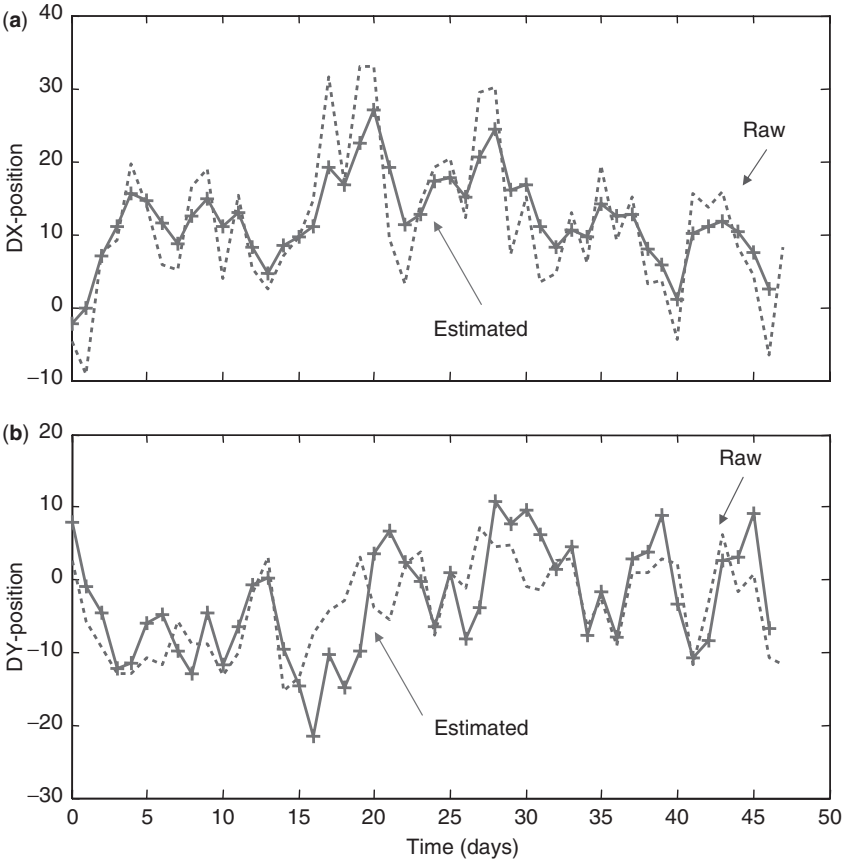


FIGURE 10.3 Simulated XY-position deviations: Raw (dotted) and estimated (solid). (a) X-deviation. (b) Y-deviation.

shown. Recall that they should be close to zero for the alignment control system to be operating efficiently.

To validate the results, we investigate the minimum variance design procedure. We investigate the statistics of the innovations sequence and check that they are zero-mean and white for optimality. Both innovations sequences are zero-mean: ($0.30 < 3.5$; $0.95 < 3.3$) and white: (4% out; 0% out); with the *WSSR* decision function lying below the threshold for a window of size N: (Threshold $= 69.6$, $N = 25$). These statistics indicate that the minimum variance position deviation estimates have been achieved. This result is expected, since the models used in the *BP* are identical to those in the Gauss-Markov simulator. However, the results can be interpreted as the "best" (minimum variance) one could hope to do under these circumstances. Next we apply the *BP* to the actual deviation data.

10.1.5 Results: Beam Line (KDP Deviation) Data

In order to process the KDP deviation data, we must develop parameters for the underlying model embedded in the processor. We start with guesses of the noise (process and measurement) statistics from the simulator and then adjust them accordingly. For instance, if the innovation sequence lies outside its predicted bounds implying that the measurement noise variance is too small, then it can be adjusted to satisfy this constraint and "match" the data. The process noise is actually quite difficult to

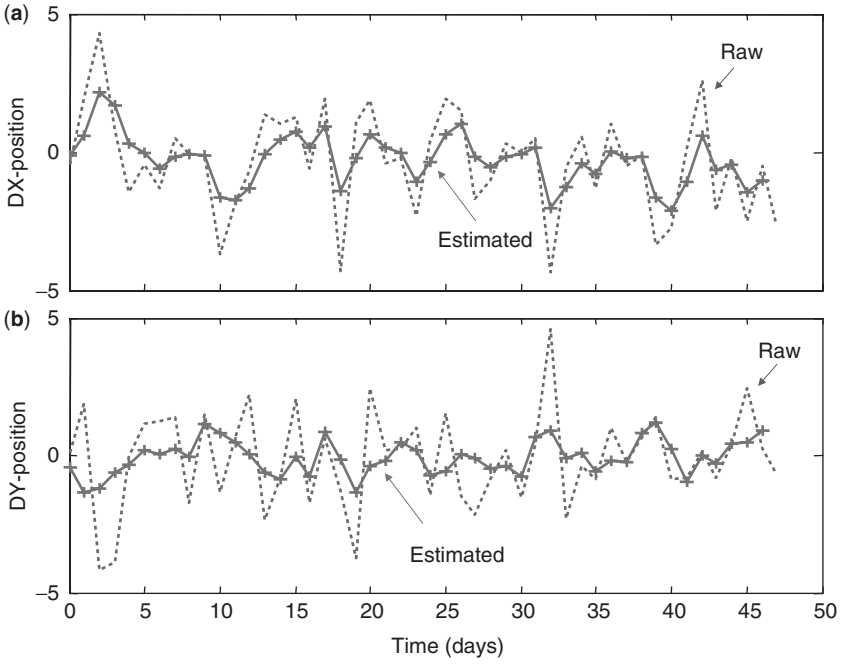


FIGURE 10.4 Actual KDP data XY-deviations and estimated XY-positions: Raw measured data (dashed). Estimated data (solid): (a) ΔX-position. (b) ΔY-position.

select, since it is directly proportional to the *BP* gain. In essence, once all of the other model parameters are reasonably adjusted, they are then held fixed and the process noise covariance is varied to achieve the "best" possible (minimum error variance) innovations statistics (zero-mean/white, *WSSR* below threshold).

The results of the *BP* design for the KDP deviation data as shown in Fig. 10.4 along with the corresponding error ellipsoids in Fig. 10.5. We see the raw KDP position deviation estimates along with the optimal processor results over the 48-day period in Fig. 10.4. It is clear that the processor is tracking the trends in the data while reducing the noise or equivalently enhancing the SNR. To confirm this we observe the three-sigma error ellipsoid plots of Fig. 10.5 where we observe that much of the uncertainty has been removed and the estimated deviations are clearly clustered (centered) around the (0,0) position and have a much smaller ellipsoid (better precision) than the raw data. Again to confirm the optimality of the processor we check the zero-mean-whiteness/*WSSR* statistics which give the following results: zero-mean: ($0.08 < 0.80$; $0.05 < 0.71$); approximately white: (4% out; 8% out); and *WSSR* decision function lies below the threshold for a window of size N: (Threshold $= 69.6$, $N = 25$). The results

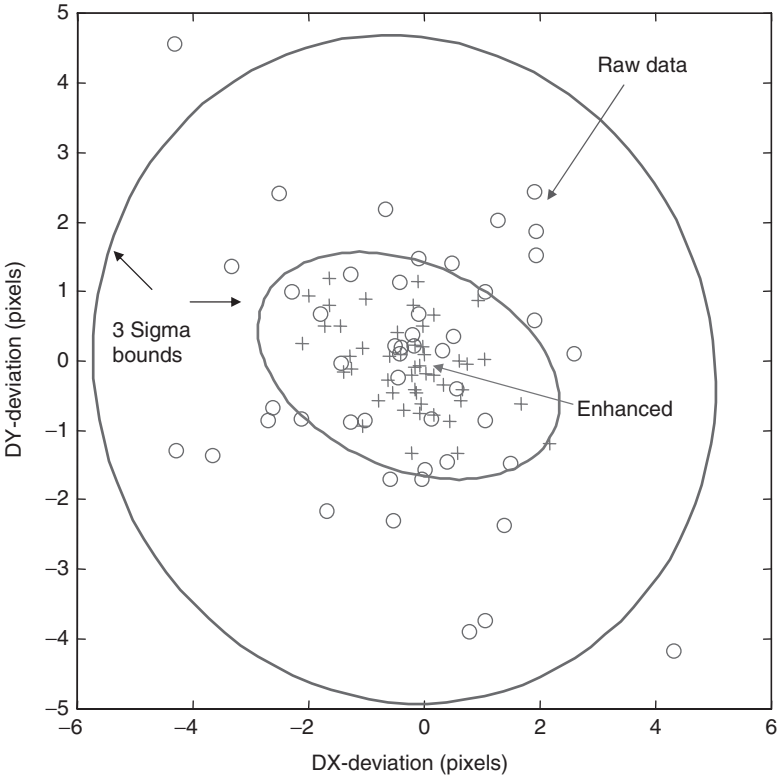


FIGURE 10.5 Three-sigma error ellipsoid for KDP XY-position deviations from simulated data: Raw deviations (O) and estimated (+) with less outliers.

are shown in Fig. 10.6. These statistics again indicate that the minimum variance design has been achieved and the processor along with its associated statistics are valid and optimal. This completes the *BP* design for the position estimation problem. Next we investigate the feasibility of using this approach for anomaly detection.

10.1.6 Results: Anomaly Detection

We use the actual KDP measurement data as before to assess the feasibility of applying the *BP* as an effective means to detect anomalies (off-normal) that could occur in the beam line. Here we assume that the *BP* has been "tuned" to normal operations for the particular beam line and optics. Thus, the innovations are zero-mean, white and the *WSSR* lies below the threshold for optimal design. To "simulate" an off-normal condition (in terms of position estimates), we increase the amplitude of both X and Y deviations fivefold in the raw measurement data during the period of 29–35 days.

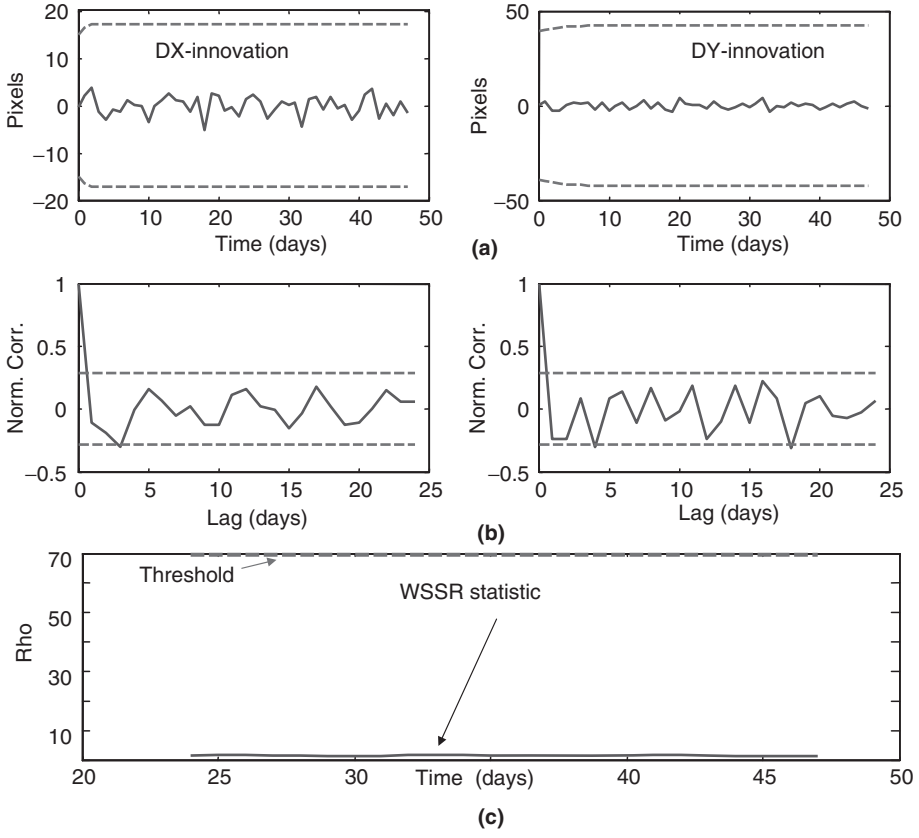


FIGURE 10.6 *BP* design for KDP XY-position deviation data: (a) Innovations for X-deviation and Y-deviation. (b) Optimality tests: Zero-mean ($0.08 < 0.80$; $0.05 < 0.71$) and whiteness (4% out; 8% out) (c) WSSR Test (threshold = 69.6, $N = 25$).

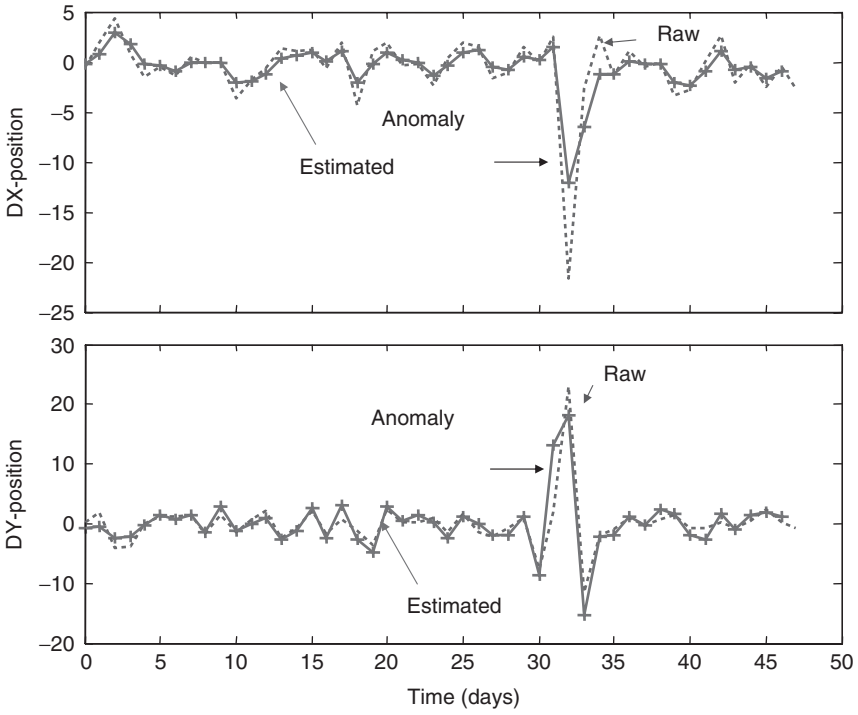


FIGURE 10.7 Raw KDP data XY-deviations with simulated anomaly: Raw (dotted) and estimated (solid).

This can be thought of as corresponding to a pulse-like transient in either or both of the measurement and/or reference position data. We applied the optimal (for no anomaly) *BP* to this data to determine whether or not it could detect and track the unknown transient anomaly.

The results of the effect on the deviation measurements are shown in Fig. 10.7. We see the normal measurement and then the transient "jump" within the prescribed time period. It is interesting to note that there is a corresponding disturbance in the estimated (filtered) measurement indicating that there is enough of a disruption at this SNR to cause the *BP* to track it. However, the *BP* is not able to track the transient extremely well—the expected results. Next we observe the deviation innovation sequences output by the *BP* in Fig. 10.8. We again observe the transient anomaly in both sequences, since the position estimates do not track it well enough. The zero-mean/whiteness tests seem a bit too insensitive to the rapid change (6 samples) and do not dramatically detect it even the whiteness tests do not indicate a non-white sequence (0% out; 4% out). On the other hand, the *WSSR* test clearly detects "change from normal" caused by the simulated beam line anomalies almost instantaneously indicating a feasible solution. This is again expected since the *WSSR* statistic ($\rho(\ell)$) can be tuned to transient disruption by selecting the appropriate window length. For

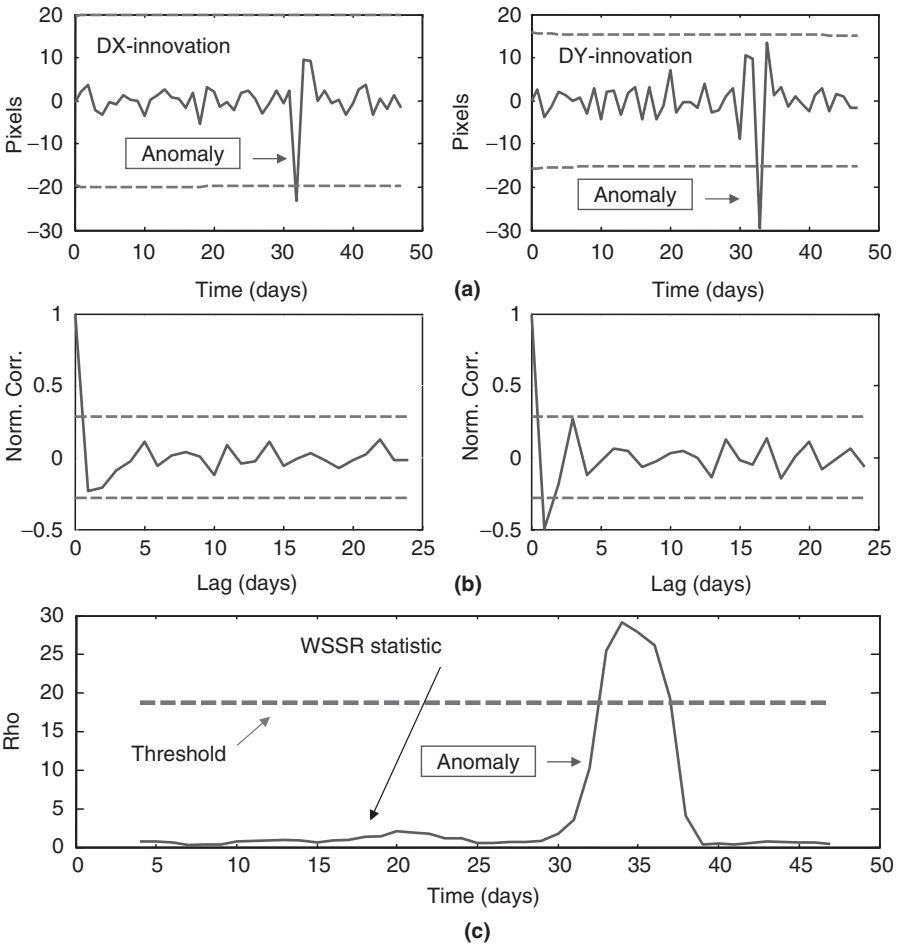


FIGURE 10.8 *BP* detection with simulated anomaly: (a) DX and DY innovations with anomaly. (b) Optimality tests: Zero-mean ($0.07 < 1.7$; $0.05 < 2.3$); whiteness (0% out; 4% out). (c) *WSSR* test (Threshold = 18; $N = 5$).

this problem we chose: ($N = 5$; Threshold = 18). To verify the performance of the zero-mean/whiteness test we observe the estimated positions (states) in Fig. 10.9. Here we see that the *BP* actually responds to such a high amplitude level change in the anomaly transient. The processor could easily be "tuned" to ignore the change of the states by decreasing the process noise variance. Finally, the scatter plots for this case are shown in Fig. 10.10: one for the optimal solution (no anomaly) and one for the solution with the anomaly. The size difference of the error ellipsoids in both cases is obvious due to the added uncertainty of the modeled anomaly. The three-sigma error ellipsoid for the case with an anomaly suspiciously indicates that something is different, since the estimated deviation uncertainty (ellipsoid) is actually about the

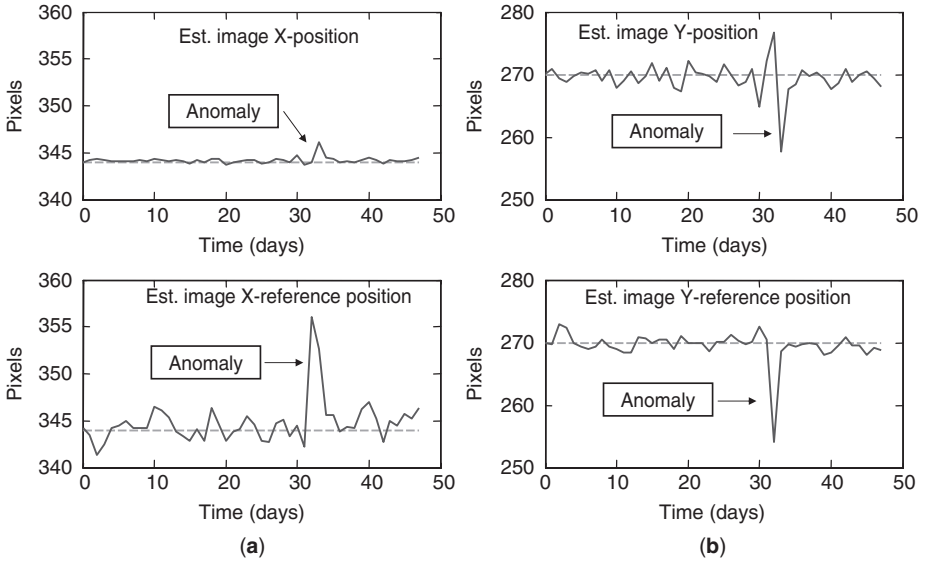


FIGURE 10.9 Estimated XY-positions with anomaly: (a) Measured position estimates. (b) Reference position estimates.

same size or larger than that of the raw data. This completes the study of applying a Bayesian anomaly detector.

10.2 BROADBAND OCEAN ACOUSTIC PROCESSING

Acoustic sources found in the ocean environment are spatially complex and broadband, complicating the analysis of received acoustic data considerably. A Bayesian approach is developed for a broadband source in a shallow ocean environment characterized by a normal-mode propagation model. Here we develop the "optimal" Bayesian solution to the broadband pressure-field enhancement and modal function extraction problem.

10.2.1 Background

Acoustic sources found in the hostile ocean environment are complex both spatially and temporally being broadband rather than narrowband. When propagating in the shallow ocean these source characteristics complicate the analysis of received acoustic data considerably—especially in littoral regions providing an important challenge for signal processing [19–24]. It is this broadband or transient source problem that leads us to a Bayesian signal enhancement solution.

The uncertainty of the ocean medium motivates the use of stochastic models to capture the random nature of the phenomena ranging from ambient noise and scattering

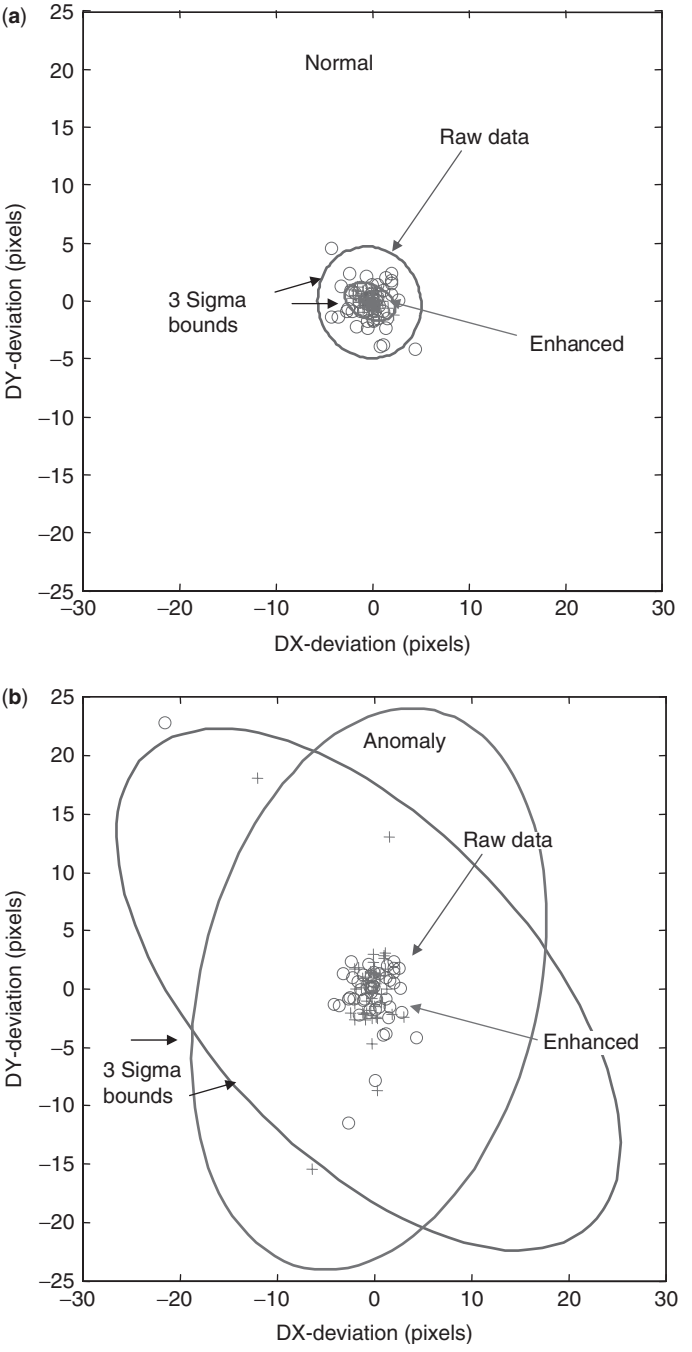


FIGURE 10.10 Three-sigma error ellipsoid and estimated XY-position deviations: (a) Measured data (○) no anomaly. (b) Measured data (○) with anomaly. Enhanced (+).

to distant shipping and the nonstationary nature of this hostile environment. When contemplating the broadband problem it is quite natural to develop temporal techniques especially if the underlying model is the full wave equation; however, if we assume a normal-mode propagation model then it seems more natural to: (1) filter the broadband receiver outputs into narrow bands; (2) process each band with a devoted processor; and then (3) combine the narrowband results either coherently [25] or incoherently [26, 27] to create a broadband solution. One apparent advantage to this approach is to utilize narrowband signal processing techniques thereby providing some noise rejection and intermediate enhancement for the next stage. This is the approach we take in this section to construct a broadband Bayesian processor (*BP*), that is, we first decompose the problem into narrow bands and construct a bank of Bayesian processors–one for each band. Finally, we incoherently or coherently combine the outputs of each to provide an overall enhanced signal useful for detection and localization.

Employing a vertical array of hydrophone sensors, the enhancement of noisy broadband acoustic pressure-field measurements using a multichannel Bayesian technique is discussed. Here the Bayesian approach is developed for the broadband source using a normal-mode propagation model. Propagation theory predicts that a different modal structure evolves for each spectral line; therefore, it is not surprising that the multichannel Bayesian solution to this problem results in a scheme that requires a "bank" of processors—each employing its own underlying modal structure for the narrow frequency band that it operates over. The Bayesian solution using state-space forward propagators is developed and shown that each processor is decoupled in modal space and recombined in the measurement space to provide enhanced estimates [28, 29].

The methodology employed is based on a state-space representation of the normal-mode propagation model [28]. When state-space representations can be accomplished, then many of the current ocean acoustic processing problems can be analyzed and solved using this more revealing and intuitive framework which is based on firm statistical and system theoretic grounds. In this application, we seek techniques to incorporate the: (1) ocean acoustic propagation model; (2) sensor array measurement model; and (3) noise models (ambient, shipping, surface and measurement) into the processor to solve the associated signal enhancement problem.

10.2.2 Broadband State–Space Ocean Acoustic Propagators

In this section we discuss the development of a broadband propagator eventually employed in a Bayesian scheme to enhance noisy pressure-field measurements from a vertical array of hydrophone sensors. First, we briefly discuss the propagator from normal-mode theory following the Green's function approach [32] and then extend it using a state–space representation to develop a forward propagation scheme for eventual use in Bayesian processor design.

It is well-known [32–34] in ocean acoustics that the pressure-field *solution* to the Helmholtz equation under the appropriate assumptions can be expressed as the sum

of normal modes

$$p(r,z,t) = \sum_{m=1}^{M} a\mathcal{H}_o(\kappa_r(m)r)\phi_m(z_s)\phi_m(z)e^{\imath\omega_o t} \tag{10.5}$$

where p is the acoustic pressure-field; a is the source amplitude; $\mathcal{H}_o$ is the zeroth-order Hankel function; ϕ_m is the m^{th} modal function evaluated at z and source depth z_s; $\kappa_r(m)$ is the horizontal wave number associated with the m^{th} mode; ω_o is the temporal source frequency, and r is the horizontal range. The wave numbers satisfy the corresponding *dispersion relation*

$$\kappa^2 = \frac{\omega^2}{c^2(z)} = \kappa_r^2(m) + \kappa_z^2(m), \quad m = 1,\ldots,M \tag{10.6}$$

where κ_z is the vertical wave number with c the depth-dependent sound speed profile. Taking the temporal Fourier transform of the pressure-field, we obtain

$$p(r,z,\omega) = \sum_{m=1}^{M} a\mathcal{H}_o(\kappa_r(m)r)\phi_m(z_s,\omega)\phi_m(z,\omega)\delta(\omega-\omega_o) \tag{10.7}$$

indicating a narrowband solution or equivalently a line at ω_o in the temporal frequency domain.

This modal representation has been extended to include a broadband source, $s(t)$, with corresponding spectrum, $S_s(\omega)$. In this case, the ocean medium is specified by its *Green's* function (impulse response) which can be expressed in terms of the inherent normal modes spanning the water column

$$\mathcal{G}(r,z,\omega) = \sum_{m} \mathcal{H}_o(\kappa_r(m)r)\phi_m(z_s,\omega)\phi_m(z,\omega) \tag{10.8}$$

and therefore the resulting pressure-field in the temporal frequency domain is given by

$$p(r,z,\omega) = \mathcal{G}(r,z,\omega)S(\omega) \tag{10.9}$$

which equivalently corresponds to a convolution of $\mathcal{G}(r,z,t)$ and $s(t)$ in the time domain.

Suppose we decompose the continuous source spectrum into a sampled or discrete spectrum using a periodic impulse (frequency) sampler, then it follows that

$$S_s(\omega) = \triangle\omega \sum_{q} S(\omega)\delta(\omega-\omega_q) = \triangle\omega \sum_{q} S(\omega_q)\delta(\omega-\omega_q) \tag{10.10}$$

from the sampling properties of Fourier transforms. Therefore a broadband signal spectrum can be decomposed into a set of narrowband components assuming an impulse sampled spectrum.

Utilizing this property in Eq. 10.10 and extracting just the q^{th} source frequency, we have that

$$p(r, z, \omega_q) = \mathcal{G}(r, z, \omega) S_s(\omega_q) \tag{10.11}$$

where $S(\omega_q)$ can be interpreted as a single narrowband impulse at ω_q with amplitude, $a_q = \triangle\omega |S(\omega_q)|$. Suppose that the broadband source is assumed to be bandlimited and sampled $S_s(\omega),\ \omega_1 \leq \omega \leq \omega_Q$ then

$$S_s(\omega) = \triangle\omega \sum_{q=1}^{Q} S(\omega_q)\delta(\omega - \omega_q) \tag{10.12}$$

Thus, the normal-mode solution to the Helmholtz equation for the broadband source problem can be decomposed into a series of narrowband solutions, that is,

$$p(r, z, \omega_q) = \sum_{m=1}^{M_q} a_q \mathcal{H}_o(\kappa_r(m, q)r)\phi_m(z_s, \omega_q)\phi_m(z, \omega_q) \tag{10.13}$$

and the dispersion relation now satisfies

$$\kappa_r^2(m, q) = \frac{\omega_q^2}{c^2(z)} - \kappa_z^2(m, q), \quad m = 1, \ldots, M_q;\ \ q = 1, \ldots, Q \tag{10.14}$$

This overall decomposition of the field into narrowband components could lead to an enhancer averaging each narrowband component of source frequency and would be the superposition of the pressure-field of Eq. 10.13 given by

$$p(r, z) = \sum_{q=1}^{Q} p(r, z, \omega_q) \tag{10.15}$$

assuming an *incoherent* approach as depicted in Fig. 10.11.

Suppose we further assume an L-element vertical sensor array, then $z \rightarrow z_\ell$, $\ell = 1, \ldots, L$ and therefore, the pressure-field at the array for the q^{th} temporal frequency of Eq. 10.13 becomes

$$p(r, z_\ell, \omega_q) = \sum_{m=1}^{M_q} \beta_m(r, z_\ell, \omega_q)\phi_m(z_\ell, \omega_q) \tag{10.16}$$

where $\beta_m(r, z_\ell, \omega_q) := a_q \mathcal{H}_o(\kappa_r(m, q)r)\phi_m(z_s, \omega_q)$ is the m^{th}-modal coefficient at the q^{th} temporal frequency.

Thus, following Eq. 10.15 the broadband pressure-field at the ℓ^{th}-sensor is simply

$$p(r, z_\ell) = \sum_{q=1}^{Q} p(r, z_\ell, \omega_q) \tag{10.17}$$

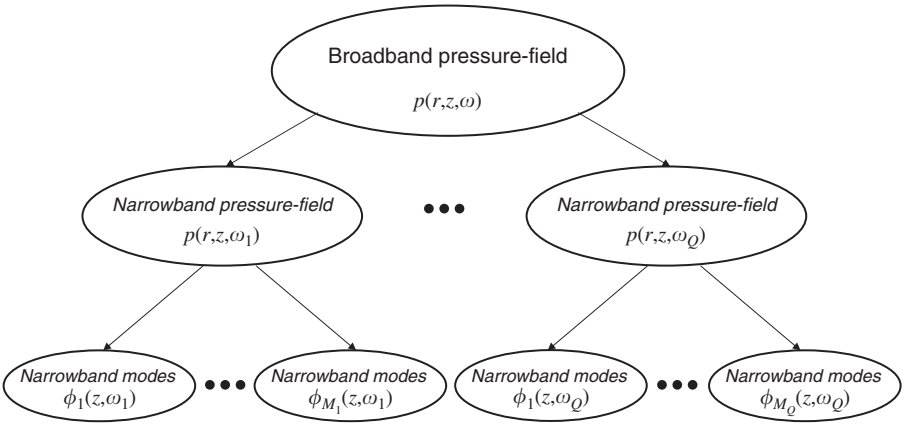


FIGURE 10.11 Decomposition of broadband signal into narrowband components based on normal-mode propagation representation.

which corresponds to incoherently summing all of the narrowband solutions over the discrete spectral lines (bins) [25].

Thus, the amount of spectral energy "seen" at the ℓ^{th}-sensor in the q^{th}-spectral bin ($\omega_q = q\triangle\omega$) is defined by $p(z_\ell, \omega_q)$ and therefore the total energy seen by the array in the q^{th}-bin is given by the set of sensor outputs, $\overline{P}(\omega_q) \equiv \{p(z_1, \omega_q), \ldots p(z_L, \omega_q)\}$. This implies that the subsequent processor must be able to process Q temporal frequencies over the array of L-sensors or must have LQ processors. One obvious technique would be to collect the array snapshots at each frequency and incoherently average the results, that is,

$$\overline{P}(\omega) = \frac{1}{Q}\sum_{q=1}^{Q}\overline{P}(\omega_q) \tag{10.18}$$

Using this approach implies that the "temporal incoherent processor" replaces the noisy broadband signal $P(\omega)$ with the smoothed signal $\overline{P}(\omega)$.

It is well known [29] that the state–space propagators for the narrowband pressure-field can be obtained from the relationship

$$p(r, z, t) = \mu(r)\phi(z)e^{-j\omega_o t}. \tag{10.19}$$

Assuming that the source range is known ($r = r_s$) and transforming to the temporal frequency domain, we have that

$$p(z, \omega_o) = \mathcal{H}_o(\kappa_r r_s)\phi(z)\delta(\omega - \omega_o) \tag{10.20}$$

with $\mathcal{H}_o(\kappa_r r_s)$ the zeroth-order Hankel function. If we sample the spatial pressure-field with a vertical line array of hydrophones as before, then we have

$$p(z_\ell, \omega_o) = [\beta_1(r_s, z_s, \omega_o)\ \ 0|\ldots|\beta_M(r_s, z_s, \omega_o)\ \ 0]\phi(z_\ell, \omega_o) \tag{10.21}$$

with $\beta_m(r_s, z_s, \omega_o) = \mathcal{H}_o(\kappa_r r_s)\phi_m(z_s, \omega_o)$ or simply

$$p(z_\ell, \omega_o) = C^T(r_s, z_s, \omega_o)\phi(z_\ell, \omega_o) \tag{10.22}$$

In terms of these models, it has been shown [29] that the *broadband state–space propagator* can be expressed as

$$\begin{aligned} \frac{d}{dz}\mathbf{\Phi}(z,\omega) &= \mathbf{A}(z,\omega)\mathbf{\Phi}(z,\omega) \\ p(z_\ell,\omega) &= \mathbf{C}^T(r_s, z_s, \omega)\mathbf{\Phi}(z_\ell,\omega) \end{aligned} \tag{10.23}$$

where $\mathbf{\Phi}(z,\omega) \in \mathbf{R}^{2M\times 1}$, $\mathbf{A}(z,\omega) \in \mathbf{C}^{2M\times 2M}$, $\mathbf{C}^T(z,\omega) \in \mathbf{R}^{1\times 2M}$ with $M \equiv \sum_{q=1}^{Q} M_q$. Here we have implicitly assumed an *incoherent* sum over the set of temporal frequencies for our pressure-field measurement model. Note we use the parameter "ω" to signify the entire set of discrete temporal frequencies, $\{\omega_q\}, q = 1, \ldots, Q$.

The internal structure of this overall processor admits the following decomposition:

$$\frac{d}{dz}\mathbf{\Phi}(z,\omega) = \begin{bmatrix} \mathbf{A}(z,\omega_1) & O & \cdots & O \\ O & \mathbf{A}(z,\omega_2) & \cdots & O \\ \vdots & & \ddots & \vdots \\ O & O & \cdots & \mathbf{A}(z,\omega_Q) \end{bmatrix} \begin{bmatrix} \mathbf{\Phi}(z,\omega_1) \\ \mathbf{\Phi}(z,\omega_2) \\ \vdots \\ \mathbf{\Phi}(z,\omega_Q) \end{bmatrix} \tag{10.24}$$

and $\mathbf{A}(z,\omega_q) = diag[\mathbf{A}_1(z,\omega_q) \ldots \mathbf{A}_{M_q}(z,\omega_q)] \in \mathbf{R}^{2M_q\times 2M_q}$ and

$$\mathbf{A}_m(z,\omega_q) = \begin{bmatrix} 0 & 1 \\ -\kappa_z^2(m,q) & 0 \end{bmatrix}, \quad m = 1, \ldots, M_q \tag{10.25}$$

with the *incoherent* pressure-field sensor measurement model given by (see Fig. 10.11)

$$p(z_\ell,\omega) = [\mathbf{C}^T(r_s, z_s, \omega_1) \ldots \mathbf{C}_Q^T(r_s, z_s, \omega_Q)] \begin{bmatrix} \mathbf{\Phi}(z_\ell,\omega_1) \\ \vdots \\ \mathbf{\Phi}(z_\ell,\omega_Q) \end{bmatrix} \tag{10.26}$$

This deterministic model can be extended to a stochastic Gauss-Markov representation [11] given by

$$\begin{aligned} \frac{d}{dz}\mathbf{\Phi}(z,\omega) &= \mathbf{A}(z,\omega)\mathbf{\Phi}(z,\omega) + \mathbf{w}(z,\omega) \\ p(z_\ell,\omega) &= \mathbf{C}^T(z,\omega)\mathbf{\Phi}(z_\ell,\omega) + v(z,\omega) \end{aligned} \tag{10.27}$$

where $\mathbf{w}$, v are additive, zero-mean Gaussian noise sources with respective spectral covariance matrices $\mathbf{R}_{ww}(z,\omega)$, and $\mathbf{R}_{vv}(z,\omega)$. Next we investigate the internal structure of the broadband Bayesian processor.

10.2.3 Broadband Bayesian Processing

The broadband pressure-field can be characterized by the function, $p(\mathbf{z}, \omega)$. If we assume that the field is measured by a vertical array, then at each sensor the acquired time series is given by $p(z_\ell, t)$. Clearly, this sensor measurement contains all of the information about the source, both temporally and spatially, but due to the complexity of this received data coupled with the noise, the required signal processing is quite a challenging problem. If we take the Fourier transform in the temporal domain, then the broadband source can be thought of as being viewed through a bank of narrowband temporal filters, that is, the broadband pressure-field surface or image is then given by: $p(\mathbf{z}, \omega) \rightarrow p(z_\ell, \omega_q)$; for $\ell = 1, \ldots, L$, and $q = 1, \ldots, Q$. Once this surface is constructed, it is possible to infer other useful information about the dynamics of the ocean as well as the source. If we assume further that we have a shallow water environment characterized by trapped wave propagation, then we may represent the underlying Green's function or channel impulse response in terms of a normal-mode model. In this case it may also be of interest to observe the surface created by the various broadband modal functions, that is, as a function of temporal frequency, $\phi_m(z_\ell, \omega_q)$; for $m = 1, \ldots, M_q$. It is clear that as we decompose this complex temporal pressure-field measurement into these narrow frequency bands, each band will contain oceanographical and source information. Thus, the problem that we address first here, is that of receiving a set of temporal noisy broadband pressure-field measurements and developing the enhancement necessary to construct the surfaces created by both the broadband pressure-field and modal functions.

With this problem in mind, we now recast our incoherent measurement model of Eq. 10.26 into that of a broadband system obtained by "stacking" all of the narrowband pressure-field measurements, rather than performing the coherent or incoherent addition. The resulting vector *broadband pressure-field measurement model* at the ℓ^{th} vertical sensor is now given by

$$\mathbf{p}(z_\ell, \omega) = \mathbf{C}^T(r_s, z_s, \omega)\mathbf{\Phi}(z_\ell, \omega) + \mathbf{v}(z_\ell, \omega) \tag{10.28}$$

where $\mathbf{p}, \mathbf{v} \in C^{Q\times 1}$, $\mathbf{C}^T \in R^{Q\times 2M}$, and $\mathbf{\Phi} \in R^{2M\times 1}$. Expanding this model over the set of discrete temporal frequencies $\{\omega_q\}, q = 1, \ldots, Q$, we have

$$\begin{bmatrix} p(z_\ell, \omega_1) \\ \vdots \\ p(z_\ell, \omega_Q) \end{bmatrix} = \begin{bmatrix} C^T(z_\ell, \omega_1) & \cdots & O \\ \vdots & \ddots & \vdots \\ O & \cdots & C^T(z_\ell, \omega_Q) \end{bmatrix} \begin{bmatrix} \mathbf{\Phi}(z_\ell, \omega_1) \\ \vdots \\ \mathbf{\Phi}(z_\ell, \omega_Q) \end{bmatrix} + \begin{bmatrix} v(z_\ell, \omega_1) \\ \vdots \\ v(z_\ell, \omega_Q) \end{bmatrix} \tag{10.29}$$

We assume that the corresponding measurement spectral covariance is given by:

$$\mathbf{R}_{vv}(z_\ell, \omega) = diag[R_{vv}(z_\ell, \omega_1) \ldots R_{vv}(z_\ell, \omega_Q)] \tag{10.30}$$

In order to obtain the optimal (minimum error variance) estimator, we cast our problem into a probabilistic framework under these Gauss-Markov assumption;

therefore, our sequence of pressure-field measurements at each sensor are Fourier transformed to yield a discrete set of frequencies $\{\omega_q\}, q = 1, \ldots, Q$ and the set of broadband sensor measurements, $\{\mathbf{p}(z_\ell, \omega)\}, \ell = 1, \ldots, L; \mathbf{p} \in C^{Q\times 1}$. Note that the window length of the Fourier transform is determined by the temporal correlation time of the measurement (source) to assure the independence of the frequency samples. For our pressure-field/modal function estimation problem, we define the underlying *broadband pressure-field/modal function enhancement problem* as:

GIVEN a set of noisy broadband pressure-field measurements, $\{\mathbf{p}(z_\ell, \omega)\}, \ell = 1, \ldots, L$ and the underlying Gauss-Markov model (previous section), FIND the best (minimum error variance) estimate of the broadband modal functions, $\hat{\Phi}(z_\ell, \omega)$, and pressure-field, $\hat{\mathbf{p}}(z_\ell, \omega)$.

The Bayesian solution to this problem can be obtained by the maximizing the *a posteriori* density. Define the set of broadband pressure-field measurements as: $P_L \equiv \{\mathbf{p}(z_1, \omega), \ldots, \mathbf{p}(z_L, \omega)\}; \mathbf{p} \in C^{Q\times 1}$, then the maximum a-posteriori (*MAP*) estimator of the modal functions must maximize the posterior density given by

$$\Pr(\mathbf{\Phi}(z_\ell, \omega)|P_\ell) = \frac{\Pr(\Phi(z_\ell, \omega), P_\ell)}{\Pr(P_\ell)} \tag{10.31}$$

but it follows directly from Bayes' rule that we have

$$\Pr(\mathbf{\Phi}(z_\ell, \omega)|P_\ell) = \frac{\Pr(\mathbf{p}(z_\ell, \omega)|\mathbf{\Phi}(z_\ell, \omega), P_{\ell-1}) \times \Pr(\mathbf{\Phi}(z_\ell, \omega)|P_{\ell-1})}{\Pr(\mathbf{p}(z_\ell, \omega)|P_{\ell-1})} \tag{10.32}$$

Under Gauss-Markov assumptions, we have that

$$\begin{aligned}
\Pr(\mathbf{p}(z_\ell, \omega)|P_{\ell-1}) &\sim \mathcal{N}(\mathbf{C}(r_s, z_s, \omega)\hat{\mathbf{\Phi}}(z_{\ell|\ell-1}, \omega), \\
&\qquad \mathbf{C}(r_s, z_s, \omega)\widetilde{\mathbf{P}}(z_{\ell|\ell-1}, \omega)\mathbf{C}^T(r_s, z_s, \omega) + \mathbf{R}_{vv}(z_{\ell-1}, \omega)) \\
\Pr(\mathbf{p}(z_\ell, \omega)|\mathbf{\Phi}(z_\ell, \omega), P_{\ell-1}) &\sim \mathcal{N}(\mathbf{C}(r_s, z_s, \omega)\mathbf{\Phi}(z_\ell, \omega), \mathbf{R}_{vv}(z_{\ell-1}, \omega)) \\
\Pr(\mathbf{\Phi}(z_\ell, \omega)|P_{\ell-1}) &\sim \mathcal{N}(\hat{\mathbf{\Phi}}(z_{\ell|\ell-1}, \omega), \widetilde{\mathbf{P}}(z_{\ell|\ell-1}, \omega))
\end{aligned} \tag{10.33}$$

where the modal estimation error covariance is given by [29]

$$\widetilde{\mathbf{P}}(z_{\ell|\ell-1}, \omega) = \mathbf{A}(z_{\ell-1}, \omega)\widetilde{\mathbf{P}}(z_{\ell-1|\ell-1}, \omega)\mathbf{A}^T(z_{\ell-1}, \omega) + \mathbf{R}_{ww}(z_{\ell-1}, \omega) \tag{10.34}$$

Here the notation $\hat{\mathbf{\Phi}}(z_{\ell|\ell-1}, \omega) \equiv E\{\mathbf{\Phi}(z_\ell, \omega)|P_{\ell-1}\}$ is the conditional mean, that is, the "best" (minimum variance) estimate at depth z_ℓ based on the previous sensor measurement up to the depth $z_{\ell-1}$.

Now substituting these distributions into the *a posteriori* density and performing the necessary manipulations [17], we obtain the desired relations. That is, by

maximizing the *a posteriori* density or equivalently its logarithm, we obtain the *MAP equation*

$$\frac{\partial}{\partial \Phi} ln \Pr(\mathbf{\Phi}(z_\ell, \omega)|P_\ell)\Big|_{\mathbf{\Phi}=\hat{\mathbf{\Phi}}_{MAP}} = \mathbf{0} \tag{10.35}$$

Differentiating the posterior density, setting the result to zero and solving for $\Phi(\cdot)$ gives the desired *MAP* estimator [29]

$$\hat{\mathbf{\Phi}}(z_{\ell|\ell}, \omega) = \hat{\mathbf{\Phi}}(z_{\ell|\ell-1}, \omega) + \mathbf{K}(z_\ell, \omega)\epsilon(z_\ell, \omega) \tag{10.36}$$

where $\hat{\mathbf{\Phi}}(z, \omega_q) \in R^{2M\times 1}$, $\mathbf{K}(z_\ell, \omega) \in \mathbf{C}^{2M\times Q}$, and $\epsilon(z_\ell, \omega) \in \mathbf{C}^{Q\times 1}$ is the corrected estimate (below) and shown in the linear space-varying broadband Kalman filter algorithm. The overall structure of the estimator can be seen by expanding the gain matrix over the set of discrete frequencies for $\omega \to \omega_q$, one for each of the Q-columns to give

$$\hat{\mathbf{\Phi}}(z_{\ell|\ell}, \omega) = \hat{\mathbf{\Phi}}(z_{\ell|\ell-1}, \omega) + \sum_{q=1}^{Q} K(z_\ell, \omega_q)\epsilon(z_\ell, \omega_q) \tag{10.37}$$

where $\hat{\mathbf{\Phi}}(z, \omega) \in R^{2M\times 1}$, $\mathbf{K}(z_\ell, \omega_q) \in \mathbf{C}^{2M\times 1}$, and $\epsilon(z_\ell, \omega_q) \in \mathbf{C}^{1\times 1}$. Now let us rewrite this equation in a slightly different manner by expanding over the set of discrete frequencies and expressing the gain in terms of $2M_q \times Q$ block rows $\mathbf{K}_q^T(z_\ell, \omega)$ which have been decomposed further into its individual column vectors defined by $\mathbf{K}_{qn}(z_\ell, \omega_n) \in \mathbf{C}^{2M_q\times 1}$ to obtain the *narrowband recursion* for the corrected estimate as

$$\hat{\mathbf{\Phi}}(z_{\ell|\ell}, \omega_q) = \hat{\mathbf{\Phi}}(z_{\ell|\ell-1}, \omega_q) + \sum_{n=1}^{Q} \mathbf{K}_{qn}(z_\ell, \omega_n)\epsilon(z_\ell, \omega_n) \tag{10.38}$$

with $\hat{\mathbf{\Phi}}(z_{\ell|\ell}, \omega_q) \in \mathbf{C}^{2M_q\times 1}$ the gain $\mathbf{K}_{qn}(z_\ell, \omega_n) \in \mathbf{C}^{2M_q\times 1}$, and $\epsilon(z_\ell, \omega_n) \in \mathbf{C}^{1\times 1}$ showing how each narrowband frequency line ω_q can be combined to form the optimal *MAP* estimate.

These relations suggest an efficient parallel, but *suboptimal* approach to implementing this broadband processor might be achieved by constructing a "local" narrowband processor for each spectral line ω_q and then combining their outputs to obtain the final broadband estimate, that is,

$$\hat{\mathbf{\Phi}}(z_{\ell|\ell}, \omega_q) = \hat{\mathbf{\Phi}}(z_{\ell|\ell-1}, \omega_q) + \mathbf{K}_{qq}(z_\ell, \omega_q)\epsilon(z_\ell, \omega_q) \tag{10.39}$$

where we have discarded the other $2M_q \times 1$ submatrices, $\mathbf{K}_{qn}(z_\ell, \omega_n) = \mathbf{0}\, n \neq q$ to give

$$\mathbf{K}_{qq}(z_\ell, \omega_q)\frac{\widetilde{\mathbf{P}}(z_{\ell|\ell-1}, \omega_q)\mathbf{C}^T(r_s, z_s, \omega_q)}{\mathbf{R}_{\epsilon\epsilon}(z_\ell, \omega_q)}$$

The structure of the suboptimal broadband implementation of the *BP* is illustrated in Fig. 10.12. Thus the optimal algorithm will consist of a bank of narrowband

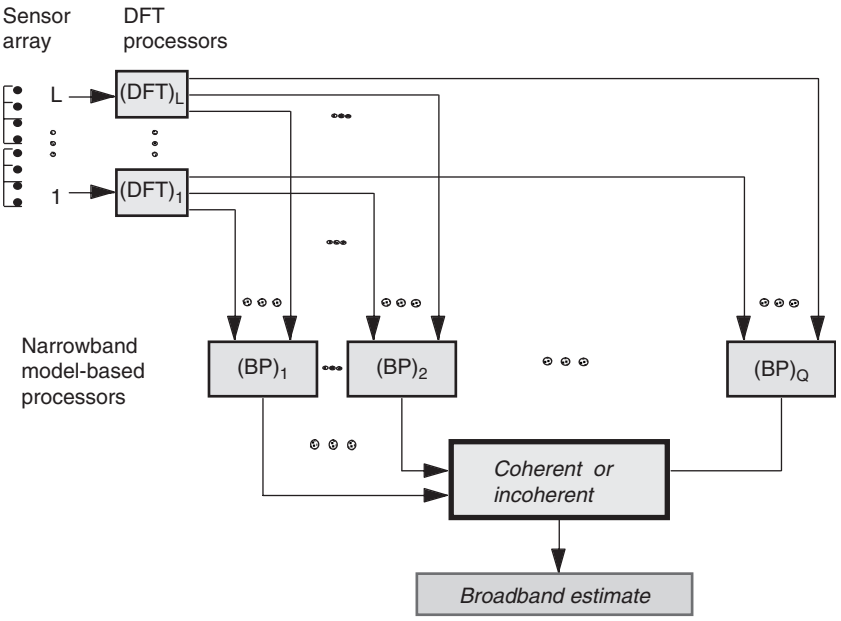


FIGURE 10.12 Structure of the broadband Bayesian processor using narrowband modal/pressure-field decomposition.

Bayesian predictors combined during the update (correction) phase of the algorithm to create the broadband *MAP* estimator. The algorithm then proceeds for each $\{\omega_q\}, q = 1, \ldots, Q$ as:

Prediction:

$$\frac{d}{dz}\hat{\mathbf{\Phi}}(z, \omega_q) = \mathbf{A}(z, \omega_q)\hat{\mathbf{\Phi}}(z, \omega_q)$$

$$\hat{p}(z_\ell, \omega_q) = \mathbf{C}^T(r_s, z_s, \omega_q)\hat{\mathbf{\Phi}}(z_{\ell|\ell-1}, \omega_q)$$

Innovation:

$$\epsilon(z_\ell, \omega_q) = p(z_\ell, \omega_q) - \hat{p}(z_\ell, \omega_q)$$

Correction:

$$\hat{\mathbf{\Phi}}(z_{\ell|\ell}, \omega_q) = \hat{\mathbf{\Phi}}(z_{\ell|\ell-1}, \omega_q) + \sum_{n=1}^{Q} \mathbf{K}_{qn}(z_\ell, \omega_n)\epsilon(z_\ell, \omega_n)$$

Gain:

$$\mathbf{K}(z_\ell, \omega) = \widetilde{\mathbf{P}}(z_{\ell|\ell-1}, \omega)\mathbf{C}^T(r_s, z_s, \omega_q)\mathbf{R}_{\epsilon\epsilon}^{-1}(z_\ell, \omega) \tag{10.40}$$

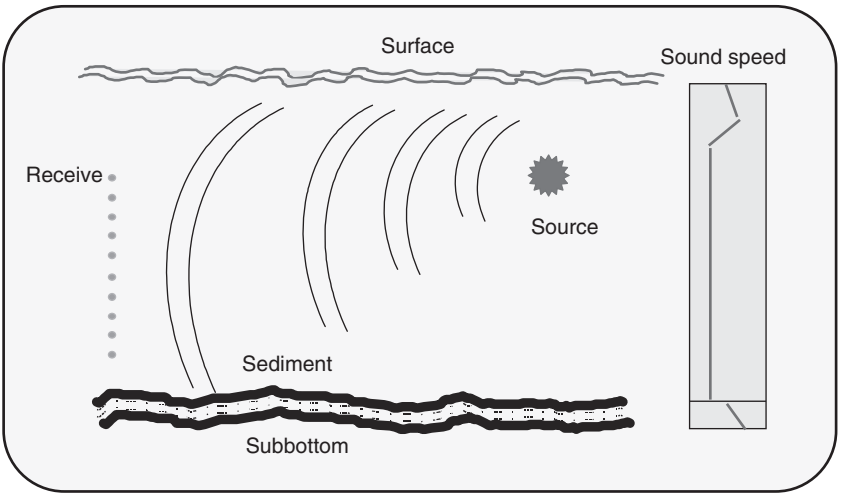


FIGURE 10.13 Shallow ocean environment problem: channel (100 m) with broadband (50 *Hz*) source located at range, $r_s = 10\,Km$ and depth, $z_s = 50\,m$.

This completes the development of the broadband Bayesian processor. Next we consider its application.

10.2.4 Broadband *BSP* Design

In this subsection we discuss the application of the Bayesian processor to data synthesized by a broadband normal-mode model using the state–space forward propagator and the underlying Gauss-Markov representation.

Let us consider a basic shallow water channel depicted in Fig. 10.13. We assume a flat bottom, range independent three layer environment with a channel depth of $100\,m$, a sediment depth of $2.5\,m$ and a subbottom. A vertical line array of 100-sensors with spacing of $\triangle z = 1\,m$ spans the entire water column and a broadband source of unit amplitude and $50\,Hz$ bandwidth ranging from 50–$100\,Hz$ in $10\,Hz$ increments is located at a depth of $50\,m$ and a range of $10\,Km$ from the array. The sound speed profile in the water column and the sediment are sketched in the figure and specified along with the other problem parameters. SNAP, a normal-mode propagation simulator [35] is applied to solve this shallow water problem and executed over the set of discrete temporal source frequencies. This boundary value problem was solved using SNAP and the results at each narrowband frequency are shown [29]. We note that as the temporal frequency increases, the *number* of modes increases thereby increasing the corresponding order of the state–space. Details of the problem parameters are given in [29].

The parameters obtained from SNAP are now used to construct the broadband state–space and measurement models of the previous subsection. Here we use the set of horizontal wave numbers, $\{\kappa(m,q)\}, m = 1,\ldots,M_q; q = 1,\ldots,Q$, and sound speed, $\{c(z_\ell)\}$, to implement the state models along with the corresponding modal function values, $\{\phi_{m1}(z_s,\omega_q)\}$, as well as the Hankel functions, $\{\mathcal{H}_o(\kappa(m,q)r_s)\}$ to construct the measurement models.

The final set of parameters for our simulation are the modal and measurement noise covariance matrices required by the Gauss-Markov model (see [29] for more details).

It is important to realize that the state–space "forward" propagators do *not* offer an alternative solution to the Helmholtz equation (not to be confused with a marching method), but rather use the parameters *from* the boundary value solution to obtain a set of initial conditions/parameters for the propagator construction. Even the adaptive Bayesian processors still utilize the boundary value solutions to "initialize" the processing [28].

With this information in hand, the Gauss-Markov simulation was performed at $SNR_{in} = 25\,dB$ (noise free) and $SNR_{out} = -30\,dB$. The "true" pressure-field surface is shown in Fig. 10.14 along with the corresponding noisy surface—both produced

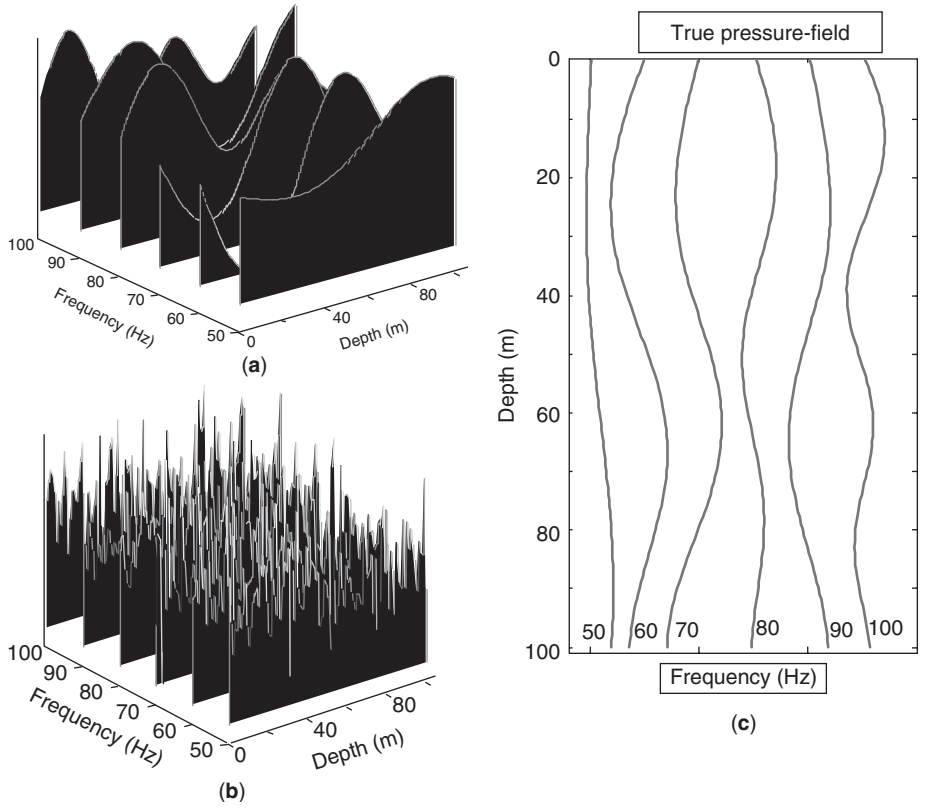


FIGURE 10.14 Synthesized broadband pressure-field surface: (a) True pressure-field. (b) Noisy (−30 dB) pressure-field. (c) Narrowband DFT filter outputs of true field.

as outputs of a set of narrowband DFT filters at each ω_q. We also show the true pressure-field functions which are expected to be extracted by the optimal processor along with the modal function estimates.

The optimal Bayesian or minimum variance processor was designed using the identical set of parameters used in the Gauss-Markov simulation thereby eliminating any "mismatch" between model and environment. We can consider this simulation as a bound on the best one could hope to achieve, since it is in fact the minimum variance estimates satisfying the Cramer-Rao lower bound. In minimum variance estimation it is important to realize the overall design philosophy. First, the key issue is that a necessary and sufficient condition for optimality is that the innovation sequences (difference between measured and predicted pressure-fields) are zero-mean and uncorrelated (white). Thus, actual minimum variance designs are *not* considered "tuned" unless this condition is satisfied; therefore, the free parameters in the processor (usually initial conditions and process/measurement noise vectors/matrices) are adjusted (if possible) until this condition is achieved. Once satisfied, then and only then can the state (modal function) and measurement (pressure-field) estimates along with their associated covariances be considered viable.

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 this feasibility test for the broadband processor design on synthesized data. There are other tests that can be used with real data to check the consistency of the processor (see Chapter 5 for more details).

10.2.5 Results

We use *SSPACK_PC* [30], a Bayesian processing toolbox available in *MATLAB* [31] to design our broadband *MBP*. The results of the minimum variance design are shown in Fig. 10.15, where we see the enhanced pressure-field and the corresponding innovations sequences at each discrete temporal frequency as a function of depth. Each of the innovations sequences tested zero-mean and white with the following test results: 50 Hz: (2% out; $0.08 < 7.5$); 60 Hz: (2% out; $0.98 < 2.5$); 70 Hz: (3.9% out; $0.66 < 2.6$); 80 Hz: (2% out; $0.70 < 3.8$); 90 Hz: (0% out; $0.70 < 1.9$); 100 Hz: (2% out; $1.90 < 3.4$). The corresponding *WSSR* statistic lies below the threshold in Fig. 10.16. Thus we have (as expected) achieved a minimum variance "broadband" design. Note that the enhanced pressure-field estimate at each temporal frequency, $\omega_q = \{50, 60, 70, 80, 90, 100\}$, is governed by the Gauss-Markov model of the previous subsection. The corresponding modal functions were then extracted from the noisy pressure-field producing viable estimates. The estimated modal functions correspond to the two (2) modes at $50\,Hz$ and three (3) at $60\,Hz$. The other estimated modes (from the noisy data) are at $70\,Hz$ (3 modes), $80\,Hz$ (4 modes), $90\,Hz$ (4 modes) and $100\,Hz$ (5 modes). Again note that the modal estimates $\hat{\mathbf{\Theta}}(z_\ell, \omega_q)$ along with the measurement model at each temporal frequency, $C^T(r_s, z_s, \omega_q)$ are used to construct the pressure-field, $\hat{\mathbf{p}}(z_\ell, \omega_q)$ at each temporal frequency solving the enhancement problem.

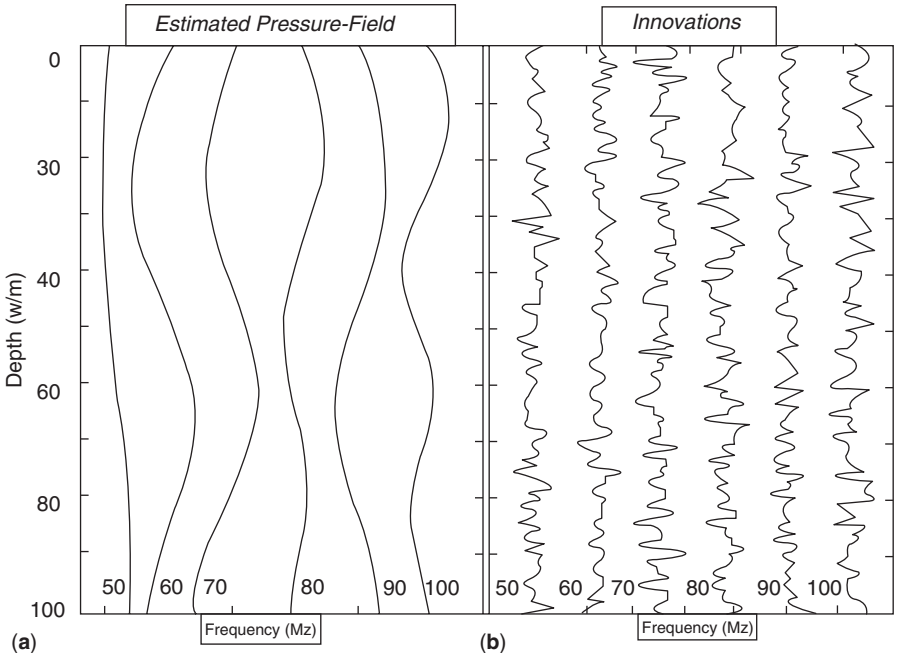


FIGURE 10.15 Broadband Bayesian processor design: (a) Enhanced pressure-field estimates. (b) Innovation sequences.

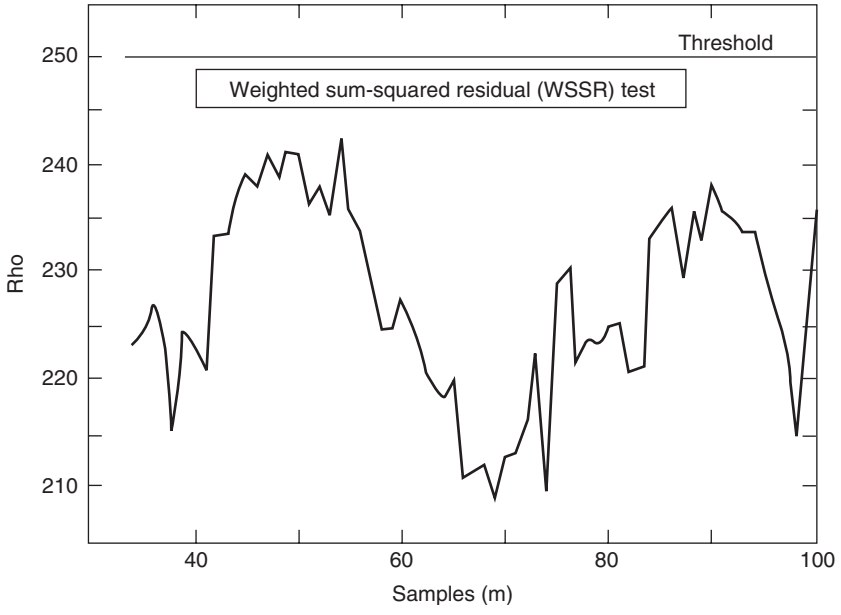


FIGURE 10.16 Innovation sequence whiteness testing: WSSR statistic (Window = 35 samples; Threshold = 250.

This completes the application of the broadband Bayesian processor designed to enhance the pressure-field surface and extract the corresponding modal functions.

10.3 BAYESIAN PROCESSING FOR BIOTHREATS

The design of a "smart" physics-based processor for microcantilever sensor arrays to detect various target species in solution based on the deflections of a functionalized array is discussed. A proof-of-concept design is demonstrated and shown to perform quite well on experimental data.

10.3.1 Background

Smart sensors with embedded processors offer unique advantages for applications that must gather large amounts of data and continuously monitor evolving conditions for potential changes (e.g., machine condition monitoring) or potential threats (e.g., biological, chemical, nuclear). Unfortunately, the usual processing techniques such as nonparametric methods like wavelets or parametric methods like autoregressive-moving average (*ARMA*) models do not capture the true essence of the problem physics required to extract the desired information, detect the change or monitor the environment for threats. The underlying physical phenomenology governing the propagation physics is usually quite complex governed by nonlinearities typically characterized by nonlinear differential/difference equations. Coupling the resulting nonlinear processor to the sensor performing the measurement has not been considered a realistic possibility until now with the evolution of high-speed microcomputer chips that can easily be incorporated into the sensor design. We consider the design of an algorithm coupled to a microelectromechanical sensor (MEMS) to estimate the presence of critical materials or chemicals in solution.

Microcantilevers are powerful transducers for sensing inorganic, organic and biological molecules, since they readily bend or deflect in the presence of a very small number of target molecules (nanomolar to femtomolar concentrations) [36] as shown in Fig. 10.17. The number of potential target chemicals is large, ranging from DNA [37] to explosives [38], implying that these sensors may be useful in defense, medicine, drug discovery, and environmental monitoring. Microcantilevers are capable of recognizing antibodies [39] and nerve agent products (hydrofluoric acid) in solution [40]. However, a major limitation of these sensors is that their signal-to-noise ratio (*SNR*) is low in many operational environments of interest. Therefore, we discuss the design of a "smart sensor" design combining the array with a physics-based processor to minimize its inherent limitations and maximize the output *SNR* for enhancement.

We investigate the physics-based Bayesian approach [17] to develop a multichannel processor evolving into a smart sensor for this application. This approach is essentially incorporating mathematical models of both physical phenomenology (chemistry/flow dynamics) and the measurement process (cantilever array including noise) into the processor to extract the desired information. In this way the resulting Bayesian signal processor (*BSP*) enables the interpretation of results directly in terms of the problem physics.

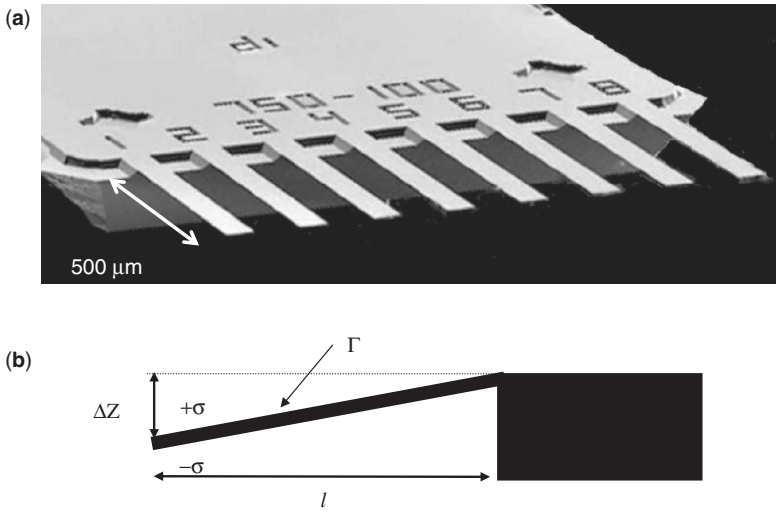


FIGURE 10.17 Micromachined cantilever array: (a) Eight(8)-element lever array. (b) Lever deflection.

We discuss the design of physics-based signal processing to micromachined cantilever measurement arrays to estimate the critical materials in solution. We briefly present the underlying physical phenomenology and reduce it to a simple model for processor development. Unknown parameters in this model are "fit" from independent experimental data. Once these parameters are estimated, we use minimum error variance techniques for the *BSP* design [17]. We then apply the resulting processor to experimental data demonstrating the overall enhancement that would lead to an eventual "smart sensor" design. The resulting processor is based on nonlinear evolution equations leading to an extended Bayesian processor (XBP) or classically, the extended Kalman filter, (*EKF*) that can be implemented in an on-line manner yielding the enhanced data as its output.

10.3.1.1 Microcantilever Sensors The dynamics of the fluids flowing over the cantilever array of Fig. 10.17 is influenced by two major factors: temperature and flow. The temperature is dependent on many variables and the dynamics are relatively slow, creating a disturbance to the cantilever sensor system. Flow in the medium associated with the array induces a stress along with the chemical forces created by the molecules bonding with the functionalized levers leading to the measured deflection.

Micromachined cantilevers can function as detection devices when one side is fabricated to be chemically distinct from the other, as shown in Fig. 10.17b. Functionalization can be accomplished, for example, by evaporating a thin (10's of nm) film of metal such as gold on the top of the chip, then immersing the cantilever chip in a "probe" chemical that will bind preferentially to the thin gold film. The lever acts as a sensor when it is exposed to a second "target" chemical that reacts with the probe, since the reaction causes a free energy change that induces stress at the cantilever surface. Differential surface stress, $\triangle\sigma$, in turn, induces a deflection of the cantilever

that can be measured optically or electronically. In this subsection, we describe results of experiments with gold-coated cantilevers exposed to 2-mercaptoethanol, a small sulfur-terminated molecule with high affinity for gold.

When the microcantilever is immersed in a fluid and it has been functionalized to attract the target molecules, the changes in surface stress can be predicted as a function of surface loading. The evolution dynamics of this chemical interaction is captured by the well-known Langmuir kinetics in terms of a set of ordinary nonlinear differential equations. Rather than propagate these equations directly, we choose to use their solution in our processor that accounts for the adsorption–desorption kinetics. We developed an approximation to the Langmuir evolution equations based on a stirred tank reactor to estimate the target concentration as a function of time under continuous flow conditions. Experimentally, the applied chemical input signal is a constant concentration initiated by a step (function) increase at time, t_{ON}, and terminated at time, t_{OFF} [41].

Based on this representation of the process evolution physics, the dynamic (normalized) surface concentration of the interacting molecules on the surface of the cantilever, $\Gamma(t) = \frac{\widetilde{\Gamma}(t)}{\widetilde{\Gamma}_{MAX}}$, is given by the relations

$$\Gamma(t) = \begin{cases} 0 & t < t_{\text{ON}} \\ \left(\frac{c(t)}{c(t)+k_a/k_d}\right)\{1 - \exp\left[-(k_a c(t) + k_d)(t - t_{ON})\right]\} & t_{\text{ON}} \le t \le t_{\text{OFF}} \\ \sqrt{\frac{1}{2k_d(t-t_{OFF})}} & t > t_{\text{OFF}} \end{cases} \tag{10.41}$$

where k_a, k_d and $c(t)$ are the respective adsorption rate constant $[M]^{-1}s^{-1}$, desorption rate constant ($\text{cm}^{-2}M^{-1}s^{-1}$) and bulk concentration of the target molecules in solution (moles/liter) with $\widetilde{\Gamma}_{MAX}$, the maximum surface concentration of the species of interest ($\text{cm}^{-2}M^{-1}$).

The total free energy change of the cantilever surface, $\triangle G$, is related to the surface stress difference $\triangle\sigma$, between top and bottom side of the cantilever by

$$\Delta\sigma(t) = \Delta G(t)\Gamma(t)/M_A \tag{10.42}$$

where ΔG has units of J/mole and is the change in the sum of all of the contributions to the free energy of the surface of the cantilever with M_A is Avogadro's number (constant). The differential surface stress in the cantilever induces a chemically induced deflection, $\Delta z^C(t)$, using a variant of Stoney's equation [41], which implies that the deflection of the cantilever is directly proportional to the difference in surface stress (signal) on the cantilever surface relating this stress difference to the surface coverage and free energy of absorption, that is,

$$\Delta z^C(t) = \beta\Delta\sigma(t) \quad \text{for } \beta = \frac{3\ell^2(1-\nu)}{E\delta^2} \tag{10.43}$$

where E is the Young's modulus, ν is the Poisson's ratio, and ℓ and δ are the cantilever length and thickness, respectively. This model can be used to predict changes in surface stress as a function of surface loading.

The measurement model is more complicated, since it is the superposition of both the chemical and temperature deflection phenomena

$$y_\ell(t) = \Delta z_\ell^C(t) + \Delta z^T(t) \quad \text{for } \ell = 1, \ldots, L \tag{10.44}$$

where Δz_ℓ^C is the chemical deflection, different for different cantilevers, and $\Delta z^T(t)$ is the thermal deflection, assumed to be the same for all cantilevers. Since we have approximated the physics developing the well-founded formulation of Eq. 10.41, we know that there is uncertainty in both the measurements (noise) and the model parameters. Therefore, we cast the problem into a Gauss-Markov (*GM*) state-space framework [17] representing these uncertainties as additive Gaussian processes, that is,

$$\begin{aligned} \Delta G(t) &= \Delta G(t-1) + w(t-1) && \text{[state]} \\ y_\ell(t) &= \beta_\ell \Gamma(t;\Theta)\Delta G(t) + \Delta z^T(t) + v_\ell(t) && \text{[measurements]} \end{aligned} \tag{10.45}$$

where $\ell = 1, \ldots, L$; $\Gamma(t;\Theta)$ is given by Eq. 10.41 with unknown model parameters defined by the vector, $\Theta = [k_a \; k_d \; \widetilde{\Gamma}_{MAX}]$ and free energy (state) modeled as a random walk ($\Delta\dot{G}(t) = 0$). This representation, therefore, creates the foundation for our physics-based processor design. The process uncertainty is modeled by w and the corresponding measurement uncertainty as v, both zero-mean Gaussian with respective covariances, R_{ww} and R_{vv}. With this representation in mind, the cantilever signal enhancement problem is defined as:

GIVEN a set of noisy deflection measurements $\{y_\ell(t)\}$ with known bulk concentration inputs, $\{c(t)\}$ and unknown parameters Θ, FIND the best (minimum error variance) estimate of the deflection, $\hat{y}(t|t-1)$, that is, the conditional mean at t based on the data up to time $t-1$.

The design of the processor for this problem is illustrated in Fig. 10.18 and Fig. 1.4, 1.5. After the cantilever physics model is developed, it is used (1) to extract the required parameters using a physics-based parameter estimator; (2) to synthesize "data" for the initial processor designs; and (3) to enhance the noisy measurements being incorporated into the final *BSP* structure. In the figure, we see that the complex mathematical model of Eq. 10.41 (dashed box) is used to perform the physics-based parameter estimation using independent experimental data to extract the required parameters (adsorption/desorption rate constants and maximum concentration) as well as initially simulate data for *BP* design studies, once these parameters are extracted. The actual experimental data replaces the synthesized and is used to validate the processor performance.

10.3.2 Parameter Estimation

The basic approach is to first estimate the model parameters, Θ, (off-line) from an independent set of deflection measurements, and then, incorporate them into the *BSP*

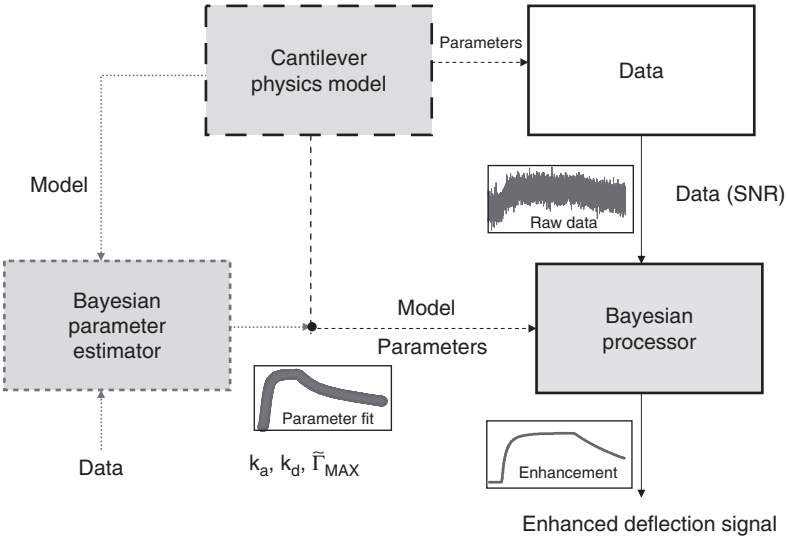


FIGURE 10.18 Physics-based approach to microcantilever "smart sensor" design: physics evolution model, parameter estimation (off-line) data, Bayesian processor and data enhancement.

to enhance the experimental (proof-of-concept) data. That is, we extract the critical absorption, desorption rate constants and maximum concentration parameters for each channel as: $\Theta_\ell = [k_a(\ell)\ k_d(\ell)\ \widetilde{\Gamma}_{MAX}(\ell)]$. The parameter estimator employed was a nonlinear least-squares solution using the Nelder-Meade polytope search algorithm [43]. This algorithm is based on minimizing

$$\min_{\Theta_\ell} J(\Theta) = \sum_{t=1}^{N_t} \varepsilon_\ell^2(t;\Theta) \quad \text{for } \varepsilon(t;\Theta) := y_\ell(t) - \hat{y}_\ell(t;\Theta) \tag{10.46}$$

where the estimated cantilever measurement at the ℓ^{th}-lever is given by

$$\hat{y}_\ell(t;\hat{\Theta}) = \Delta\hat{z}_\ell^C(t;\hat{\Theta}) + \Delta\hat{z}^T(t) = \beta_\ell \Gamma(t;\hat{\Theta})\Delta G(t) + \Delta\hat{z}^T(t) \tag{10.47}$$

We executed this estimator on raw experimental deflection data and estimated the parameters for each lever. The extracted parameters reasonably predicted the filtered cantilever response and the resulting error was uncorrelated as discussed below.

10.3.3 Bayesian Processor Design

Next using these estimated parameters, we developed *BSP* for the multichannel deflection data. The *GM* model was used to synthesize the multichannel data and provide known truth data to "tune" or adjust the processor noise covariance matrices that provide the "knobs" for *BSP* design (see Fig. 10.18 and [17] for details). Since the

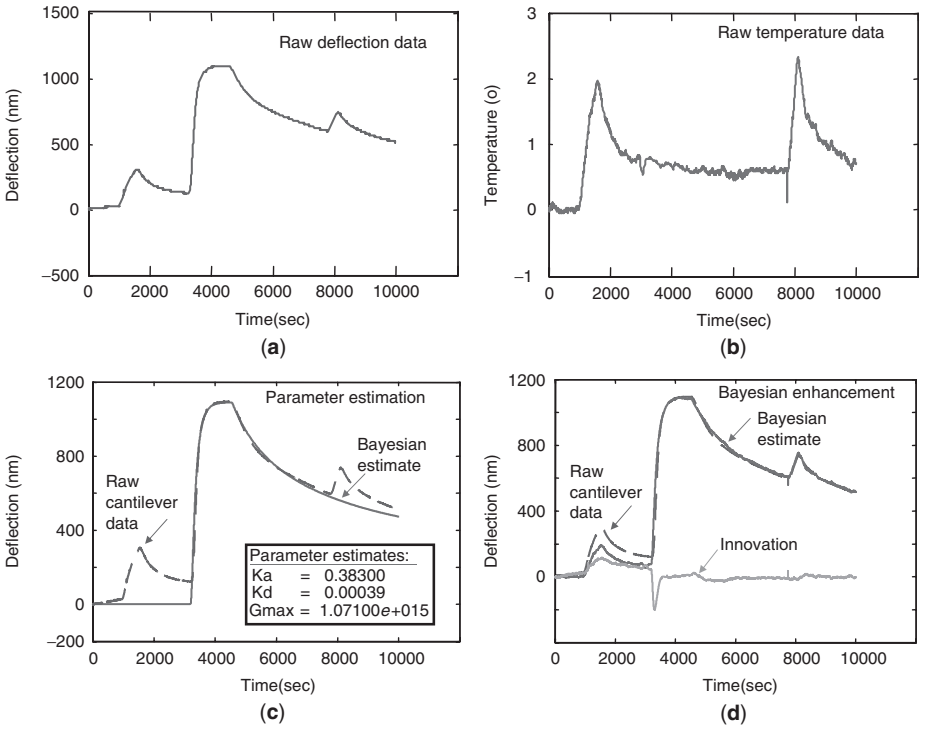


FIGURE 10.19 Bayesian processing for the "average" microcantilever sensor array signal: (a) Raw deflection data. (b) Raw temperature data. (c) Parameter estimation. (d) Bayesian estimation (enhancement).

simulation model and that used in the processor are identical, optimal (minimum error variance) performance is achieved (zero-mean, uncorrelated errors) providing the starting point for application to the experimental data. The final (simplified) *BSP* algorithm (assuming the parameters, $\hat{\Theta}$, have been estimated) is shown below. Here we see that the algorithm has a classical predictor-corrector form where the prediction estimates (conditional mean) the free energy, filters the measurement, estimates the innovation and corrects the final (filtered) free energy estimate.

$$\begin{aligned}
\Delta\hat{G}(t|t-1) &= \Delta\hat{G}(t-1|t-1) && \text{[Free Energy Prediction]}\\
\hat{\mathbf{y}}_\ell(t|t-1) &= \beta_\ell\Gamma(t;\hat{\Theta})\Delta\hat{G}(t|t-1)+\Delta\hat{z}^T(t) && \text{[Deflection Prediction]}\\
\boldsymbol{\varepsilon}_\ell(t) &= \mathbf{y}_\ell(t)-\hat{\mathbf{y}}_\ell(t|t-1) && \text{[Innovation or Residual]}\\
\Delta\hat{G}(t|t) &= \Delta\hat{G}(t|t-1)+\mathbf{k}'(t)\varepsilon(t) && \text{[Free Energy Correction]}
\end{aligned} \tag{10.48}$$

where $\mathbf{k}$ is the corresponding weight or gain and $\hat{\Theta}$ is the output estimate from the parameter estimator.

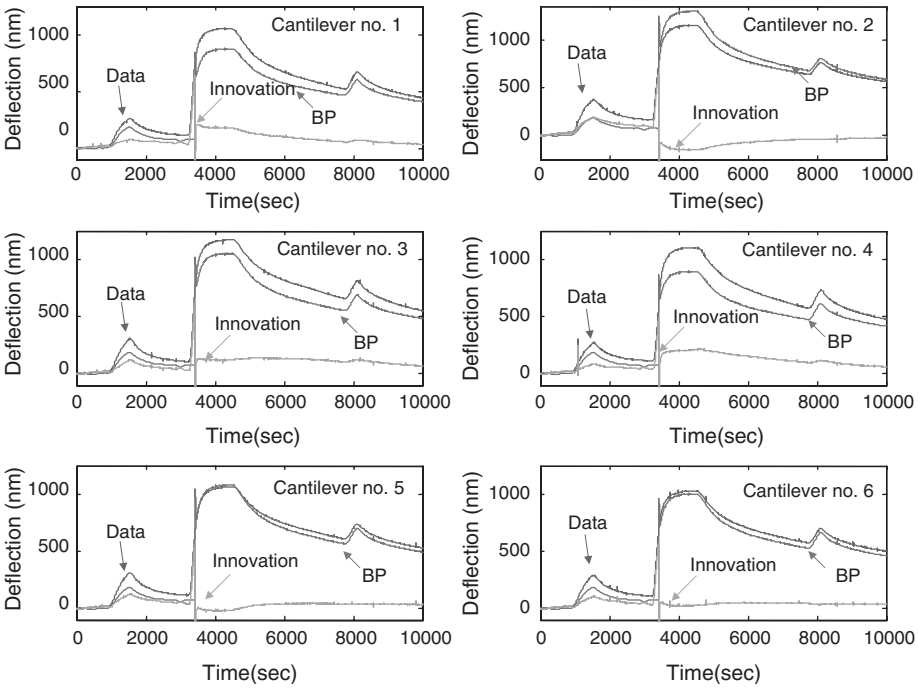


FIGURE 10.20 Bayesian processing of microcantilever experimental (proof-of-concept) data: Raw data, enhanced (*BP*) deflection measurement and residual results for each lever.

10.3.4 Results

First, we take the filtered measurement signals, *average* them to a single measurement, fit the parameters using an off-line optimization technique [43] and use the parameters in the Bayesian processor. The results are shown in Fig. 10.19 where we see the raw deflection and temperature measurements in a and b. The parameter estimator results are shown in Fig. 10.19c along with the state (signal) estimator in d. The parametric fit is quite reasonable as is the signal enhancement. However, it is clear from the innovations that the optimal processor is clearly not Gaussian, since it is not zero-mean or white. Next, we used the *BSP* with the free energy as our piecewise constant parameter (state) and the nonlinear cantilever array model with 6 elements along with a filtered estimate of the temperature profile in the processor, $\Delta\hat{z}^T(t)$. By tuning the measurement noise covariance parameters (R_{vv}) we demonstrate that the *BSP* is capable of tracking the noisy cantilever deflection data reasonably well; however, the performance is again suboptimal, since the innovations (shown in each figure), although quite small, are not uncorrelated. The results are shown in Fig. 10.20 where we see the raw measured cantilever data, Bayesian processor estimates and the corresponding residual errors or innovations. The results are quite reasonable

except for the systematic bias error in the processor output at each lever. The bias is created by out lack of knowledge of the initial concentration input and can easily be compensated (gain) at each lever as well. The dynamics appear to be captured by the model especially in cantilever 5. From the figure we note that the dynamics of the individual levers (on-set and off-sets) are quite close to the expected. This design demonstrates that even complex physical systems can be incorporated into physics-based processors enabling the development of a "smart" sensor.

10.4 BAYESIAN PROCESSING FOR THE DETECTION OF RADIOACTIVE SOURCES

With the increase in terrorist activities throughout the world, the need to develop techniques capable of detecting radioactive sources in a timely manner is a critical requirement. The development of Bayesian processors for the detection of contraband stems from the fact that the posterior distribution is clearly multimodal eliminating the usual Gaussian-based processors. The development of a sequential bootstrap processor for this problem is discussed and shown how it is capable of providing an enhanced signal for detection.

10.4.1 Background

Radionuclide source detection is a critical technology to detect the transportation of illicit radiological materials by potential terrorists. Detection of these materials is particularly difficult due to the inherent low-count emissions produced. These emissions result when sources are shielded to disguise their existence or, when being transported, are in relative motion with respect to the sensors. This section addresses the first step in investigating the problem of enhancing radionuclide signals from noisy radiation measurements using a Bayesian approach. Some work has been accomplished on this problem [44–48]. Here we model the source radionuclides by decomposing them uniquely as a superposition (union) of monoenergetic sources. Each γ-ray emitted is then smeared and distorted as it is transported on a path to the output of the detector for measurement and counting.

We start with the "physics-based" approach to solving this suite of problems and then discuss the measurement system employed to detect γ-rays and show how the mononenergetic approach leads to a compound Poisson driven Markov process [56] which is amplified, shaped and digitized for further processing. The processor is developed using state-space representations of the transition probability and associated likelihood and we apply it to synthesized data to evaluate its performance.

10.4.2 Physics-Based Models

Radiation detection is the unique characterization of a radionuclide based on its electromagnetic emissions. It has been and continues to be an intense area of research

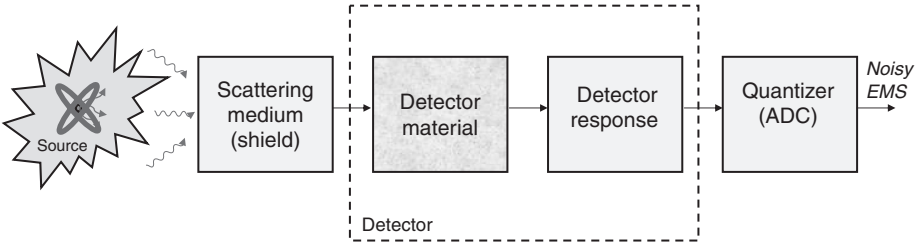


FIGURE 10.21 Gamma-ray evolution and measurement: radionuclide source (*EMS*), medium transport (physics), detector material interaction, detector temporal response (preamplification/pulse shaping) and A/D conversion with quantization noise.

and development for well over 50 years [57–61]. It is well known that a particular radionuclide can be uniquely characterized by two basic properties: its *energy* emitted in the form of photons or gamma-rays (γ-rays) and its radioactive *decay rate*. Knowledge of one or both of these parameters is a unique representation of a radionuclide. Mathematically, we define the pair, $[\{\epsilon_i\}, \{\lambda_i\}]$, as the respective energy level (MeV) and decay rate (probability of disintegration/nuclei/sec) of the i^{th}-component of the elemental radionuclide. Although either of these parameters can be used to uniquely characterize a radionuclide, only one is actually necessary—unless there is uncertainty in extracting the parameter. Gamma ray spectrometry is a methodology utilized to estimate the energy (probability) distribution or spectrum by creating a histogram of measured arrival data at various levels (counts vs. binned energy) [58]. It essentially decomposes the test sample γ-ray emissions into energy bins discarding the temporal information. The sharp lines are used to identify the corresponding energy bin "detecting" the presence of a particular component of the radionuclide. In the ideal case, the spectrum consists only of lines or spikes located at the correct bins of each constituent energy, ϵ_i, uniquely characterizing the test radionuclide sample.

Gamma-ray interactions are subject to the usual physical interaction constraints of scattering and attenuation as well as uncertainties intrinsic to the detection process. Energy detectors are designed to estimate the γ-ray energy from the measured electron current. A typical detector is plagued with a variety of extraneous measurement uncertainty that creates inaccuracy and spreading of the measured current impulse (and therefore γ-ray energy). The evolution of a γ-ray as it is transported through the medium and interacts with materials, shield and the detector is shown in Fig. 10.21. It is important to realize that in the diagram, the source radionuclide is represented by its constituents in terms of monoenergetic (single energy level) components and arrival times as $\xi(\epsilon_i, \tau_i)$. Since this representation of the source radionuclide contains the constituent energy levels and timing, then all of the information is completely captured by the sets, $[\{\epsilon_i\}, \{\tau_i\}]$, $i = 1, \ldots, N_\epsilon$. The arrivals can be used to extract the corresponding set of decay constants, $\{\lambda_i\}$ which are reciprocally related (1/mean rate). Thus, from the detector measurement of arrivals, or equivalently the so-called

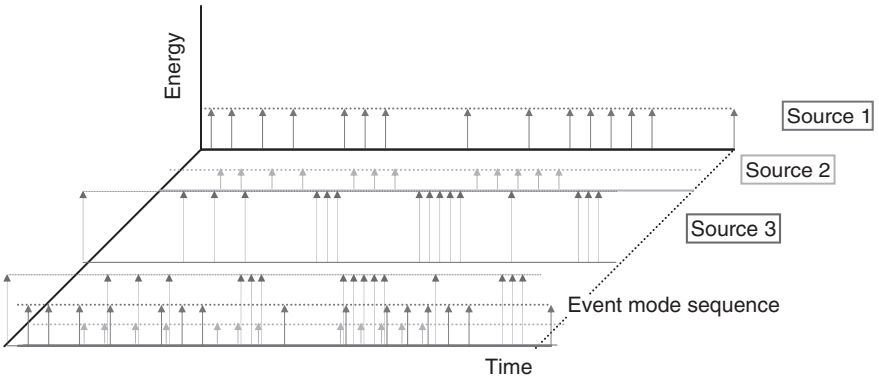


FIGURE 10.22 Monoenergetic source decomposition: individual source constituent *EMS* from ideal composite (superposition).

event mode sequence (*EMS*), a particular radionuclide can be uniquely characterized. The constituent energy levels (spikes), $\{\epsilon_i\}$ and arrival times, $\{\tau_i\}$, extracted from the *EMS* are depicted in Fig. 10.22, where we show the union (superposition) of each of the individual constituent monoenergetic sequences composing the complete radionuclide *EMS*. Note that there is no overlapping of arrivals—a highly improbable event.

So we see that the signal processing model developed from the transport of the γ-ray as it travels to the detector is measured and evolves as a distorted *EMS*. First, we develop a model of the event mode sequence in terms of its monoenergetic decomposition. Define $\xi(t;\epsilon_i,\tau_i,\lambda_i)$ as the component *EMS* sequence of the i^{th}-monoenergetic source at time t of *energy level* (amplitude), ϵ_i and *arrival time*, τ_i with *decay rate*, λ_i—as a single impulse, that is, $\xi(t;\epsilon_i,\tau_i,\lambda_i)=\epsilon_i\delta(t-\tau_i)$ and rate λ_i. Thus, we note that the ideal *EMS* is composed of sets of energy-time pairs, $\{\epsilon_i,\tau_i\}$. In order to define the entire emission sequence over a specified time interval, $[t_o,T)$, we introduce the set notation, $\underline{\tau}_i := \{\tau_i(1)\ldots\tau_i(N_\epsilon(i))\ \}$ at the n^{th}-arrival with $N_\epsilon(i)$ the total number of *counts* for the i^{th}-source in the interval. Therefore, $\xi(t;\epsilon_i,\underline{\tau}_i,\lambda_i)$ results in an unequally-spaced impulse train given by (see Fig. 10.22)

$$\xi(t;\epsilon_i,\underline{\tau}_i,\lambda_i)=\sum_{n=1}^{N_\epsilon(i)}\xi(t;\epsilon_i,\tau_i(n),\lambda_i)=\sum_{n=1}^{N_\epsilon(i)}\epsilon_i\delta(t-\tau_i(n)) \tag{10.49}$$

The *interarrival* time, is defined by $\triangle\tau_i(n)=\tau_i(n)-\tau_i(n-1)$ for $\triangle\tau_i(0)=t_o$ with the corresponding set definition (above) of $\underline{\triangle}\tau_i$ for $i=1,\ldots,N_\epsilon(i)-1$. Next we extend the *EMS* model from a single source representation to incorporate a set of N_ϵ-monoenergetic sources.

Suppose we have a radionuclide source whose *EMS* is decomposed into its N_ϵ-monoenergetic source components, $\xi(t;\epsilon,\tau,\lambda)$. From the composition of the *EMS* we know that

$$\xi(t;\epsilon,\tau,\lambda)=\sum_{i=1}^{N_\epsilon}\sum_{n=1}^{N_\epsilon(i)}\xi(t;\epsilon_i,\tau_i(n),\lambda_i)=\sum_{i=1}^{N_\epsilon}\sum_{n=1}^{N_\epsilon(i)}\epsilon_i\delta(t-\tau_i(n)) \tag{10.50}$$

Clearly, since the *EMS* is the superposition of Poisson processes, then it is also a composite Poisson process [56] with parameters: $\lambda=\sum_{i=1}^{N_\epsilon}\lambda_i, \epsilon=\sum_{i=1}^{N_\epsilon}\epsilon_i$, $N_\xi=\sum_{i=1}^{N_\epsilon}N_\epsilon(i)$ for λ the total decay rate, ϵ the associated energy levels and N_ξ the total counts in the interval, $[t_o,T)$. Note that the composite decay rate is the superposition of *all* of the individual component rates. This follows directly from the fact that the sum of exponentially (Poisson) distributed variables are exponential (Poisson). We note that the (composite) *EMS* of the radionuclide directly contains information about λ, but not about its individual components—unless we can extract the monoenergetic representation (Eq. 10.50) from the measured data.

Statistically, the *EMS* can be characterized by the following properties:

- non-uniform arrival time samples, $\tau_i(n)$
- monoenergetic source components, $\xi(t;\epsilon_i,\tau_i(n),\lambda_i)$ having their own unique decay rate, λ_i
- unique energy level, ϵ_i
- gamma distributed arrival times, $\tau_i(n)$, $\Gamma(k,\tau_i)$
- Poisson distributed counts, $N_\epsilon(i)$, $\mathcal{P}(N_\epsilon(n)=m)$
- exponentially distributed interarrival times, $\triangle\tau_i(n)$, $\mathcal{E}(\lambda_i\triangle\tau_i(n))$
- composite decay rate, λ

Next we consider the measurement of the *EMS* along with its inherent uncertainties.

10.4.3 Gamma-Ray Detector Measurements

Using the mathematical description of the *EMS* in terms of its monoenergetic source decomposition model discussed previously, we show how this ideal representation must be modified because of the distortion and smearing effects that occur as the γ-rays propagate according to the transport physics of the radiation process. Typically, these are quantified in terms of γ-ray spectral properties of energy "peak width" and "peak amplitude". The uncertainties evolve from three factors inherent in the material and instrumentation: inherent statistical spread in the number of charge carriers, variations in the charge collection efficiency and electronic noise [58]. In general, the energy resolution is defined in terms of a Gaussian random variable, $\epsilon_i\sim\mathcal{N}(\bar{\epsilon}_i,\sigma^2_{\epsilon_i})$.

Next we consider uncertainties created in the associated pulse processing system that consists of an amplifier and pulse shaping circuits. Here we concentrate on the amplitude output of the pulse shaper, since it carries not only the quantified γ-ray energy information, but also it is used for the detector timing circuits (gating pulses, logic pulses, etc.). The shaped pulse is converted to a logic pulse in order to extract the energy amplitude and precise timing information (arrival times, interarrival times, etc.). We consider the pulse shaper circuitry capable of taking the "raw" detector charge pulse, amplifying and shaping it to create a Gaussian pulse shape [58]. Once the Gaussian pulse amplitude, which is proportional to the original γ-ray energy (after scaling), is digitized or quantized by the analog-to-digital converter (*ADC*), the critical *EMS* parameters, $[\{\epsilon_i\}, \{\tau_i\}, \{\lambda_i\}]$, energy level, arrival time and decay rate can be extracted for further analysis and processing. From this data all other information can be inferred about the identity and quantity of the test radionuclide.

Next we define a *signal processing model* that captures the major characteristics of a solid state detector in order to formulate our Bayesian approach to the radiation detection problem. Consider the diagram again of the overall detector system shown in Fig. 10.21. Here we see how the γ-ray is transported through the medium (scattering and attenuation) to the detector. Each photon is deposited in the detector material, charge is collected and a charging current created. This current passes into measurement electronics that are also contaminated with random noise followed by the quantization to produce the noisy output measurement. Thus, from the i^{th}-monoenergetic component we have

$$\begin{aligned} p_{m_i}(t) &= \sum_{n=1}^{N_\epsilon(i)} \xi(t; \epsilon_i, \tau_i(n), \lambda_i) \star r(t) + w_{\tau_i}(t) \\ &= \sum_{n=1}^{N_\epsilon(i)} \epsilon_i r(t - \tau_i(n)) + w_{\tau_i}(t) \end{aligned} \tag{10.51}$$

where $r(t)$ is a rectangular window of unit amplitude defined within $\tau_i(n) \le t \le \tau_i(n-1)$. The uncertain (random) amplitude is Gaussian, $\epsilon_i \sim \mathcal{N}(\overline{\epsilon}_i, \sigma^2_{\epsilon_i})$, with inherent uncertainty representing the material charge collection process time "jitter" by the additive zero-mean, Gaussian noise, $w_{\tau_i} \sim \mathcal{N}(\overline{\tau}_i, \sigma^2_{w_{\tau_i}})$ and $\underline{\tau}(n) \to \tau_i(n)$; $n = 1, \ldots, N_\epsilon(i)$. Therefore, the material output pulse train for the i^{th}-source is given by $s(t) = H_S(t) \star p_{m_i}(t) + v(t)$. Extending the model to incorporate all of the N_ϵ-sources composing the radionuclide leads to the superposition of all of the monoenergetic pulse trains, that is, $p_m(t) = \sum_{i=1}^{N_\epsilon} p_{m_i}(t)$. The uncertain material pulse, $p_m(t)$, is then provided as input to the pulse shaping circuitry. Here the preamplifier and pulse shaper are characterized by a Gaussian filter with impulse response, $H_S(t)$ with output given by

$$s(t) = H_S(t) \star p_m(t) + v(t) \tag{10.52}$$

where the uncertainty created by instrumentation noise is modeled through the additive zero-mean, Gaussian noise source, $v \sim \mathcal{N}(0, \sigma_v^2)$. The shaped pulse is then quantized ($t_k \rightarrow t$) and digitally processed to extract the energy levels and timing information for further processing. Due to quantization limitations the *ADC* inherently contaminates the measured pulse with zero-mean, Gaussian quantization noise, $v_q(t_k)$ while there exists background radiation noise, $b(t_k)$ that must also be taken into account. At this point, we could also develop a signal processing model of the background, but we choose simplicity. We just simply model it as an additive disturbance at the output of the quantizer given by $b(t_k)$ giving us the final expression at the output of the quantizer as

$$z(t_k) = s(t_k) + b(t_k) + v_q(t_k) \tag{10.53}$$

with $v_q \sim \mathcal{N}(0, \sigma_q^2)$.

So we see that the entire *EMS* can be captured in a signal processing model with the key being the monoenergetic source decomposition representation of radiation transport. Next we start with this model and convert it to state-space Markovian form directly for Bayesian processing.

In our problem, the *EMS* is the noisy input sequence characterized by both input and noise processes, that is, ξ and $w_\tau \rightarrow w$. The states are part of the preamplifier and Gaussian pulse shaping system and the output is the quantized measurement, that is, $z(t_k) \rightarrow y(t)$. To be more specific, we use $\xi(t; \epsilon_i, \tau_i, \lambda_i)$, the i^{th}-monoenergetic source including both amplitude and timing uncertainties as a Poisson input to our Markovian model above along with the matrices, A, B, C, specifying the pulse shaping circuit parameters transformed to state-space form.

To see this consider the state-space representation for a *single* monoenergetic source is given by the following set of relations:

$$\begin{aligned}
\dot{x}_i(t) &= A_i x_i(t) + \mathbf{b}_i \xi(t; \epsilon_i, \underline{\tau}_i, \lambda_i) + \mathbf{w}_i w_{\tau_i}(t) \quad \text{[Source]} \\
y(t) &= \mathbf{c}_i' x_i(t) + v(t) \quad \text{[Pulse Shaper]} \\
z(t_k) &= y(t_k) + v_q(t_k); \quad i = 1, \ldots, N_\epsilon \quad \text{[ADC]}
\end{aligned} \tag{10.54}$$

Expanding this model over i to incorporate the N_ϵ-monoenergetic source components gives the extended state vector, $x(t) = [x_i(t) \mid x_2(t) \mid \ldots \mid x_{N_\epsilon}(t)]'$ where each component state is dimensioned N_x and therefore, $x \in \mathcal{R}^{N_x N_\epsilon \times 1}$. Thus, the overall *radiation detection* state–space model for N_ϵ monoenergetic sources is given by: $A = \text{diag}[A_i]$, $B = \text{diag}[B_i]$, $C = [\ \mathbf{c}_1' \mid \mathbf{c}_2' \mid \ldots \mid \mathbf{c}_{N_\epsilon}']$.

It is interesting to note some of the major properties of this model. The first feature to note is that the *monoenergetic decomposition* of the radionuclide source is *incorporated* directly into the model structure. For instance, it we are searching for a particular radionuclide and we know its major energy lines that uniquely describe its spectrum, we can choose the appropriate value of N_ϵ and specify its corresponding mean energy levels and decay rates directly—this is the physics-based approach.

We also note that the corresponding noise and statistics are easily captured by this structure as well. This formulation is a continuous-discrete or simply "sampled-data" model, since the *ADC* in used in the detection scheme.

10.4.4 Bayesian Physics-Based Processor

In this section we discuss the development of a Bayesian processor for a problem of enhancing a noisy *EMS* measurement with all of the information required "known" *a priori*. We demonstrate how a radiation detector can be modeled (simply) from a physics/statistical signal processing perspective, develop the mathematical representations and incorporate them into a Bayesian framework to enhance the constituent monoenergetic representation. We then demonstrate the Bayesian framework with an illustrative simulation.

A simple radiation transport synthesizer was developed for signal analysis purposes [62]. It consists of specifying the radionuclide in terms of its *EMS* and corresponding monoenergetic source decomposition then transporting this sequence through the medium (shield) along with its inherent scattering to the detector. At the detector the "surviving" or escaping γ-ray photons are transported through the detector material (semiconductor) again being absorbed and scattered with the final surviving photons providing the current pulse input to the shaping circuitry as shown in Fig. 10.21. After initializing the radionuclide and its corresponding monoenergetic source decomposition, the simulator transports the "ideal" *EMS* through the shield that incorporates both absorption (attenuation) and scattering (Compton) properties using the prescribed shield parameters. The output of this step is specified by the percentage of the photons escaping the shield and those captured or absorbed by the material and converted to thermal energy. The surviving photons escaping are then transported to the detector material where they undergo further absorption and scattering with the survivors converted to charge (electrons) provided as the input to the detector shaping circuitry.

To illustrate the Bayesian approach using physics-based signal processing models, we choose a single monoenergetic source sequence to represent a radionuclide with parameters, $\{\epsilon_o, \lambda_o, N_\epsilon(o)\}$ and generate the distortion and Gaussian smearing to synthesize the noisy detector output as illustrated previously in Fig. 10.21. Next we investigate the development of a sequential Bayesian processor for the following problem which can be stated formally as:

GIVEN a set of noisy γ-ray detector measurements, $\{z(t_k)\}$ and a set of *a priori* parameters $\{\epsilon_o, \lambda_o, N_\epsilon(o)\}$ or equivalently its state-space representation, $\Sigma_o = \{A_o, B_o, C_o\}$, along with a known (generated) *EMS*, $\{\xi_o(t)\}$, **FIND** best estimate of the underlying radionuclide *EMS*, $\{\hat{y}(t_k)\}$.

For our problem we assume we have a "good" synthetic model of the *EMS* and we construct the ideal physics-based processor with known parameters $\{\epsilon_o, \underline{\tau}_o, N_\epsilon(o)\}$ or equivalently known (generated by model) *EMS*. Note that we use the simplified

notation, $\xi_o(t) \rightarrow \xi(t;\epsilon_o,\underline{\tau}_o,\lambda_o)$. Therefore, the state–space representation is given by

$$\begin{aligned} \dot{x}_o(t) &= A_o x_o(t) + b_o \xi_o(t) + w_{\tau_o}(t) \quad \text{[Process]} \\ y(t) &= c_o' x_o(t) + v(t) \quad \text{[Measurement]} \\ z(t_k) &= y(t_k) + v_q(t_k) \quad \text{[ADC]} \end{aligned} \tag{10.55}$$

where $w_{\tau_o} \sim \mathcal{N}(0, R_{w_o w_o})$, $v \sim \mathcal{N}(0, R_{vv})$ and $v_q \sim \mathcal{N}(0, R_{v_q v_q})$. Under these linear assumptions with additive Gaussian noise processes, the optimal processor is the Kalman filter [17].

In order to develop the particle filter for this problem we require that the transition and likelihood distributions; therefore, under the modeling assumptions (Gaussian noise, known input, parameters, etc.), we have that:

$$\mathcal{A}(x(t)|x(t-1)) \sim \mathcal{N}(A_o x(t-1) + b_o \xi_o(t), R_{w_{\tau o} w_{\tau o}}(t-1))$$

$$\mathcal{C}(y(t)|x(t)) \sim \mathcal{N}(c_o' x(t), R_{vv})$$

Therefore, the *bootstrap* particle filter implementation for this problem for $i = 1, \ldots, N_p$ is:

- Draw: $x_i(t) \sim \mathcal{A}(x(t)|x_i(t-1))$; $w_i \sim \Pr(w_i(t))$;
- Weight: $W_i(t) = \mathcal{C}(y(t)|x(t))$;
- Normalize: $\mathcal{W}_i(t)$;
- Resample: $\hat{x}_i(t) \Rightarrow x_i(t)$;
- Posterior: $\hat{\Pr}(x(t)|Y_t) \approx \sum_i \mathcal{W}_i(t)\delta x(t) - \hat{x}_i(t)$;
- Inferences: $\hat{x}(t|t), \hat{x}_{MAP}(t)$.

This completes the formulation and Bayesian processor realizations both for the Kalman and particle filter designs, next we synthesize a radiation detection problem and apply the processors.

Suppose we have a nuclide represented by a single monoenergetic source of energy level, $\epsilon_o = 3.086\,keV$. Using the transport simulator with the following Gaussian noise variances: $R_{ww} = 10^{-6}$ and $R_{vv} = 10^{-2}$, we generated a realization of the noisy *EMS*. Next we construct the *EMS* signal enhancer and the results are shown in Fig. 10.23 where we observe the raw synthesized data illustrated along with the enhanced Bayesian processor estimates (both conditional mean and maximum a-posteriori). We see the enhanced *EMS* signal in (a) along with a zoomed version to observe the actual enhancement. Note the zero amplitude level noise has been minimized as part of the enhancement process. The optimal, X_{opt} (Kalman filter), and particle filter inferences for both conditional mean and maximum a-posteriori are annotated in Fig. 10.23; however, all of the realizations overlay one another so they are hard to differentiate.

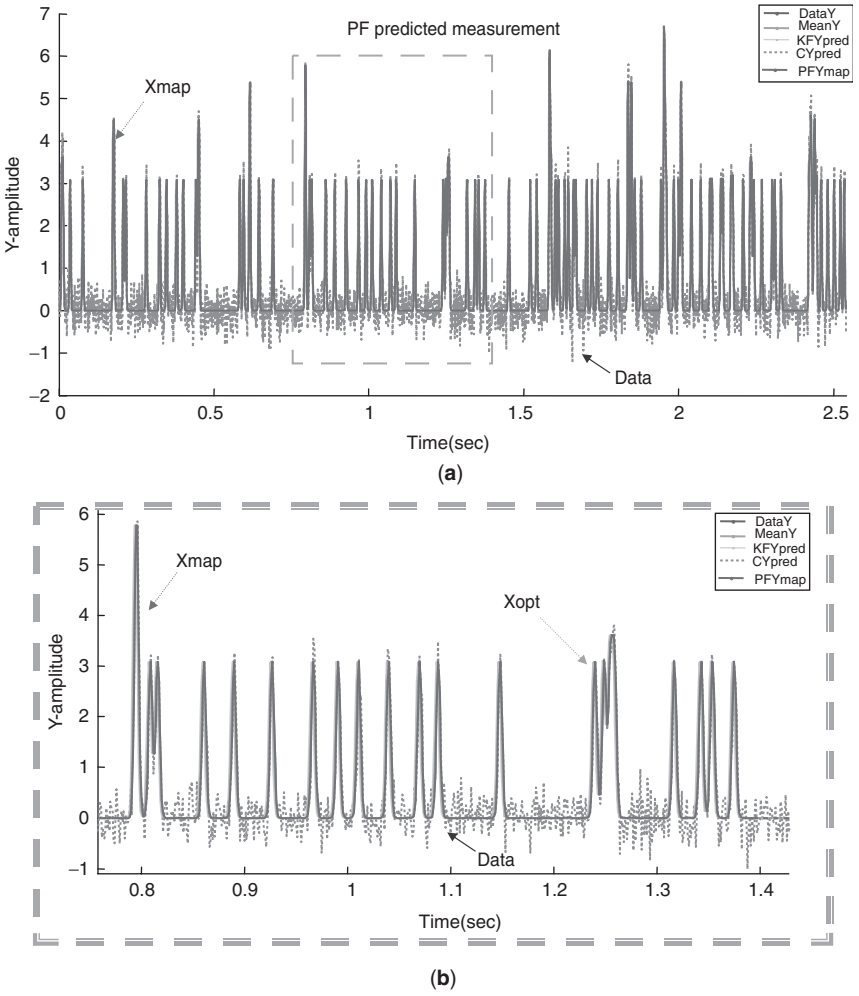


FIGURE 10.23 Bayesian Processor for radiation detection signal enhancement. (a) Entire *EMS* enhancement with box annotating zoom area. (b) Zoomed *EMS* with raw and enhanced processor outputs.

10.4.5 Physics-Based Bayesian Deconvolution Processor

In this section we consider extending the *BP* 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 [63] as well as transient problems [17, 64].

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

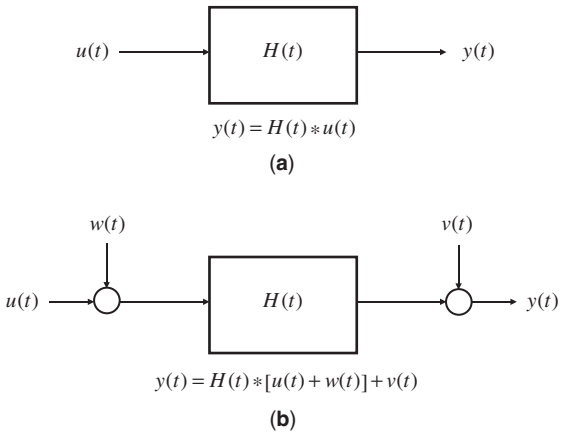


FIGURE 10.24 Model-based deconvolution problem: (a) Deterministic problem. (b) Stochastic problem (Gauss-Markov formulation).

depicted in Fig. 10.24*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 Fig. 10.24*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.

The solution to this problem using the Bayesian processor involves developing a model for the input and augmenting the state vector [17]. Suppose we utilize a discrete Gauss-Markov model and augment the following Gauss-Markov model of the input signal:

$$u(t) = F(t-1)u(t-1) + n(t-1) \tag{10.56}$$

where $n \sim \mathcal{N}(0, R_{nn}(t))$. The augmented Gauss-Markov model is given by $X_u := [x' \mid u']'$ and $w'_u := [w \mid n]$:

$$X_u(t) = A_u(t-1)X_u(t-1) + w_u(t-1) \tag{10.57}$$

and

$$y(t) = C_u(t)X_u(t) + v(t) \tag{10.58}$$

The matrices in the augmented model are given by

$$A_u(t-1) = \begin{bmatrix} A(t-1) & B(t-1) \\ 0 & F(t-1) \end{bmatrix}; \quad R_{w_u} = \begin{bmatrix} R_{ww}(t-1) & R_{wn}(t-1) \\ R_{nw}(t-1) & R_{nn}(t-1) \end{bmatrix}$$

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 [64]. The *BP* for this problem is the standard *Kalman filter* Bayesian algorithm with the augmented matrices given by the equations:

$$\underline{\text{State prediction}}: \hat{X}_u(t|t-1) = A_u\hat{X}_u(t-1|t-1)$$

$$\underline{\text{Innovation}}: e(t) = y(t) - \hat{y}(t|t-1) \text{ where } \hat{y}(t|t-1) = C_u\hat{X}_u(t|t-1)$$

$$\underline{\text{State correction}}: \hat{X}_u(t|t) = \hat{X}_u(t|t-1) + K(t)e(t)$$

with $K(t)$, the Kalman gain calculated using the state error and innovations covariance matrices where $\hat{X}_u(t|t) := E\{X_u(t)|Y_t\}$, that is, the conditional mean estimate of the augmented state given all of the previous data up to time t. Note that this is an optimal estimator under Gaussian assumptions (see Candy [17] for details).

One approach to estimating the unknown input sequence, $u(t)$ is to use a Taylor-series representation [64] given by

$$u(t + \Delta T) = \alpha_0 + \alpha_1\left(\frac{\Delta T}{1!}\right) + \alpha_2\left(\frac{\Delta T^2}{2!}\right) + \text{H.O.T.} \tag{10.59}$$

where $\alpha_i = \frac{d^i u(t)}{dt^i}$ for $u(t + \Delta T) \approx \sum_i \alpha_i \left(\frac{\Delta T^i}{i!}\right)$.

For our problem [47, 48] we consider two cases: the composite system of the pre-amplifier and pulse shaping circuits; and (ii) the pre-amplifier subsystem. In case (i) we assume all of the required information about the *EMS* is available at the output of the pulse shaping circuitry and the quantifier (ADC) merely extracts the maximum amplitude of the Gaussian shaped pulse and corresponding arrival times, $[\{\xi_i\}, \{\tau_i\}]$. However, we also consider case (ii) where we measure the output of the pre-amplifier (separately). Here the energy deposited by the γ-ray and subsequent charge curve reveals more detailed information about the γ-ray physics (arrival times, multiple arrivals, etc.). We digitize the actual pre-amplifier output generating a time series of the pulse and then perform the deconvolution to extract an "enhanced" γ-ray pulse through

the recovered (deconvolved) charging curve leading to the enhanced *EMS*. Once the *EMS* is recovered, the inherent photon information can be extracted (amplitudes and arrivals) and counted or employed as the input to a parameter estimator capable of providing improved energy (amplitude) estimates and corresponding arrival times while minimizing the noise and uncertainty.

In this section we concentrate on the model and deconvolved *EMS*. We accomplish the deconvolution by performing a system identification [17] of both pre-amplifier and pulse shaper to obtain transfer function estimates and then incorporate these estimates in the deconvolution algorithm. In this manner we will eventually be able to construct the final Bayesian sequential processor.

Once the deconvolved *EMS* is available from the processor, a Bayesian detector can be constructed to "decide" whether or not the threat radionuclide is present. If we assume the deconvolved and enhanced *EMS* is captured by $\hat{\xi}(t;\epsilon,\tau,\lambda)$. Thus, the binary detection problem is defined by testing the hypotheses

$$\begin{aligned} \mathcal{H}_o &: y(t) = \nu(t) && \text{[Noise]} \\ \mathcal{H}_1 &: y(t) = \hat{\xi}(t;\epsilon,\tau,\lambda) + \nu(t) && \text{[Signal + Noise]} \end{aligned} \tag{10.60}$$

with $\nu \sim \mathcal{N}(0, R_{\nu\nu})$. Thus, under the Neyman-Pearson criterion the optimal sequential decision function is the log-likelihood ratio given by [65]

$$\Lambda_\xi(t) = \Lambda_\xi(t-1) + \ln \Pr(y(t)|\mathcal{H}_1) - \ln \Pr(y(t)|\mathcal{H}_o) \tag{10.61}$$

here the distributions are specified by the inherent statistics associated with the *EMS* that still must be determined. Our approach will be to develop particle filters capable of estimating the appropriate posterior distributions. The *sequential detector* is therefore given by

$$\Lambda_\xi(t) \underset{\mathcal{H}_o}{\overset{\mathcal{H}_1}{\gtrless}} \tau_\xi \tag{10.62}$$

Ultimately, this scheme will be implemented to perform the radionuclide contraband detection.

10.4.6 Results

In this section we discuss the results of developing the models for both pre-amplifier and pulse shaper and applying them to perform the deconvolution operation [47], [48]. We injected a set of pulses into both subsystem components individually obtaining the required transfer functions and then developed the physics-based deconvolution processor as discussed in the previous section.

The test of the algorithm is on a simulated radionuclide (^{60}Co) *EMS* with 1.17 and 1.33 MeV lines generating the random detector input sequence. Here we convolved the

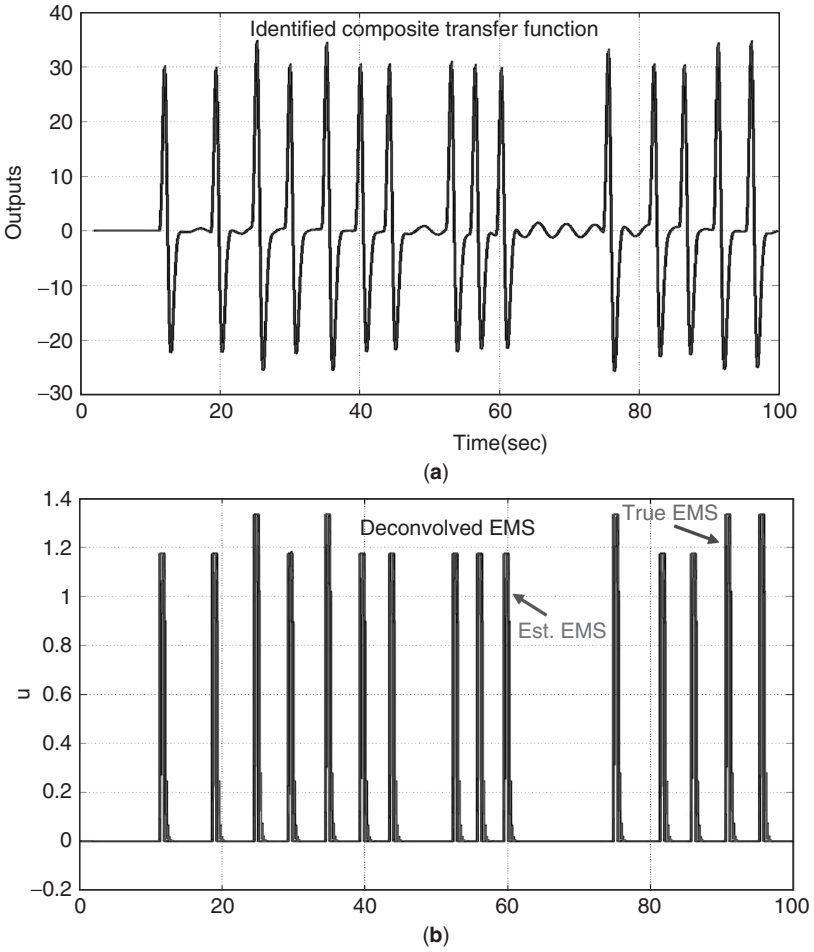


FIGURE 10.25 Bayesian deconvolution processor design. (a) Response estimate from system identification of composite (preamplifier and pulse shaper) system. (b) Deconvolution processing using identified impulse response (transfer function) with synthesized (known) *EMS* (^{60}Co: 1.17 and 1.33 MeV lines).

simulator output (deposited energy) with the identified composite transfer function. The results are shown in Fig. 10.25, where we see the transfer function validation run in (*a*) and the actual deconvolution processor output in (*b*). The processor is capable of extracting the *EMS* successfully and improving the overall γ-ray spectrum significantly as shown in Fig. 10.26. In (a) we see the "true" spectrum indicating two sharp energy lines at the correct energies (1.17 and 1.33 MeV), (*b*) the estimated (deconvolved) spectrum has captured the lines with some uncertainty (spreading shown), but its performance is quite reasonable and demonstrates the enhancement as observed from the measured spectrum of (*c*). Thus the Bayesian deconvolver works quite well

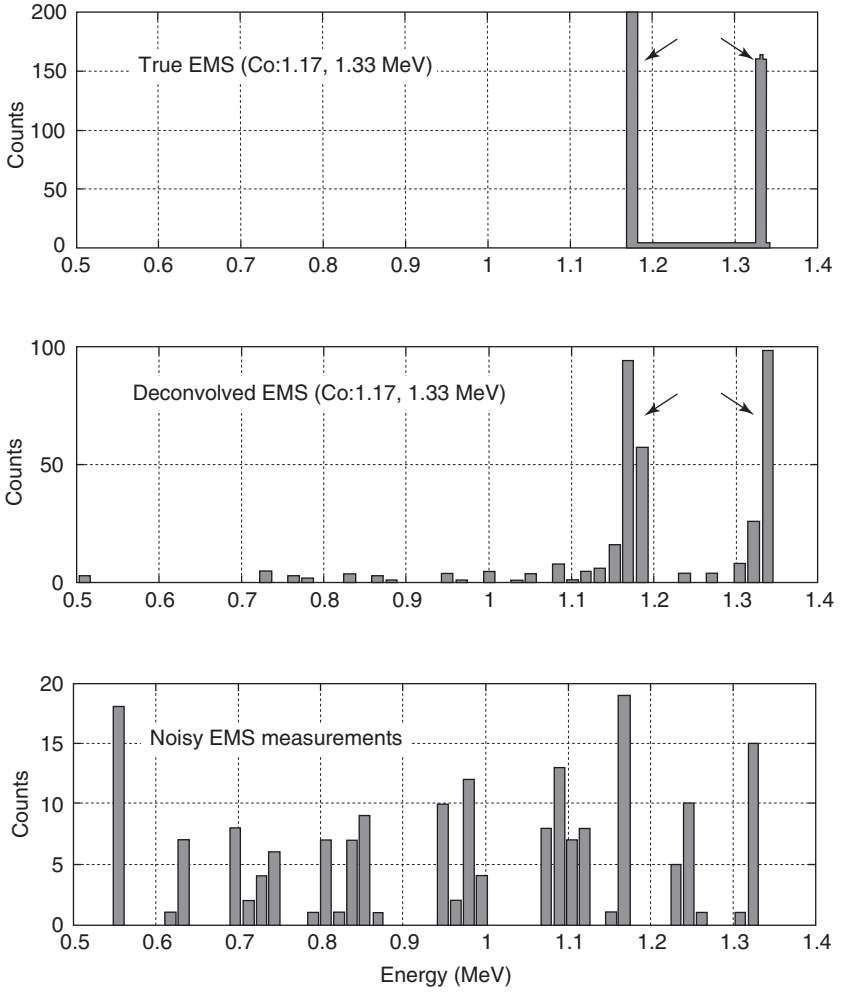


FIGURE 10.26 Deconvolution processor performance/enhancement (^{60}Co: 1.17 and 1.33 MeV lines). (a) Histogram of True *EMS* (synthesized). (b) Processed *EMS* histogram. (c) Raw (synthesized) measured detector output histogram.

on the synthesized data set. Thus, it appears that the processor can reliably extract the input excitation using this physics-based approach.

REFERENCES

1. D. Speck, E. Bliss, J. Glaze, J. Herris, F. Holloway, J. Hun, B. Johnson, D. Kuizenga, R. Ozarski, H. Patton, P. Ruppert, G. Suski, C. Swift and C. Thompson, "The Shiva laser-fusion facility," *IEEE J. Quantum Electr.*, **QE-17**, 9, 1599–1619, 1981.

2. J. Liu, H. Furuhashi, A. Torii, R. Sharma, V. Chitnis, B. Singh, J. Yamada and Y. Uchida, "Automatic mask alignment in the theta direction using moiré sensors," *Nanotechnology*, **6**, 135–138, 1995.
3. W. Blum, H. Kroha and P. Widmann, "A novel laser-alignment system for tracking detectors using transparent silicon strip sensors," *IEEE Trans. Nuclear Sci.*, **43**, 3, 1194–1199, 1996.
4. G. Seward, J. Leszcynski and E. Mulhern, "Rapid alignment of laser beams within optical systems," *Opt. Eng.*, **36** (5), 1414–1420, 1997.
5. A. Adolph, A. Boscheron, A. Dulac and E. Journot, "Final optics design for the megajoule laser," *SPIE Proceedings*, **3492**, pt. 1–2, 44–50, 1999.
6. W. He, Q. Chen, R. Xu, Z. Peng, H. Yang, C. Zhu and J. Zhao, "Image transfer based automatic alignment technique for laser-fusion facility," *Acta Optica Sinica*, **19**, 9, 1279–1283, 1999.
7. S. Roth, S. Schael and G. Schmidt, "A test of the laser alignment system ALMY at the TTF-FEL," *Nuclear Instr. Methods Phys. Res. A*, **475**, 537–544, 2001.
8. N. Fleurot, A. Adolf, M. Andre, J. Bruneau, C. Cavailler, M. Novaro, P. Philippe, F. Kovacs, B. Le Garrec, J. Di Nicola and J. Leidinger, "The ligne d'integration laser (LIL): construction status and first 1-w early results, *Proc. SPIE*, **4948**, 418–424, 2003.
9. D. Liu, J. Zhu, R. Zhu and D. Fan, "Laser beam automatic alignment in multipass amplifier," *Opt. Eng*., **43** (9), 2066–2070, 2004.
10. R. Zacharias, N. Beer, E. Bliss, S. Burkhart, S. Cohen, S. Button, R. Van Atta, S. Winters, J. Salmon, M. Latta, C. Stoiz, D. Pigg and T. Arnold, "Alignment and wavefront control systems of the National Ignition Facility," *Opt. Eng.*, **43** (12), 2873–2884, 2004.
11. J. Candy, W. Mcclay, A. Awwal and S. Ferguson, "Optimal position estimation for the automatic alignment of a high energy laser beam," *J. Optical Soc. Amer.*, **22**, 7, 1348–1356, 2005.
12. E. Moses, "The national ignition facility comes to life," *Science Technology Review*, LLNL Report, pp. 4–14, Sept. 2003.
13. F. Holderner, E. Ables, E. Bliss, S. Boege, R. Boyd, C. Chocol, D. Davis, R. Demaret, R. English, C. Laumann, J. Miller and S. Thomas, "Beam control and diagnostic functions in the NIF transport spatial filter," *Proceedings of SPIE **3047***, 692–699, 1997.
14. E. Bliss, F. Holderner, J. Salmon and J. Severyn, "Beam control and laser diagnostic systems," *LLNL Report*, UCRL-LR-105821-99-1, 79–97, 1999.
15. A. Awwal, W. Mcclay, W. Ferguson, J. Candy, T. Salmon and P. Wegner, "Composite amplitude modulated phase-only filter based detection and tracking of the back-reflection of KDP images," *Photonic Devices and Algorithms for Computing VI*, *Proc. of SPIE 5556*, 2004.
16. A. Jazwinski, *Stochastic Processes* (New York: Academic Press, 1970).
17. J. Candy, *Model-Based Signal Processing* (Hoboken, New Jersey: Wiley/IEEE Press, 2006).
18. K. Castleman, *Digital Image Processing* (Englewood Cliff's, NJ: Prentice-Hall, 1979).
19. A. Parvulescu, "Signal detection in a multipath medium by MESS processing," *J. Acoust. Soc. Am.*, **29**, 223–228, 1965.

20. C. Clay, "Optimum time domain signal transmission and source localization in a waveguide," *J. Acoust. Soc. Am.*, **81**, 660–664, 1987.
21. S. Li and C. Clay, "Optimum time domain signal transmission and source localization in a waveguide: experiments in an ideal wedge waveguide," *J. Acoust. Soc. Am.*, **82**, 1409–1417, 1987.
22. R. Brienzo and W. Hodgkiss, "Broadband matched-field processing". *J. Acoust. Soc. Am.*, **94**, 1409–1417, 1994.
23. 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**, (2), 571–587, 1988.
24. E. Westwood, "Broadband matched-field source localization," *J. Acoust. Soc. Am*, **91**, (5), 2777–2789, 1992.
25. T. Yang, "Broadband source localization and signature estimation," *J. Acoust. Soc. Am*, **93**, (4), 1797–1806, 1993.
26. I. Lu, H. Chen and P. voltz. "A matched-mode processing technique for localizaing a transient source in the time domain," *J. Acoust. Soc. Am*, **93**, (3), 1365–1373, 1993.
27. H. Chen and I. Lu, "Localization of a broadband source using a matched-mode procedure in the time-frequency domain," *IEEE Oceanic Engr.*, **19**, (2), 166–174, 1994.
28. J. Candy and E. Sullivan, "Ocean acoustic signal processing: a model-based approach." *J. Acoust. Soc. Am.*, **92**, (12), 3185–3201, 1992.
29. J. Candy and E. Sullivan. "Broadband model-based processing for shallow ocean environments" *J. Acoust. Soc. Am.*, **104**, (1), 275–287, 1998.
30. J. Candy and P. Candy, "SSPACK_PC: A model-based signal processing package on personal computers," *DSP Applic.*, **2**, (3), 33–42, 1993 (see also **http://www.technisoft.net**.
31. MathWorks, "MATLAB User's Guide," *MathWorks Inc.*, 1993.
32. C. Clay and H. Medwin, *Acoustical Oceanography*. (New York: Wiley, 1977).
33. F. Jensen, W. Kuperman, M. Porter and H. Schmidt, *Computational Ocean Acoustics* (New York: AIP Press, 1994).
34. A. Robinson and D. Lee, *Oceanography and Acoustics* (New York: AIP Press, 1994).
35. F. Jensen and M. Ferla, "SNAP: the SACLANTCEN normal-mode acoustic propagation model," *SACLANTCEN Report*, **SM-121**, SACLANT Undersea Research Centre, La Spezia, Italy, 1982.
36. Y. Tang, J. Fang, X. Xu, H. Ji, G. Brown, and T. Thundat, "Detection of femtomolar concentrations of HF using SiO_2 microcantilever," *Analytical Chemistry*, **76**, 2478–2481, 2004.
37. M. Su, S. Li, and V. Dravid, "Microcantilever resonance-based DNA detection with nanoparticle probes," *Applied Physics Letters*, **82**, 3562–3564, 2003.
38. G. Muralidharan, A. Wig, L. Pinnaduwage, D. Hedden, T. Thundat, and R. Lareau, "Absorption-desorption characteristics of explosive vapors investigated with microcantilevers," *Ultramicroscopy*, **97**, 433–439, 2003.
39. R. Raiteri, M. Grattarola, H. J. Butt, and P. Skladal, "Micromechanical cantilever-based biosensors," *Sensors and Actuators B-Chemical*, **79**, 115–126, 2001.

40. N. Lavrik, C. Tipple, M. Sepaniak and P. Datskos, "Gold nano-structures for transduction of biomolecular interactions into micrometer scale movements," *Biomedical Microdevices* **3**, 35–44, 2001.
41. J. Candy, D. Clague and J. Tringe, "A model-based processor design for smart microsensor arrays," *IEEE Signal Process. Magaz.*, **24**, 1, 125–126, 2007.
42. J. Tringe, D. Clague, J. Candy, A. Simensky, C. Lee, R. Rudd and A. Burnham, "Model-based processing of microcantilever sensor arrays," *IEEE Journal of MEMS Systems*, **15**, no. 5, 1379–1391, 2006.
43. P. Gill, W. Murray and M. Wright, *Practical Optimization* (New York: Academic Press, 1981).
44. C. Andrieu, E. Barat and A. Doucet, "Bayesian deconvolution of noisy filtered point processes," *IEEE Trans. Sig. Proc.*, **49**, 134–146, 2001.
45. A. Doucet and P. Duvaut, "Bayesian estimation of state space models applied to deconvolution of Bernoulli-Gaussian processes," *Signal Process.*, **57**, 147–161, 1997.
46. S. Gulam Razul, W. Fitzgerald and C. Andrieu, "Bayesian model selection and parameter estimation of nuclear emission spectra using RJMCMC," *Nucl. Instrum. and Methods in Physics Res.* A, **497**, 492–510, 2003.
47. K. Sale, J. Candy, E. Breitfeller, B. Guidry, D. Manatt, T. Gosnell and D. Chambers, "A Bayesian sequential processor approach to spectroscopic portal system decisions," *Proc. SPIE **670701***, 1–23, 2007.
48. J. Candy, K. Sale, B. Guidry, E. Breitfeller, D. Manatt, D. Chambers and A. Meyer, "Bayesian processing for the detection of radiactive contraband from uncertain measurements," *Proc. IEEE CAMSAP 07*, 2–15, 2007.
49. A. Doucet, N. de Freitas and N. Gordon, *Sequential Monte Carlo Methods in Practice* (New York: Springer-Verlag, 2001).
50. J. Liu, *Monte Carlo Strategies in Scientific Computing* (New York: Springer-Verlag, 2001).
51. B. Ristic, S. Arulampalam and N. Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Applications* (Boston: Artech House, 2004).
52. S. Godsill and P. Djuric, "Special Issue: Monte Carlo methods for statistical signal processing." *IEEE Trans. Signal Proc.*, **50**, 173–499, 2002.
53. M. Arulampalam, S. Maskell, N. Gordon and T. Clapp. Á tutorial on particle filters for online nonlinear/non-gaussian Bayesian tracking, *IEEE Trans. Sig. Process.*, **50**, 2, 174–188, 2002.
54. P. Djuric, J. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. Bugallo and J. Miguez, "Particle Filtering." *IEEE Signal Proc. Mag.* **20**, No. 5, 19–38, 2003.
55. A. Doucet and X. Wang, "Monte Carlo methods for signal processing," *IEEE Signal Proc. Mag.* **24**, No. 5, 152–170, 2005.
56. D. Snyder and M. Miller, *Random Point Process in Time and Space* (New York: Springer-Verlag, 1991).
57. R. Evans, *The Atomic Nucleus* (New York: McGraw-Hill, 1985).
58. G. Knoll, *Radiation Detection and Measurement, 3rd Ed.* (Hoboken NJ: John Wiley, 2000).
59. G. Gilmore and J. Hemingway, *Practical Gamma-Ray Spectrometry* (Hoboken NJ: John Wiley, 2003).
60. S. Labov et al., "Foundations for improvements to passive detection systems," LLNL Report, UCRL-TR-207129, 2004.

61. L. Meng and D. Ramsden, "An inter-comparison of three spectral-deconvolution algorithms for gamma-ray spectroscopy," *IEEE Trans. Nucl. Sci.*, **47**, 4, 1329–1336, 2000.

62. D. Chambers, "Signal processing model for radiation transport," *LLNL Report*, LLNL-TR-405952, 2008.

63. J. Mendel, *Maximum Likelihood Deconvolution* (New York: Springer-Verlag, 1990).

64. J. Candy and J. Zicker, "Deconvolution of noisy transient signals: A Kalman filtering application," *Proc. IEEE CDC Confr.* and *LLNL Report*, UCID-87432, 1982.

65. J. Candy, E. Breitfeller, B. Guidry, D. Manatt, K. Sale, D. Chambers, M. Axelrod, A. Meyer, "Bayesian sequential detection of radioactive contraband: A physics-approach," *LLNL Report*, LLNL-JRNL-408506, 2008.

Appendix A

PROBABILITY AND STATISTICS OVERVIEW

A.1 PROBABILITY THEORY

Defining a sample space (outcomes), Ω, 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, then we see that

$$\begin{array}{lll} \text{Sample space:} & \Omega & = \{H, T\} \\ \text{Events:} & B & = \{0, \{H\}, \{T\}\} \\ \text{Probability:} & \Pr(H) & = p \\ & \Pr(T) & = 1-p \end{array}$$

△△△

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 Ω, which consists of all the possible outcomes of the experiment. A random variable X and its realization x are written as

$$X(\omega) = x \quad \text{for } \omega \epsilon \Omega \tag{A.1}$$

Consider the following example of a simple experiment.

Bayesian 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,

$$\Omega = \{0\ H\ T\} \Longrightarrow X(\omega) = x$$
$$\omega = \{H, T\}$$

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. $\triangle\triangle\triangle$

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

$$\mathrm{P}_X(x_i) = \textstyle\sum_i \mathrm{F}_X(x_i)\delta(x - x_i) \tag{A.4}$$
$$\mathrm{F}_X(x_i) = \textstyle\sum_i \mathrm{P}_X(x_i)\mu(x - x_i) \tag{A.5}$$

where δ, and μ 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,

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

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 either of these functions, we can calculate probabilities that the random variable takes on values in any set of events on the 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

$$
\begin{array}{lll}
\text{Sample space:} & \Omega & = \{H, T\} \\
\text{Events:} & B & = \{0, \{\mathrm{H}\}, \{\mathrm{T}\}\} \\
\text{Probability:} & \mathrm{P}_X(x_1 = H) & = p \\
 & \mathrm{P}_X(x_0 = T) & = 1 - p \\
\text{Random variable:} & X(\omega_1 = H) & = x_1 = 1 \\
 & X(\omega_2 = T) & = x_2 = 0 \\
\text{Distribution:} & \mathrm{F}_X(x_i) & = \begin{cases} 1 & x_i \geq 1 \\ 1 - p & 0 \leq x_i \leq 1 \\ 0 & x_i < 0 \end{cases}
\end{array}
$$

the mass and distribution functions for this example are shown in Fig. 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 ω:

$$X(t, \omega) \quad \omega \epsilon \Omega, \quad t \epsilon T \tag{A.6}$$

where T is a set of index parameters (continuous or discrete) and Ω is the sample space.

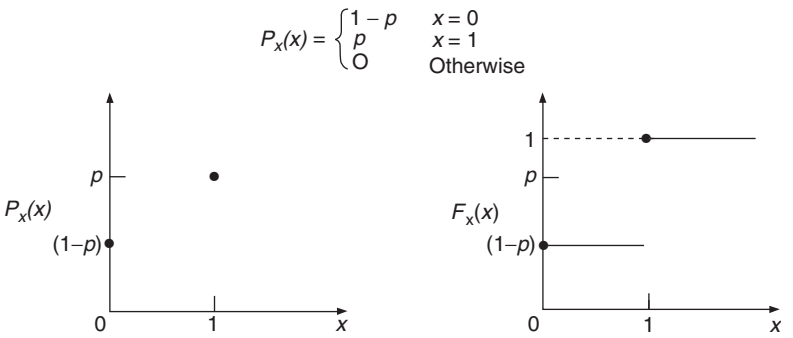


FIGURE A.1 Probability mass and distribution functions for coin-tossing experiment.

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, *auto-covariance* given by

$$R_{xx} = E\{(X - m_x)^2\}$$

The basic statistical measures are called *ensemble statistics* because they are measured across the ensemble ($i = 1, 2, \ldots$) 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:

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:	$R_{xx} =$	$E\{(X_i - m_x)^2\}$	$= \sum_i (X_i - m_x)^2 \mathrm{P}_X(x_i)$
Covariance:		R_{xy}	$= E\{(X_i - m_x)(Y_j - m_y)\}$
Standard Deviation:		σ_{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:

Linearity:	$E\{ax + b\}$	$= aE\{x\} + b = am_x + b$
Independence:	$E\{xy\}$	$= E\{x\}E\{y\}$
Variance:	$R_{xx}(ax + b)$	$= a^2 R_{xx}$

[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.

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 $t = 0, 1, ...$ and calculate the mean for each value of time across ($i = 1, 2, \ldots$) the ensemble. Dealing with stochastic processes is similar to dealing with random variables except that we must account for the time indices (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 of 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 we have two random variables, x_1, and x_2 and we know that the latter has already assumed a particular value, the we can define the *conditional* probability mass function of x_1 given that $X(\omega_2) = x_2$ has occurred by:

$$\Pr(x_1 \mid x_2) := \mathrm{P}_X(X(\omega_1) \mid X(\omega_2) = x_2) \tag{A.9}$$

and 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' 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* [1–3],

$$\Pr(x_1, \ldots, x_N) = \Pr(x_1 \mid x_2, \ldots, x_N)\Pr(x_2 \mid x_3, \ldots x_N) \ldots \Pr(x_{N-1} \mid x_N)\Pr(x_N) \tag{A.13}$$

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

$$I(X_i; X_j) = E_{x_i x_j}\{I(x_i, x_j)\} = \sum_i \sum_j \mathrm{P}_X(x_i, x_j) I(x_i, x_j) \tag{A.16}$$

which 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

$$\mathbf{m}_z := E\{\mathbf{z}\} \quad \text{and} \quad \mathbf{R}_{zz} = \text{Cov}(\mathbf{z}) := E\{(\mathbf{z} - \mathbf{m}_z)(\mathbf{z} - \mathbf{m}_z)'\}$$

Certain properties of the Gaussian vectors are useful such as:

- *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

$$Y \sim \mathcal{N}(\mathbf{A}\mathbf{m}_z + \mathbf{b}, \mathbf{A}\mathbf{R}_{zz}\mathbf{A}') \tag{A.19}$$

- *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

$$\mathbf{m}_z = E\left\{ \begin{matrix} \mathbf{x} \\ \mathbf{y} \end{matrix} \right\} = \begin{bmatrix} \mathbf{m}_x \\ \mathbf{m}_y \end{bmatrix}; \quad \text{and} \quad \mathbf{R}_{zz} = \text{Cov}(\mathbf{z}) = \begin{bmatrix} \mathbf{R}_{xx} & \mathbf{R}_{xy} \\ \mathbf{R}_{yx} & \mathbf{R}_{yy} \end{bmatrix}$$

  then the conditional distribution for $\mathbf{x}$ and $\mathbf{y}$ is also Gaussian with conditional mean and covariance given by

$$\begin{aligned} \mathbf{m}_{\mathbf{x}|\mathbf{y}} &= \mathbf{m}_x + \mathbf{R}_{xy}\mathbf{R}_{yy}^{-1}(\mathbf{y} - \mathbf{m}_y) \\ \mathbf{R}_{\mathbf{x}|\mathbf{y}} &= \mathbf{R}_{xx} - \mathbf{R}_{xy}\mathbf{R}_{yy}^{-1}\mathbf{R}_{yx} \end{aligned}$$

  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 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 $\mathbf{z} = \mathbf{T}\mathbf{x}$ and therefore

$$\mathbf{R}_{zz} = \text{Cov}(\mathbf{z}) = \text{Cov}((\mathbf{T}\mathbf{x})(\mathbf{T}\mathbf{x})') = \mathbf{T}\mathbf{R}_{xx}\mathbf{T}' = \mathbf{I} \tag{A.20}$$

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 semi-definite, 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

$$\mathbf{T} = \mathbf{U}^{-1} = \mathbf{R}_{xx}^{-1/2} \quad \text{(inverse matrix square root)} \tag{A.23}$$

and therefore

$$\mathbf{R}_{zz} = (\mathbf{R}_{xx}^{-1/2}\mathbf{U})\mathbf{U}'\mathbf{R}_{xx}^{-T/2} = \mathbf{R}_{xx}^{-1/2}\mathbf{R}_{xx}\mathbf{R}_{xx}^{-T/2} = \mathbf{I} \tag{A.24}$$

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 and S. Pillai, *Probability, Random Variables and Stochastic Processes* (New York: McGraw-Hill, 2002).
2. R. Hogg, J. McKlean and A. Craig, *Introduction to Mathematical Statistics* (Englewood Cliffs, NJ: Prentice-Hall, 2005).
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).

INDEX

Bayesian Signal Processing. By James V. Candy